

A
MAJOR PROJECT-IV REPORT
on
AGRI GUARDIAN - A Deep Learning Approach
For Rice Leaf Disease Detection

Submitted by:

D Veera Harsha Vardhan Reddy (210258)

Godavarthi Sai Nikhil (210214)

under mentorship of

Dr. Manisha Saini (Assistant Professor)

Dr. Himanshu Upreti (Assistant Professor)



Department of Computer Science Engineering

School of Engineering and Technology

BML MUNJAL UNIVERSITY, GURUGRAM (INDIA)

Dec 2024

CANDIDATE’S DECLARATION

I hereby certify that the work on the project entitled, “**AGRI GUARDIAN - A DEEP LEARNING APPROACH FOR RICE LEAF DISEASE DETECTION**”, in partial fulfillment of requirements for the award of Degree of **Bachelor of Technology** in School of Engineering and Technology at BML Munjal University, having University Roll Nos. **210258, 210214** is an authentic record of our work carried out during a period from **Aug 2024 to Dec 2024** under the supervision of **Dr. Manisha Saini** and **Dr. Himanshu Upreti**.

By-

D Veera Harsha Vardhan Reddy

Godavarthi Sai Nikhil

SUPERVISOR’S DECLARATION

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Faculty Supervisor Name: Dr. Manisha Saini

Signature:

Faculty Supervisor Name: Dr. Himanshu Upreti

Signature:

ACKNOWLEDGEMENT

I am highly grateful to **Dr. Manisha Saini and Dr. Himanshu Upreti**, Assistant Professors at BML Munjal University, Gurugram, for providing supervision to carry out the Major Project IV from August to December 2024.

Dr. Manisha Saini and Dr. Himanshu Upreti has provided great help in carrying out my work and is acknowledged with reverential thanks. Without wise counsel and able guidance, it would have been impossible to complete the project.

I extend my sincere thanks to **Dr. Manisha Saini and Dr. Himanshu Upreti** for their continuous encouragement throughout the project. I am also thankful to the entire team at BML Munjal University for their contributions. Additionally, I am grateful to my friends who generously dedicated their time and assistance, contributing significantly to the successful completion of our group project.

D Veera Harsha Vardhan Reddy
Godavarthi Sai Nikhil

List of Figures

| Figure No | Figure Description | Page No |
|------------------|---|----------------|
| Figure 3.1.1: | Categories of rice leaf diseases. | 20 |
| Figure 4.5.1 | Modular work Flow | 25 |
| Figure 4.5.2 | Faster RCNN Flow Process | 25 |
| Figure 4.5.3 | Flowchart for the YOLOv11 model architecture | 26 |
| Figure 4.5.4 | Flowchart for the EigenCAM Implementation | 26 |
| Figure 4.6 | Firestore Database for Agri Guardian | 27 |
| Figure 4.7.1 | YOLOv11 Model architecture | 28 |
| Figure 4.7.2 | Faster RCNN Model architecture | 29 |
| Figure 4.7.3 | Mobilenet v3 Large Model architecture | 32 |
| Figure 4.7.4 | DenseNet121 Model architecture | 32 |
| Figure 4.7.5 | ResNet 50 Model architecture | 33 |
| Figure 5.5.1 | Comparison of mAP for Different Models Across Datasets | 40 |
| Figure 5.5.2 | Results of Modified Faster R-CNN on Kaggle Dataset | 41 |
| Figure 5.5.3 | Detection Results of Modified Faster R-CNN on Datamnj Dataset | 41 |
| Figure 5.5.4 | AGRI GUARDIAN Application Interface – Home Screen | 42 |
| Figure 5.5.5 | AGRI GUARDIAN Application Interface – Disease Detection Results | 42 |

List of Tables

| Table No. | Table Description | Page No. |
|------------------|---|-----------------|
| Table 2.1 | Comparison across various Research Papers | 16 |
| Table 5.1.1 | Performance Metrics for YOLO v11 | 35 |
| Table 5.1.2 | Performance Metrics for Faster RCNN | 35 |
| Table 5.2.1 | Performance Metrics for Different Backbones on Kaggle Dataset | 36 |
| Table 5.2.2 | Performance Metrics for Different Backbones on Dataninja Dataset | 36 |
| Table 5.3.1 | Performance Metrics for ResNet-50 and Modified Model on Kaggle Dataset | 38 |
| Table 5.3.2 | Performance Metrics for ResNet-50 and Modified Model on dataninja Dataset | 38 |

List of Abbreviations

| Abbreviation | Full Form |
|---------------------|--|
| CAMs | Class Activation Maps |
| C2PSA | Channel and Spatial Attention |
| DL | Deep Learning |
| Eigen CAM | Eigen-Class Activation Map |
| Faster RCNN | Faster Region-Convolutional Neural Network |
| FPN | Feature Pyramid Network |
| IoU | Intersection over Union |
| mAP | Mean Average Precision |
| ROI | Region of Interest |
| RPN | Region Proposal Network |
| SPPF | Spatial Pyramid Pooling Fast |
| UI | User Interface |
| ViT | Vision Transformer |
| YOLO | You Only Look Once |

Table of Contents

| CONTENTS | Page no. |
|---|----------|
| Abstract | 9 |
| 1. Introduction | 10 |
| 1.1 Overview | 10 |
| 1.2 Existing System | 10 |
| 1.3 User Requirement Analysis | 11 |
| 1.4 Feasibility Study | 11 |
| 2. Literature Review | 13 |
| 2.1 Comparison | 16 |
| 2.2 Objectives | 18 |
| 3. Exploratory Data Analysis | 19 |
| 3.1 Dataset Description | 19 |
| 3.2 Data Split | 19 |
| 3.3 Data Preprocessing | 20 |
| 4. Methodology | 22 |
| 4.1 Intro Introduction to Languages | 22 |
| 4.2 Supporting Languages/Packages | 22 |
| 4.2.1 Additional Supporting Packages | 23 |
| 4.3 User Characteristics | 23 |
| 4.4 Constraints | 24 |
| 4.5 Flow Chart | 25 |
| 4.6 Database Design | 27 |
| 4.7 Algorithm Discussion | 27 |
| 4.7.1 YOLOv11 | 27 |
| 4.7.2 Faster RCNN | 29 |
| 4.7.3 Mobilenet v3 Large as Backbone For Faster RCNN | 31 |
| 4.7.4 Densenet121 as Backbone for Faster R-CNN | 32 |
| 4.7.5 ResNet-50 as Backbone for Faster R-CNN | 33 |
| 4.7.6 Eigen-CAM for Explainability in Faster R-CNN with ResNet-50 | 34 |
| 5. Results | 35 |
| 5.1 Performance Comparison of YOLO v11 and Faster R-CNN | 35 |
| 5.2 Optimization of Faster R-CNN Using Different Backbones | 36 |
| 5.3 Optimizing Faster R-CNN with ResNet-50 | 37 |
| 5.4 Integration of Eigen-CAM for Explainability | 39 |
| 5.5 Qualitative Analysis | 40 |

| | |
|--------------------------------|----|
| 6. Conclusion and Future Scope | 43 |
| 6.1 Conclusion | 43 |
| 6.2 Future Scope | 43 |
| References | 44 |
| Appendix | 46 |
| Plagiarism Check Report | 48 |
| Similarity | 48 |
| AI Check | 49 |

ABSTRACT

In the agricultural sector, the timely detection and management of crop diseases are crucial for ensuring food security and maximizing crop yields. Among various crops, rice holds significant importance as a staple food for a large portion of the global population. However, rice cultivation is particularly vulnerable to various diseases that can cause substantial losses in yield and revenue for farmers. Accurate and efficient identification of rice leaf diseases is essential for implementing targeted treatment strategies and minimizing crop damage.

To address these challenges, “**AGRI GUARDIAN**” a mobile application, has been developed as an integrated solution for rice leaf disease detection and management. The application utilizes a modified Faster R-CNN model with a ResNet-50 backbone, achieving a mAP score of 88.35. The detection model was fine-tuned using two datasets and optimized through comparative analyses, which demonstrated the superior performance of Faster R-CNN over YOLO v11, with ResNet-50 outperforming alternative backbones such as MobileNet_v3_Large and DenseNet121. To enhance model interpretability, Eigen-CAM was integrated, providing transparent visualization of the predictions.

Beyond disease detection, the application offers comprehensive features, including a cart system for purchasing agricultural inputs, detailed disease prevention guidelines, and financial tools to help farmers manage their resources effectively. By integrating explainable AI techniques with user-friendly functionality, this solution empowers farmers to make informed decisions, reduces crop losses, and contributes to global food security.

1. INTRODUCTION

1.1 OVERVIEW

In many parts of the world, rice is a staple food that provides essential nutrients to billions of people. However, rice fields are susceptible to numerous diseases, which can severely impact food security and significantly reduce productivity. Timely and accurate disease identification is crucial for implementing control measures and minimizing crop damage.

Advancements in technology have enabled innovative solutions, where deep DL and mobile application development converge to create powerful tools for rice leaf disease diagnosis. DL techniques allow for the extraction of complex patterns from large image datasets. By training models on extensive datasets of healthy and diseased rice leaves, these systems can precisely identify and classify diseases.

The Rise of Mobile-Based Rice Leaf Disease Detection:

Mobile-based applications, developed using frameworks like Flutter, support the deployment of these advanced DL models, providing a practical and intuitive interface for farmers. These applications allow users to diagnose rice leaf diseases directly in the field, ensuring timely and effective decision-making.

The integration of mobile technology with DL for rice leaf disease detection has the potential to transform the rice-growing sector in several ways:

- a) Improved food security through early detection and precise disease management.
- b) Giving farmers easy access to diagnostic tools.
- c) Supporting informed decision-making for effective disease prevention strategies.
- d) Promoting sustainable agricultural practices by lowering reliance on use of chemicals.

This development marks a significant step toward empowering farmers with efficient and accessible tools to protect their crops and livelihoods. The following sections provide a detailed overview of this approach, focusing on the creation of a mobile application for rice leaf disease detection by combining DL and modern application frameworks.

1.2 EXISTING SYSTEM

Traditionally, disease detection in crops, including rice, has relied heavily on manual inspection by agricultural experts or IoT-based solutions like JioKrishi. Manual methods are prone to inaccuracies and delays, often leading to suboptimal treatment strategies. While IoT devices like JioKrishi focus on soil-related parameters, they do not address leaf disease detection, which is critical for accurate diagnosis and timely intervention.

Moreover, most existing AI-based systems and applications lack explainability features, leaving users with limited understanding of the model's decision-making process. Although

some mobile applications aim to bridge this gap, they are often limited by low accuracy, insufficient datasets, or inadequate functionality to meet the diverse needs of farmers. These limitations underscore the necessity for a comprehensive, user-centric solution that integrates accurate disease detection with features such as disease management guidelines, resource purchasing options, and financial tools.

1.3 USER REQUIREMENT ANALYSIS

Understanding the demands and preferences of end customers is critical when developing an effective and user-friendly solution. Farmers, agricultural extension agents, and other crop management stakeholders are the primary users of rice leaf disease detection. Surveys and research were conducted to identify the specific needs, issues, and preferences of target users and assemble their requirements.

Key user needs include:

- a) **Accuracy:** High precision in disease detection to minimize treatment errors.
- b) **Accessibility:** A mobile-based solution that operates efficiently with limited hardware resources and is usable by individuals with varying levels of technical proficiency.
- c) **Real-time Detection:** The ability to detect conditions in real-time to reduce crop damage and promote quick responses.
- d) **User-Friendly Interface:** Simple navigation for users with varying levels of digital literacy.
- e) **Comprehensive Support and Offline Functionality:** Features like disease prevention guidelines, input procurement, and financial management tools are offline capabilities to ensure continuous use in rural areas with limited connectivity.

1.4 FEASIBILITY STUDY

A feasibility study was conducted to evaluate the technical, operational, and economic viability of implementing a mobile-based rice leaf disease detection solution.

- a) **Technical Feasibility:** A diverse range of rice leaf disease images is available through public sources and curated datasets, with collaborations enhancing dataset specificity for local diseases. While training deep learning models requires computational resources, platforms like university workstations and Google Colab provide adequate support. The solution is compatible with modern smartphones and cameras, ensuring high-resolution image capture. Faster R-CNN with a ResNet-50 backbone has proven highly effective for image recognition tasks, making it suitable for diagnosing rice leaf diseases. Eigen-CAM

was incorporated to improve interpretability, providing clear visualizations of affected regions.

- b) **Operational Feasibility:** The application was designed to perform efficiently even under limited connectivity and constrained hardware resources. It supports intuitive design principles, making it accessible to users with varying levels of digital literacy. A well-coordinated team proficient in DL concepts, combined with faculty support, ensures effective execution. With clear planning, efficient task distribution, and consistent team communication, the development process is manageable within the available timeline.
- c) **Economic Feasibility:** By leveraging open-source datasets and pre-trained models, development costs were minimized. The application's multifunctionality, encompassing disease detection, resource procurement, and financial tools, adds significant value. The approach addresses the needs of farmers, researchers, and agricultural professionals by offering an affordable, accessible, and comprehensive solution. While commercial alternatives exist, this solution uniquely integrates affordability, explainability, and user-centric features to reach a broader audience.

The study confirmed the practicality and scalability of this solution, making it suitable for addressing rice disease management challenges.

2. LITERATURE REVIEW

Rice Leaf Disease Detection Using Deep Learning

Recent studies have focused on applying DL algorithms for automating rice leaf disease identification. These methods offer significant advantages, including objectivity, precision, and speed. This study reviews various research papers and journals that have explored the use of DL techniques for diagnosing rice leaf diseases, highlighting the potential of neural networks and advanced architectures in improving diagnostic accuracy and efficiency.

Deep Learning Techniques for Rice Leaf Disease Detection

DL has gained significant recognition in object detection, particularly with architectures such as YOLO and Faster R-CNN. These models excel in identifying and localizing objects within images, providing precise and efficient solutions for complex detection tasks. YOLO and Faster R-CNN can learn intricate features and spatial relationships within images, enabling them to detect multiple objects in real time without the need for manual feature extraction.

a) YOLO Network:

The YOLO family of algorithms has seen significant advancements, with impactful applications in agricultural disease detection and remote sensing. For instance, the Pyramid-YOLOv8 algorithm by Cao et al. [1,3] integrates a multi-attention feature fusion network and a lightweight C2F-Pyramid module, achieving a mean Average Precision (mAP) of 84.3% and a detection speed of 62.5 FPS. While it outperforms models like Faster-RCNN and YOLOv3-SPP, challenges remain in detecting small and dense targets in complex datasets [1,3].

Another study by Maretanio et al. [2] employed YOLOv8n for estimating rice field areas using drone mapping, a critical aspect of smart agriculture. The model achieved respectable precision and recall scores of 0.759 and 0.752, respectively, but encountered significant errors, with a sample showing a 38% misdetection rate. These findings emphasize the potential of YOLOv8 for remote sensing tasks and suggest future improvements through expanded datasets and fine-tuning to address complex field conditions [2].

Sapkota et al. [4] further explored the versatility of YOLO-based models by evaluating YOLOv8, YOLOv9, YOLOv10, and YOLOv11 for fruitlet detection in orchards. YOLOv9 Gelan-base achieved the highest mAP@50 score of 0.935, excelling in precision, while YOLOv11s offered the fastest inference speed. These results underline the trade-offs between speed and accuracy, emphasizing that the choice of a YOLO variant should align with the task requirements, such as precision and recall for high-stakes agricultural applications or speed for real-time use cases [4].

In earlier work, Haque et al. [5] demonstrated the effectiveness of YOLOv5 in detecting rice leaf diseases, achieving 90% precision, 67% recall, and 81% mAP. The study highlighted

YOLOv5's superiority over traditional methods like SVM and VGG16, using metrics like confusion matrices and precision-recall curves for evaluation. However, the relatively low recall suggests room for improvement. This research underscores the progressive refinement of YOLO models for agricultural tasks, particularly in enhancing recall and adapting to diverse plant diseases [5].

These studies highlight the advancements in YOLO algorithms for agriculture, showcasing their potential in disease detection and field mapping. Despite their success, challenges like small target detection and improving recall remain, emphasizing the need for further innovations in future YOLO models.

b) Faster RCNN

The Faster R-CNN algorithm has been widely applied for object detection tasks, demonstrating remarkable accuracy and versatility. A. K. and A. S. Singh [6] utilized Faster R-CNN and Mask R-CNN to detect paddy crop diseases, achieving accuracies of up to 96% for rice blast diseases. While effective, their work highlighted limitations in handling datasets with diverse diseases, suggesting video-based inputs for future improvements. Sheng et al. [10] introduced CF-RCNN, an advanced Faster R-CNN variant integrating CBAM and FPN, to address challenges with occluded or small-scale objects, achieving a mAP of 76.2% on PASCAL VOC2012, albeit with increased computational costs.

Faster R-CNN's comparative performance has been explored extensively. Dalmar et al. [7] and Saieshan Reddy et al. [8] evaluated Faster R-CNN alongside other algorithms like YOLO and SSD. Dalmar et al. highlighted Faster R-CNN's highest accuracy mAP (61.2%) for small object detection, albeit at the cost of higher computational resource usage, while Reddy et al. demonstrated its edge in accuracy (99.58%) but noted its slower detection speed compared to alternatives like SSD and YOLO. These studies emphasize Faster R-CNN's robustness in precision-critical tasks, despite its limitations in real-time scenarios.

Wenbo Dong [9] and Avola et al. [12] examined specific applications of Faster R-CNN. Dong compared it with YOLOv3, demonstrating Faster R-CNN's higher mAP scores for accuracy-first tasks, though it lagged in real-time performance. Avola et al. developed MS-Faster R-CNN, a multi-stream variant tailored for aerial tracking using UAV images, achieving a state-of-the-art mAP of 97.3% but introducing scalability concerns due to increased computational complexity. These enhancements underline the adaptability of Faster R-CNN while addressing its constraints through architectural innovations.

Further insights into Faster R-CNN's foundational advancements were provided by Wenzhe Li [11] and Ismaili Alaoui et al. [13]. Li analyzed the integration of Region Proposal Networks (RPN) into Faster R-CNN, demonstrating significant improvements in both accuracy (8.2% mAP over Fast R-CNN) and speed (235 times faster test time compared to R-CNN). Ismaili

Alaoui and Hmidani presented a comprehensive survey of the R-CNN family, showing how Faster R-CNN reduces computational overhead by replacing selective search with RPN, making it 11 times faster than Fast R-CNN. These studies underscore the algorithm's evolution and its ongoing optimization for real-world applications.

Faster R-CNN has proven to be a robust and precise object detection algorithm, excelling in accuracy-focused applications while facing challenges in real-time scenarios. Innovations like CF-RCNN and MS-Faster R-CNN continue to refine its performance, expanding its applicability to domains like agriculture, UAV-based tracking, and high-precision object detection tasks [6,7,9,12]. However, its computational demands necessitate further research into lightweight models to balance speed and precision in real-time contexts [10,11,13].

c) Vision Transformers (ViT) for Classification:

One of the unique approaches is implemented by the researchers [13]. In their paper they compared three approaches: (A) CNNs with channel and spatial attention for improved feature extraction, (B) transfer learning with Inception V3 for leveraging pre-trained knowledge, and (C) ViT models (ViT-1 and ViT-2) with different hyperparameter configurations. They used Grad-CAM for visualizing disease locations in CNNs and a DINO technique for ViT attention visualization to understand which image regions the models focus on for classification. This multi-pronged approach aimed to achieve robust and accurate disease classification, surpassing existing methods, which they achieved with good enough to call it a success.

2.1 Comparison

Table 2.1: Comparison across various Research Papers

| Paper Title | Dataset | Methodology | Results | Limitations |
|--|---|---|--|--|
| Pyramid-YOLOv8: A Detection Algorithm for Precise Detection of Rice Leaf Blast [1] | Rice Leaf Blast Dataset | YOLOv8x framework, Multi attention feature fusion, C2F-Pyramid module | Achieved mAP of 84.3%, outperformed Faster-RCNN, RT-DETR, and YOLOv3-SPP with a detection speed of 62.5 FPS | Struggles with dense, small targets and real-time performance optimization |
| Estimation of Rice Field Area Using YOLO Method to Support Smart Agriculture System [2] | Drone Mapping Data | YOLOv8n for rice field detection, drone mapping for smart agriculture | Achieved 0.765 accuracy for detecting rice fields, precision and recall values of 0.759 and 0.752 respectively | Significant land detection errors (38%-56%), limited dataset causing missed detections |
| Pyramid-YOLOv8: A Detection Algorithm for Precise Detection of Rice Leaf Blast [3] | Rice Leaf Blast Dataset | YOLOv8x, Multi attention feature fusion, Lightweight C2F-Pyramid module | mAP of 84.3%, 62.5 FPS speed, reduced computational load | Limited accuracy in detecting dense and small target regions, further optimization needed |
| Comprehensive Performance Evaluation of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 on Fruitlet Detection [4] | Orchard Fruitlet Datas | YOLOv8, YOLOv9, YOLOv10, YOLOv11 for fruitlet detection | YOLOv9 Gelan-base and YOLOv11 achieved high mAP scores of 0.935 and 0.933, with YOLOv11 having the fastest inference speed (2.4ms) | Trade-off between precision, recall, and computational efficiency; YOLOv11's performance is best in speed but not always in accuracy |
| Rice Leaf Disease Classification and Detection Using YOLOv5 [5] | Rice Leaf Disease Dataset (1500 images) | YOLOv5 for rice leaf disease detection | 90% precision, 81% mAP, and 76% F1 score, outperformed methods like SVM, VGG16, and ResNet V2 | Recall rate improvement needed, requires high-quality images for optimal performance |
| Detection of Paddy Crops Diseases and Early Diagnosis Using Faster RCNN [6] | Rice disease dataset | Used Faster R-CNN and Mask RCNN for detecting diseases like Blast and Brown Spot in paddy crops. | Achieved detection accuracy of 96% (Blast) and 95% (Brown Spot). | Limited to detecting only a few diseases; dataset was not diverse. |
| A comprehensive survey of the R-CNN family for object detection [13] | Multiple datasets (PASCAL VOC, COCO) | Analyzed the evolution of the R-CNN family, comparing accuracy and efficiency improvements across | Faster R-CNN emerged as a benchmark for high accuracy with significant speed | Real-time performance remains a challenge despite improvements. |

| | | | | |
|--|------------------------------|--|--|--|
| | | variants like R-CNN, Fast R-CNN, and Faster R-CNN. | improvements over its predecessors. | |
| Comparative Study of Convolutional Neural Network Object Detection Algorithms for Image Processing [8] | PASCAL VOC and COCO | Explored the performance of CNN-based detection models, including Faster R-CNN and YOLO, across datasets. | Found Faster R-CNN more suitable for tasks requiring precision, particularly for overlapping object detection. | Computationally expensive, limiting real-time applications. |
| A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs. SSD vs. Faster R-CNN [7] | COCO | Compared YOLO, SSD, and Faster R-CNN for object detection tasks, focusing on mAP and inference speed. | Faster R-CNN outperformed YOLOv3 in mAP@0.5 - 61.2% with strong small object detection capabilities. | Slower inference speed and higher GPU memory usage compared to YOLO and SSD. |
| Faster R-CNN and YOLOv3: a general analysis between popular object detection networks [9] | PASCAL VOC | Compared Faster R-CNN and YOLOv3, analyzing strengths in accuracy and speed. | Faster R-CNN has better accuracy focused tasks like overlapping object detection. | Faster R-CNN is slower compared to YOLOv3, limiting its real-time utility |
| Faster RCNN Target Detection Algorithm Integrating CBAM and FPN [10] | PASCAL VOC | Incorporated CBAM and FPN modules into Faster R-CNN to improve small and occluded object detection. | Achieved a mAP of 76.2%, 13.9% higher than standard Faster R-CNN. | Increased computational cost, limiting its real-time feasibility. |
| Analysis of Object Detection Performance Based on Faster R-CNN [11] | PASCAL VOC and COCO | Evaluated Faster R-CNN against YOLO and SSD in terms of accuracy, speed, and feature capabilities. | Demonstrated consistent superiority in accuracy for detecting small and overlapping objects. | Struggles with maintaining real-time performance and high GPU requirements. |
| MS-Faster R-CNN: Multi-Stream Backbone for Improved Faster R-CNN Object Detection and Aerial Tracking from UAV Images [12] | UAV imagery (custom dataset) | Developed a multi-stream backbone to enhance Faster R-CNN for aerial object tracking. | Achieved state-of-the-art mAP of 97.3% for UAV applications, excelling in tracking and detection tasks. | Increased computational complexity and sensitivity to occlusion and lighting variations. |
| Vision Transformer Based Models for Plant Disease Detection and Diagnosis [14] | Plant Village | CNNs with channel and spatial attention for feature extraction, Transfer learning with Inception V3, and ViT models. | Achieved accuracies of 96.7%, 98.52%, 99.1% | Limited dataset details, challenges in real-world plant disease scenarios. |

2.2 Objectives

The goal of this project is to create a mobile application that detects rice leaf disease correctly and instantly using DL. We want to achieve these aims by providing farmers with access to this cutting-edge device.

- Develop a mobile application using Flutter, integrating a fine-tuned Faster R-CNN model with a ResNet-50 backbone for precise object detection of rice leaf diseases.
- Utilize pre-trained ViT results for reliable classification of six rice leaf categories, including healthy and diseased classes.
- Compare and evaluate YOLO v11 and Faster R-CNN models for object detection, implementing the most effective model for disease localization within rice leaf images.
- Offer disease-specific information with symptoms, causes, and prevention strategies, alongside a cart for purchasing agricultural inputs and a financial tool for tracking expenses and budgeting.
- Design an intuitive and accessible user interface suitable for farmers with varying levels of technological familiarity.
- Integrate Eigen-CAM visualization to enhance the explainability of detection results, fostering trust in AI-based recommendations.

This project aims to support food security by enhancing early disease detection and promoting improved crop management practices.

- Improved farmer empowerment by making diagnostic resources and information more readily available.
- Encouragement of informed decision-making in relation to disease-preventive strategies and ecologically sustainable farming methods.
- Reduce the dependency on chemical treatments by emphasizing early detection and proactive disease management.

This mobile application aspires to alter rice farming by providing farmers with the tools necessary to cultivate healthier crops, minimize losses, and achieve long-term agricultural sustainability.

3. Exploratory Data Analysis

3.1 Dataset Description

The datasets used for training and validating the rice leaf disease detection model were sourced from two distinct platforms, each offering unique characteristics and challenges. These datasets provide a comprehensive collection of rice leaf images, capturing varying disease types, environmental conditions, and annotations.

- a) **Kaggle Dataset:** The first dataset, sourced from Kaggle, consists of 3,678 images of rice leaves categorized into six distinct disease classes: Bacterial Leaf Blight, Brown Spot, Healthy, Leaf Blast, Leaf Scald, and Narrow Brown Spot. Initially sourced from publicly available agricultural image archives, the dataset was supplemented with additional images gathered from local agricultural sources. These supplementary images were carefully selected to capture local variations in rice leaf diseases, thus ensuring the dataset represented a diverse range of rice farming conditions. The annotations for this dataset were custom-created using Roboflow, enabling efficient labeling and consistency across class assignments.
- b) **Dataninja Dataset:** The second dataset, obtained from Dataninja, comprises 373 images of rice leaves categorized into three disease classes: Bacterial Blight, Brown Spot, and Rice Blast. Unlike the Kaggle dataset, this dataset came with pre-provided annotations, streamlining the data processing and ensuring validated class labels. However, a significant imbalance was noted in the Dataninja dataset, with the "Bacterial Blight" class representing the majority of the images. This class imbalance posed challenges during model training, requiring careful handling to avoid overfitting to the more frequent class and neglecting the less-represented classes.

Both datasets covered a variety of conditions, including varying lighting, image quality, and leaf orientations. The images ranged from healthy rice leaves to those with various levels of disease damage, which posed challenges during preprocessing. Some images have blur, low resolution, or noise, which required careful handling to ensure that the final dataset was suitable for training high-performance models.

Disease Classes:

The datasets contain images of both healthy rice leaves and five different types of rice leaf diseases, resulting in six categories to differentiate. These categories represent some of the most common and economically damaging rice diseases:

- **Bacterial Leaf Blight:** A bacterial infection that causes wilting and lesions on rice plants, leading to reduced grain yield.
- **Brown Spot:** A fungal disease causing brown lesions on leaves, reducing photosynthesis and lowering grain quality.

- **Healthy:** Images of healthy rice leaves used to distinguish between infected and non-infected leaves.
- **Leaf Blast:** A fungal infection that causes oval or elliptical lesions, significantly reducing yield.
- **Leaf Scald:** A fungal disease that creates long, moist lesions, inhibiting plant growth.
- **Narrow Brown Spot:** A fungal infection that causes dark, elongated lesions, reducing yield potential.



Fig 3.1.1: Categories of rice leaf diseases.

3.2 Data Split:

For the Kaggle dataset, the data was split into training, validation, and testing sets with a ratio of 70%, 20%, and 10%, respectively. This ensures sufficient data for training, hyperparameter tuning, and model evaluation. The training set comprises 70% of the total images, used to train the model, while the 20% validation set aids in tuning hyperparameters and monitoring performance during training. The final 10% was reserved for testing, allowing for the evaluation of the model's generalization capability.

In contrast, the Dataninja dataset had predefined splits for training, validation, and testing, ensuring appropriate data distribution across sets without requiring manual intervention.

3.3 Data Preprocessing

As part of the exploratory data analysis (EDA), several preprocessing steps were applied to the images to enhance their quality and prepare them for model training. A standard transformation

pipeline was applied to both datasets to ensure consistency and improve model performance. The transformation pipeline consisted of the following steps:

- **Resize:** All images were resized to a uniform dimension of 640x640 pixels. This resizing ensured that the input images had consistent dimensions, which is essential for efficient processing by the neural network.
- **ToTensor:** The images were converted into tensor format, the required input format for PyTorch models. This step transformed the images into numerical data that the model could process.
- **Normalize:** The images were normalized with mean and standard deviation values. This step ensured that the pixel values were appropriately scaled for optimal model performance. The mean values of [0.485, 0.456, 0.406] and standard deviation values of [0.229, 0.224, 0.225] were used a standard practice for transfer learning with pre-trained models.

4. Methodology

4.1 Introduction to Languages (Front-End and Back-End)

This project utilized a combination of programming languages to create a complete mobile application system for rice leaf disease detection.

Front-End (Flutter with Dart):

Flutter is a mobile app development framework created by Google. It uses the Dart programming language, known for its speed, efficiency, and ability to compile for Android platforms. Here we used it to build the user UI of the mobile application, allowing users to interact with the system, view detection results, and access various features such as disease information, preventive measures, and resource purchasing options.

Back-End (Python):

Python being a most versatile and widely used programming language known for its readability and extensive ecosystem of libraries became the primary language for this project by serving as the primary language for building the core functionalities, including YOLO v11, and Faster R-CNN with different backbones. Google Colab provided the necessary computational resources to develop, train, and evaluate the models efficiently.

4.2 Supporting Languages/Packages

In addition to the core front-end and back-end languages, several supporting packages and services played a crucial role in making the project functional and efficient. Here are the key tools that contributed to the project:

- **Firebase:** A Google service providing Backend-as-a-Service (BaaS), which was used for user authentication, enabling login and sign-up functionality in the application.
- **Google Cloud:** Utilized for deploying the deep learning models, ensuring efficient hosting, and managing the communication between the front-end (Flutter) and back-end systems (Python-based models).

4.2.1 Additional Supporting Packages (Python):

Several Python packages were essential for data processing, model training, evaluation, and enhancement in this project. These libraries were integral to the development pipeline, ensuring efficient and effective model training and evaluation.

a) Data Manipulation:

NumPy: A core package for numerical computing in Python, NumPy provided essential tools for manipulating and pre-processing data arrays efficiently. It was used extensively for handling large datasets and performing array-based operations during data processing.

b) Image Processing:

PIL (Python Imaging Library): Used for image loading, transformation, and manipulation during the data preparation phase. PIL provided functions for handling image

resizing, normalization, and other transformations to ensure the images were ready for model input.

c) Model Training and Evaluation:

PyTorch, a powerful deep learning framework, was used for building, training, and optimizing models like Faster R-CNN and YOLO v11. **Torchvision** supported this by providing pre-trained models, image transformations, and dataset utilities. **TQDM** was integrated to display progress bars during training, enhancing workflow efficiency. **Matplotlib** was employed for visualizing results, such as model metrics and training curves. Additionally, **scikit-learn** was used to calculate evaluation metrics like the confusion matrix and classification report, helping assess model performance and effectiveness.

d) Model Explainability:

PyTorch-Grad-CAM: A tool used to generate Class Activation Maps (CAM) for model interpretability. It helped visualize which parts of the rice leaf images were being focused on by the model during classification, enhancing the model's explainability and trustworthiness.

4.3 User Characteristics

This app aims to address the challenge of rice leaf diseases that significantly impact the yield and livelihood of farmers. The app is designed to assist farmers, agricultural extension officers, and researchers in identifying and managing rice leaf diseases effectively.

A) Target Users

The app is designed for farmers, agricultural extension officers, and agricultural researchers, particularly those working in rice cultivation. Farmers, who are the main users, may have varying levels of technological knowledge and use the app to quickly identify rice leaf diseases and make informed decisions about disease management. Agricultural extension officers, who support farmers in improving agricultural practices, need an easy-to-use tool to offer advice and enhance productivity. Additionally, agricultural researchers can use the app to study the spread of diseases and their impact on crop yield, helping them gather valuable data for their studies.

B) User Needs and Skills

1) Farmers:

Farmers need a simple, intuitive interface to easily capture rice leaf images and obtain disease diagnoses with minimal effort. The app is designed for users with limited technical skills and offers multilingual support to cater to diverse linguistic backgrounds. Offline functionality ensures usability in areas with poor internet connectivity. Farmers are expected to have basic smartphone skills and some awareness of common rice diseases, which the app further supplements with informative features.

2) Agricultural Extension Workers:

Extension workers require advanced features in the AGRI GUARDIAN app, such as detailed information on rice diseases, symptoms, and treatment options. Proficiency in using smartphones and mobile apps, along with a solid understanding of rice disease management, enables them to effectively use these tools to support farmers and enhance productivity.

C) User Experience (UX) Considerations

Accessibility is a critical consideration for the app, as many target users have limited technical expertise. To ensure ease of use, the app features larger fonts, high-contrast color schemes, and a simple, intuitive interface. Localization is addressed through functionality in areas with poor internet connectivity, making it suitable for diverse and remote communities. The app is tailored to regional needs, ensuring its accessibility to a wide range of users. To further enhance user experience, it includes in-app tutorials and guides. These provide step-by-step instructions and video demonstrations, enabling users to quickly understand and navigate the app. This thoughtful design ensures the app is effective and user-friendly for all stakeholders.

4.4 Constraints

During the development of the app we faced several challenges related to data, computational resources, user engagement, and ethical considerations. Below are the main constraints identified during the project development:

a) Data Constraints:

Data availability, quality, and class imbalance are key factors that affect model performance. Access to a diverse and comprehensive dataset is crucial for training a model that can recognize various rice leaf diseases. However, limited data can restrict its ability to generalize. Data quality also directly impacts accuracy issues like blurry images, poor lighting, or mislabeled data can lead to incorrect diagnoses. Furthermore, class imbalance, where some diseases are more common than others, can cause the model to favor identifying more frequent diseases while struggling with rarer ones.

b) Computational Constraints:

Training Time and Resources: Deep learning models, such as YOLO v11 and Faster R-CNN, require substantial computational resources and time for training. Limited access to powerful hardware, such as GPUs or TPUs, can increase training time, delaying project timelines.

c) User Constraints:

A complex user interface or lack of proper onboarding could hinder engagement, so the design should prioritize ease of use with clear navigation and instructions. Additionally, in areas with unreliable or slow internet, offline functionality is essential. Allowing access to previously downloaded disease data and reference images ensures the app

remains usable, even without a stable internet connection, improving its overall accessibility.

d) Ethical Constraints:

- **Data Privacy:** To ensure **data privacy**, the AGRI GUARDIAN app secures data storage in Firebase and obtains user consent for collecting sensitive information, such as geographic location and farming details. Transparency regarding data usage and user rights is crucial for building trust.
- **Bias and Fairness:** The model's performance can be influenced by skewed or unrepresentative training data. To mitigate this, the development process emphasizes selecting diverse datasets, applying data augmentation techniques, and evaluating the model under various conditions to ensure fairness and accuracy in disease detection.

4.5 Flow Chart

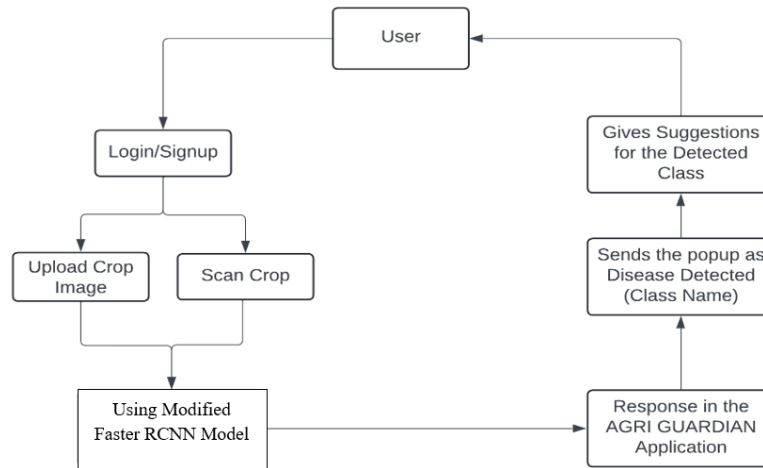


Fig 4.5.1: Modular work Flow

The Fig 4.5.1 illustrates the workflow of an AGRI GUARDIAN. It involves user login/signup, crop image upload and scanning, disease detection using a modified Faster R-CNN model, and providing suggestions and notifications to the user through the application.

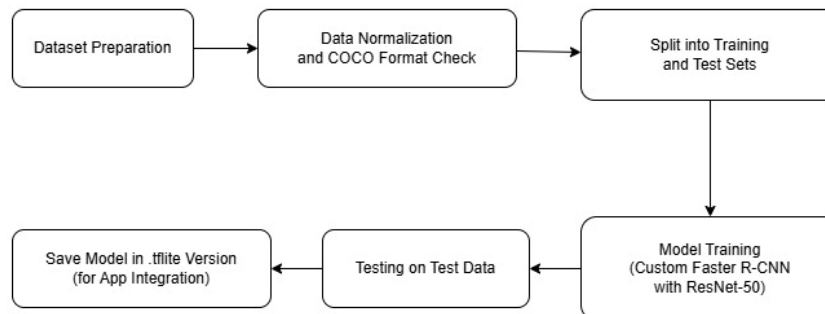


Fig 4.5.2: Faster RCNN Flow Process

This Fig 4.5.2 outlines the steps involved in training a Faster R-CNN model for rice disease detection. It includes dataset preparation, normalization, COCO format check, splitting into training and test sets, model training, testing on the test set, and saving the model in tflite format for app integration.

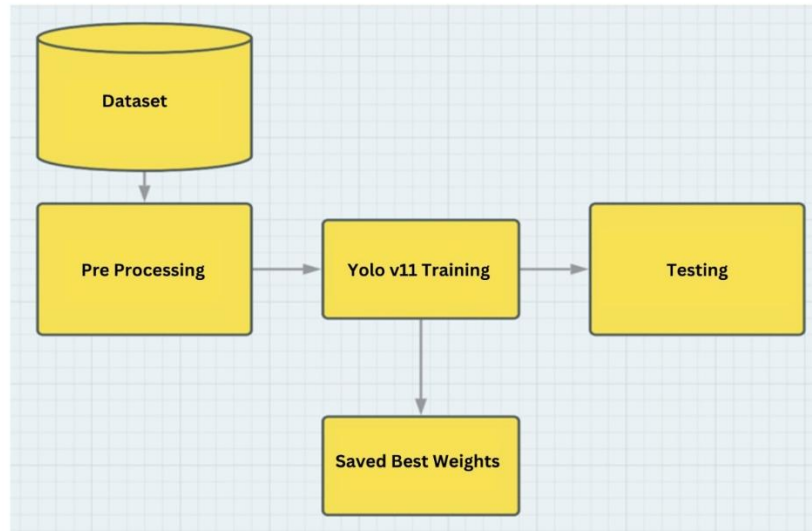


Fig 4.5.3: Flowchart for the YOLOv11 model architecture

The Fig 4.5.3 explains the workflow of training a YOLOv11 model for object detection. It involves dataset preparation, pre-processing, model training, testing, and saving the best weights for future use.

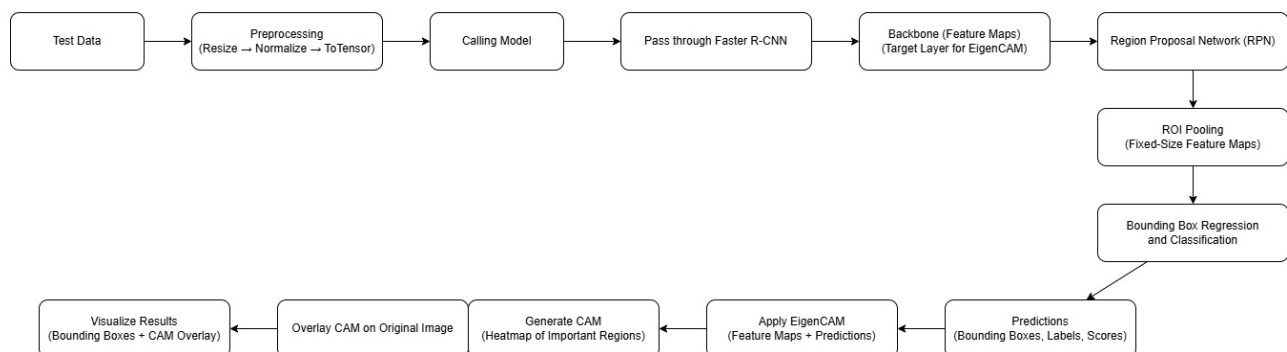


Fig 4.5.4: Flowchart for the EigenCAM Implementation.

The Fig 4.5.4 illustrates the process of using EigenCAM to visualize the decision-making process of a Faster R-CNN object detection model. It involves preprocessing test data, passing it through the model, generating CAMs using EigenCAM, and overlaying these CAMs on the original image to highlight the regions that were most influential in the model's predictions.

4.6 Database Design

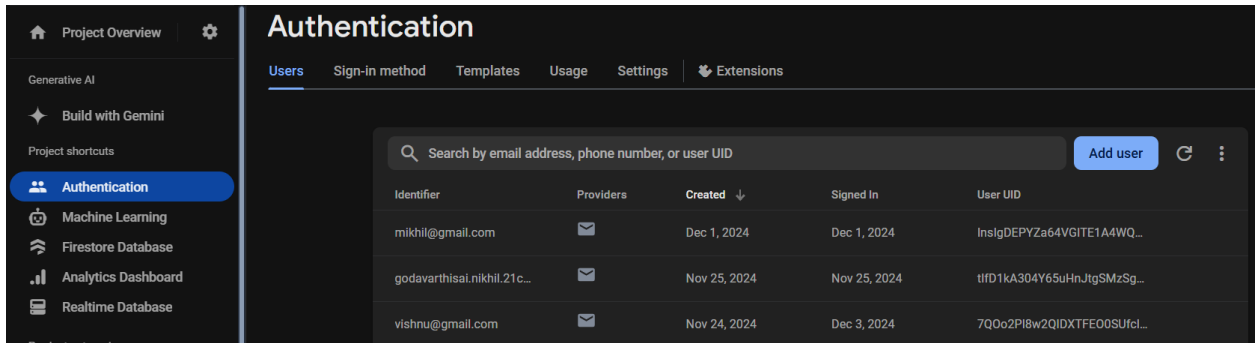


Fig 4.6: Firebase Database for Agri Guardian

In the above Figure 6, illustrates users information storage structure in Firebase for the Agri Guardian application, which enables login and signup functionalities.

These are the functionalities we used to store user's information:

Users Collection: This collection stores a separate document for each user who signups in the app.

Unique User ID (uid): It helps to identify the user and retrieves the data.

Timestamps: Records of creation and update times.

4.7 Algorithm Discussion:

4.7.1 YOLOv11

YOLOv11 is an advanced object detection framework designed for high-speed and accurate detection, especially of small and complex objects. It incorporates architectural advancements that improve performance over previous versions.

1) **Backbone:** The backbone extracts hierarchical features from the input image through convolutional layers and C3k2 modules, progressively increasing feature depth and abstraction.

- **Blocks:** YOLOv11's backbone consists of convolutional layers and C3k2 blocks. The first two convolution layers apply 3x3 convolutions with strides of 2 to down sample the image, capturing low-level features such as edges and textures. The C3k2 blocks, located at various stages, use residual connections and bottlenecks to improve feature flow and prevent gradient issues, enabling the model to learn complex patterns necessary for detecting subtle rice leaf disease symptoms.
- **Key Modules:** The C3k2 module, with residual connections and bottleneck layers, enhances feature extraction and learning efficiency. This design helps the model capture both high-level and fine-grained details for rice leaf disease detection. The backbone's convolutional layers and C3k2 blocks enable the model to identify

complex disease patterns at various scales, aiding in the accurate localization and classification of diseases like bacterial blight and brown spots.

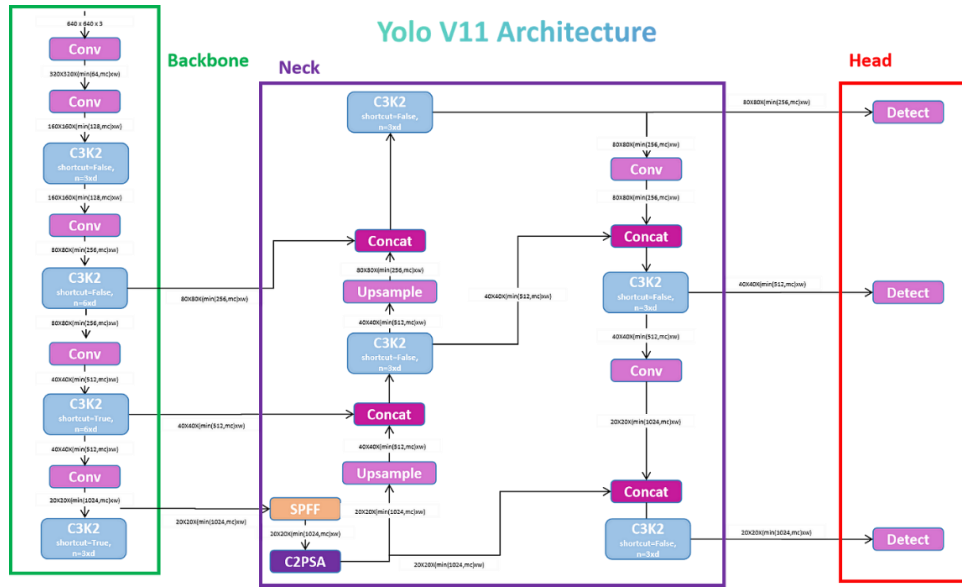


Fig 4.7.1: YOLOv11 Model architecture [15].

- 2) **Neck:** The neck module connects the backbone and the detection head, aggregating multi-scale features to enhance the model's detection capabilities. It aggregates multi-scale features and applies attention mechanisms to improve detection of objects at various sizes and highlight relevant features.
 - **Blocks:** The neck includes several important blocks for effective detection. The SPPF in Layer 9 pools features with multiple kernel sizes, capturing multi-scale representations crucial for detecting both small and large lesions. The C2PSA block in Layer 10 applies attention mechanisms to prioritize relevant features in both spatial and channel dimensions. Upsample Layers increase spatial resolution, preserving fine details, while Concat Layers combine low-level and high-level information for more accurate predictions.
 - **Key Modules:** The SPPF and C2PSA modules significantly improve YOLOv11's ability to detect rice leaf diseases. SPPF captures features at multiple scales, aiding in the detection of varying disease sizes, while C2PSA enhances the model's focus on critical features, improving accuracy, especially for subtle disease patterns. These modules enhance the model's robustness and accuracy, allowing it to localize rice leaf diseases more effectively.
- 3) **Head:** The detection head in YOLOv11 generates final predictions by combining processed features to output bounding boxes, class probabilities, and confidence scores for each grid cell. This enables precise localization and classification of rice leaf diseases.

- **Blocks:** The detection head in YOLOv11, located in Layer 23, processes feature maps from multiple scales, allowing the model to detect objects of different sizes. This multi-scale approach ensures the detection of both small and large disease symptoms on rice leaves. It outputs bounding boxes, class probabilities, and confidence scores for each predicted object, enabling accurate localization and identification of diseases.
- **Key Modules:** The Detect Module applies final convolutions to the feature maps, producing bounding boxes, class labels, and confidence scores. This facilitates accurate detection and classification of rice leaf diseases, ensuring effective diagnosis and disease management.

4.7.2 Faster RCNN

Faster R-CNN is a two-stage object detection framework that excels in accuracy and robustness. It consists of the following key components:

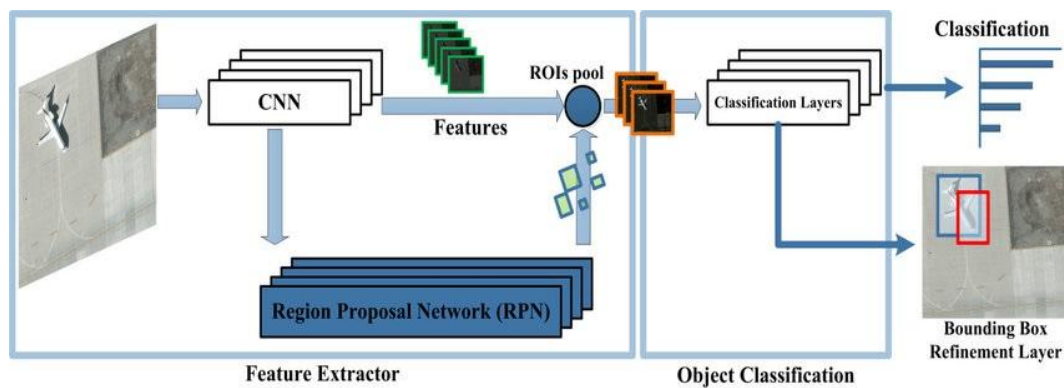


Fig 4.7.2: Faster RCNN Model architecture [16].

- 1) **Backbone:** The backbone is responsible for extracting features from the input image. Convolutional layers capture hierarchical features, starting from low-level edges and textures to high-level semantic patterns. The backbone ensures efficient training and helps the model focus on small, disease-specific patterns, such as lesions or discoloration, by sharing extracted features across multiple regions.
 - **Key Modules:** The backbone aids in feature extraction, the RPN generates precise region proposals, ROI Align improves localization accuracy, and the detection head outputs bounding boxes and class scores. Together, these modules enable the detection of subtle disease symptoms with high accuracy and robustness.
- 2) **Region Proposal Network:** The RPN is a key innovation in Faster R-CNN. It generates region proposals, i.e., bounding box candidates likely to contain objects.
 - **Blocks:** The RPN comprises several critical components designed for generating candidate bounding boxes. It employs anchor boxes of predefined sizes and aspect

ratios to detect objects of varying scales, essential for identifying both small bacterial spots and large blighted areas in rice leaves. A 3x3 convolutional layer serves as a sliding window over the feature map, extracting intermediate features from the input image. Two sibling layers followed by a classification layer that predicts the presence of an object within a region and a regression layer that refines the bounding box coordinates for precise localization.

- **Key Modules:** The RPN efficiently focuses on regions likely containing disease symptoms, reducing irrelevant background noise. By using shared feature maps from the backbone, it optimizes computation. The diverse scales and aspect ratios of the anchor boxes allow the RPN to capture different disease manifestations, improving accuracy and reducing false positives in rice leaf disease detection.

3) **ROI Pooling/ROI Align:** It extracts fixed-size feature maps from the varying-sized region proposals.

- **Blocks:** ROI Pooling and ROI Align process region proposals from the RPN, converting them into fixed-size feature maps. ROI Pooling divides each proposal into a grid and applies max pooling, while ROI Align uses bilinear interpolation to prevent misalignment, ensuring better feature alignment with the proposal boundaries.
- **Key Modules:** These modules standardize feature maps from varying-sized proposals, making them compatible with the detection head. ROI Align improves precision by eliminating misalignment, which is crucial for detecting irregularly shaped disease features. This results in more accurate localization and better class predictions, enhancing disease detection on rice leaves.

4) **Detection Head:** It classifies objects and refines their bounding box coordinates based on the ROI-pooled features.

- **Blocks:** The detection head is the final component of Faster R-CNN, where the processed features from the RPN are classified and refined. It consists of fully connected layers that convert ROI pooled features into a compact representation. The head has two key parts: class scores, which predict the disease class (e.g., Bacterial Blight), and bounding box regression, which fine-tunes the coordinates to better fit the detected object.
- **Key Features:** The detection head plays a crucial role in both classifying rice leaf diseases and refining bounding boxes for accurate localization. The class scores allow for multi-class identification, while bounding box regression ensures precise localization of disease spots. By performing both tasks simultaneously, the detection head provides high-confidence predictions and enables the detection of multiple diseases in a single image with improved accuracy and precision.

Faster R-CNN was chosen for its accuracy and robustness in detecting subtle rice leaf disease symptoms. Its two-stage approach ensures precise localization and classification, while the end-to-end training optimizes both object detection and bounding box refinement. Pretrained backbones enhance performance, enabling the model to handle diverse conditions and detect complex disease-specific features effectively.

In this object detection task, the model uses a combination of classification loss (Cross-Entropy Loss) and bounding box regression loss (Smooth L1 Loss) to optimize performance. The classification loss measures how accurately the model classifies objects, while the bounding box loss penalizes deviations between predicted and ground truth coordinates, being less sensitive to outliers than L2 loss. These losses are summed and minimized during training.

The model is optimized with Stochastic Gradient Descent (SGD), using a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005 to efficiently update parameters. A StepLR scheduler reduces the learning rate every 3 epochs by a factor of 0.1, further improving the optimization process.

4.7.3 Mobilenet v3 Large as Backbone for Faster R-CNN

It is a lightweight and efficient convolutional neural network designed for resource-constrained environments, such as mobile devices. When used as the backbone for Faster R-CNN, it effectively extracts features from input images while ensuring low computational overhead. MobileNetV3-Large integrates advanced techniques like depthwise separable convolutions, squeeze-and-excitation (SE) blocks, and the swish activation function to balance performance with computational efficiency, making it ideal for real-time object detection tasks.

- **Structure of MobileNetV3-Large:** MobileNetV3-Large consists of several key components that enhance its efficiency. Depthwise separable convolutions reduce computation by applying convolutions independently to each channel, followed by pointwise convolutions to fuse features. SE blocks are used within bottleneck layers to perform channel-wise attention, focusing on informative channels. Additionally, hard-swish activation is used to introduce non-linearity while being computationally efficient, enabling faster processing. The output feature maps are multi-resolution and shallow, making them suitable for low-resource devices.

MobileNetV3-Large is chosen for Faster R-CNN due to its efficiency, providing competitive accuracy with lower computational costs. It strikes a balance between speed and performance, making it ideal for real-time applications on resource-constrained devices. Its modular design allows scalability, adapting to various tasks without sacrificing performance. While it may not excel in complex, high-resolution detection, it is well-suited for lightweight, near real-time tasks, such as rice leaf disease detection. The model's lightweight nature, combined with SE

blocks and depthwise convolutions, enables it to focus on small-scale features like leaf spots, while its fast inference ensures timely diagnosis in agricultural contexts.

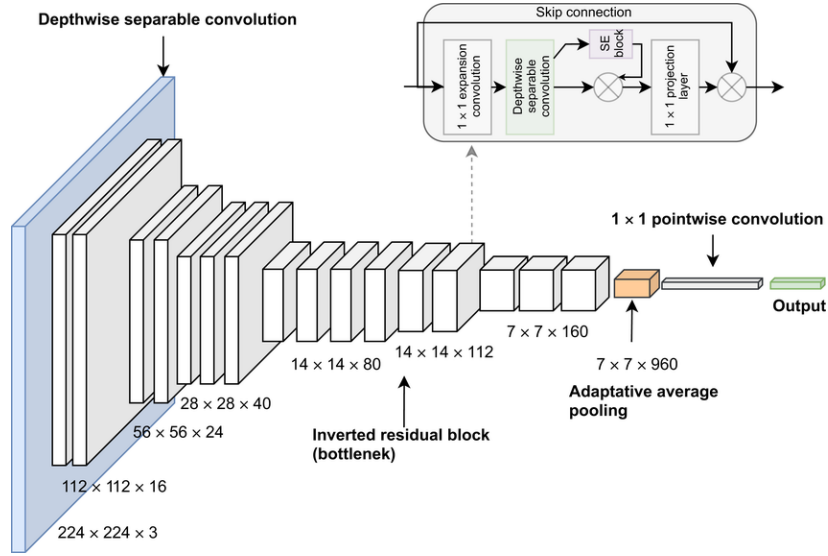


Fig 4.7.3: Mobilenet v3 Large Model architecture [17].

4.7.4 Densenet121 as Backbone for Faster R-CNN

DenseNet121 is a deep neural network that improves feature propagation by connecting each layer to every other layer. This architecture enhances feature reuse and is ideal for tasks like object detection, where hierarchical feature extraction is critical. In Faster R-CNN, DenseNet121 acts as the backbone to extract features for the RPN and detection head.

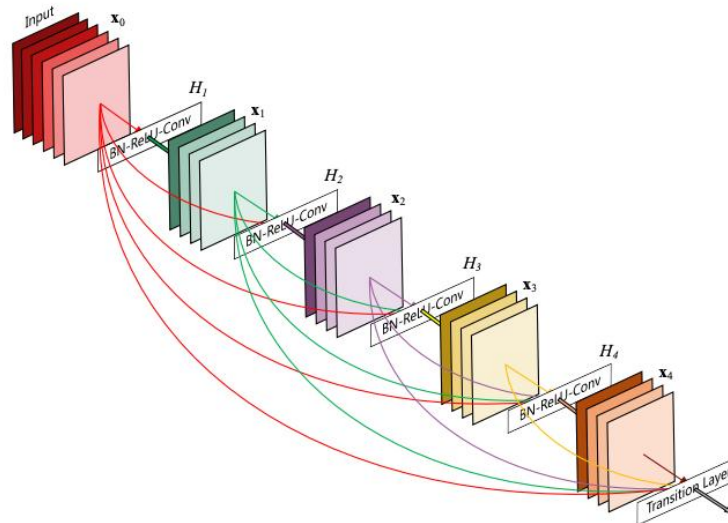


Fig 4.7.4: DenseNet121 Model architecture [18].

- **Structure of DenseNet121:** DenseNet121 includes four dense blocks, each with multiple convolutional layers. Each layer receives inputs from all previous layers, promoting feature

reuse. It utilizes 1x1 and 3x3 convolutions for feature extraction and down sampling through transition layers. The network ends with global average pooling and a classification layer, which are omitted in Faster R-CNN for feature extraction.

DenseNet121 is ideal for Faster R-CNN due to its efficient feature reuse, enabling both low- and high-level feature extraction, which is crucial for detecting subtle objects like rice leaf diseases. Its memory efficiency and ability to mitigate the vanishing gradient problem lead to faster training convergence, enhancing object detection performance. In tasks like rice leaf disease detection, DenseNet121 excels at capturing fine details such as lesions or spots, ensuring robust performance even in complex backgrounds or overlapping objects. This balance of accuracy and computational efficiency makes it well-suited for medium-sized datasets.

4.7.5 ResNet-50 as Backbone for Faster R-CNN

ResNet-50 is a deep convolutional neural network designed for image classification and feature extraction. It excels in training deep networks by overcoming vanishing gradient issues, making it ideal for object detection tasks like rice leaf disease detection. As a backbone for Faster R-CNN, ResNet-50 extracts hierarchical features from images, enabling the model to detect and classify objects effectively across various scales.

- **Structure of ResNet-50:** It consists of five stages, each containing convolutional layers and Residual Blocks. The network starts with a 7x7 convolution and max pooling, followed by residual blocks containing three convolutions: 1x1 for dimensionality reduction, 3x3 for feature extraction, and 1x1 to restore dimensions. Shortcut connections between layers preserve important features and allow for efficient training of deep networks.

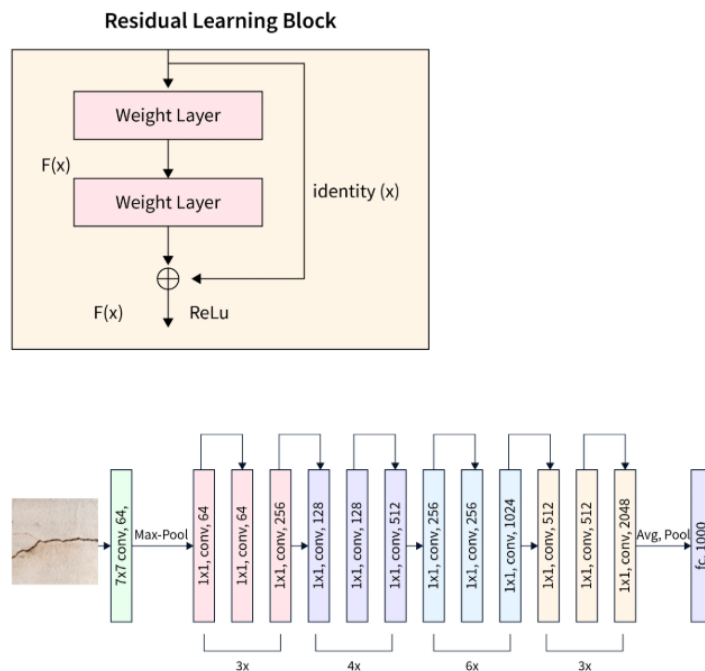


Fig 4.7.5: ResNet 50 Model architecture [19].

ResNet-50 is an ideal backbone for Faster R-CNN due to its deep feature representation and efficient training through residual connections. It captures intricate patterns, making it well-suited for detecting subtle rice leaf disease symptoms. Pretrained on ImageNet, ResNet-50 speeds up training while balancing accuracy and computational efficiency. Its ability to extract multi-scale features allows Faster R-CNN to detect both small and large symptoms, such as bacterial blight and sheath blight, ensuring precise localization and classification for effective rice leaf disease detection.

4.7.6 Eigen-CAM for Explainability in Faster R-CNN with ResNet-50

- Eigen-CAM was integrated into the modified Faster R-CNN model with a ResNet-50 backbone to enhance the explainability of the disease detection results. The purpose of this technique is to provide insights into the regions of an image that contribute most to the model's decision-making process.
- The process begins with the input image being preprocessed and passed through the Faster R-CNN model to generate predictions, which include bounding boxes, labels, and confidence scores for detected objects. The Eigen-CAM method is then applied to visualize the influence of different parts of the image on the model's output. Specifically, it targets the backbone of the model, and the bounding boxes are used to focus on the areas of interest for each class of rice leaf disease.
- Eigen-CAM generates a heatmap indicating the regions of the image that the model considers most relevant for its predictions. This heatmap is then overlaid on the original image to provide a visual explanation of how the model arrived at its conclusions. The regions highlighted by the heatmap align with the areas in the image containing the detected rice leaf diseases, making it easier for users to understand which features led to the classification.
- By incorporating Eigen-CAM, the modified Faster R-CNN model not only provides predictions but also offers interpretability, allowing users to visually validate and trust the disease detection results. This feature is particularly useful in applications where explainability is crucial for decision-making, such as agricultural disease management.

5. Experimental Results

5.1 Performance Comparison of YOLO v11 and Faster R-CNN

Initially, experiments were conducted on both the Kaggle and Dataninja datasets using YOLO v11 due to its updated architecture and capability for object detection. However, the results indicated comparatively low performance, as shown in Table 5.1.

Table 5.1.1: Performance Metrics for YOLO v11

| Dataset | mAP | Precision | Recall |
|-----------|-------|-----------|--------|
| Kaggle | 0.443 | 0.5 | 0.45 |
| Dataninja | 0.342 | 0.44 | 0.34 |

The results demonstrated suboptimal performance, particularly in precision and recall, highlighting the need for a more effective model to address the rice leaf disease detection task.

To improve detection performance, Faster R-CNN with a MobileNet_v3_Large backbone was employed. Known for its robust feature extraction capabilities and efficiency in object detection, this model significantly outperformed YOLO v11 across all evaluation metrics. The performance metrics for Faster R-CNN are summarized in Table 5.2.

Table 5.1.2: Performance Metrics for Faster RCNN

| Dataset | mAP | Precision | Recall |
|-----------|-------|-----------|--------|
| Kaggle | 0.821 | 0.928 | 0.53 |
| Dataninja | 0.723 | 0.79 | 0.76 |

Both datasets demonstrated a substantial improvement in mAP, precision, and recall compared to YOLO v11. The Kaggle dataset achieved a mAP of 0.821 with a precision of 0.928 and recall of 0.530, while the Dataninja dataset achieved an mAP of 0.723 with a precision of 0.790 and recall of 0.760.

These results confirmed that Faster R-CNN with MobileNet_v3_Large is significantly more effective than YOLO v11, particularly in achieving higher detection accuracy and precision. As a result, Faster R-CNN was selected for further experimentation and optimization, providing a

more reliable and accurate framework for the rice leaf disease detection system. This choice aligns with the objective of developing a robust and efficient solution for agricultural applications.

5.2 Optimization of Faster R-CNN Using Different Backbones

To further enhance the performance of Faster R-CNN for the rice leaf disease detection task, experiments were conducted using various backbone architectures. The backbone is a critical component of the model, responsible for feature extraction, and has a significant impact on detection accuracy and overall performance. Three different backbones were evaluated: MobileNet_v3_Large, DenseNet121, and ResNet-50. The performance metrics for both the Kaggle and Dataninja datasets are presented in Tables 5.3 and 5.4, respectively.

Table 5.2.1: Performance Metrics for Different Backbones on Kaggle Dataset

| Backbone | mAP | Precision | Recall |
|--------------------|-------|-----------|--------|
| MobileNet v3 Large | 0.821 | 0.928 | 0.53 |
| DenseNet 121 | 0.78 | 0.75 | 0.64 |
| Resnet 50 | 0.87 | 0.86 | 0.95 |

Table 5.2.2: Performance Metrics for Different Backbones on Dataninja Dataset

| Backbone | mAP | Precision | Recall |
|--------------------|-------|-----------|--------|
| MobileNet v3 Large | 0.723 | 0.79 | 0.76 |
| DenseNet 121 | 0.66 | 0.73 | 0.56 |
| Resnet 50 | 0.75 | 0.82 | 0.84 |

Performance Analysis by Backbone:

- 1. MobileNet_v3_Large:** This backbone was initially used due to its lightweight and efficient architecture. On the Kaggle dataset, it achieved a mAP of 0.821, precision of 0.928, and recall of 0.530. For the Dataninja dataset, the mAP was 0.723, with precision of 0.790 and recall of 0.760. Despite strong precision, moderate recall suggested the need for better instance detection.

2. **DenseNet121:** DenseNet121, with its enhanced gradient flow and feature reuse, achieved a mAP of 0.780, precision of 0.750, and recall of 0.640 on the Kaggle dataset. On the Dataninja dataset, it showed lower performance, with a mAP of 0.660, precision of 0.730, and recall of 0.560. Its limited adaptability across datasets indicated it was less effective for this task.
3. **ResNet-50:** ResNet-50 outperformed the other backbones, achieving the highest mAP (0.870), precision (0.860), and recall (0.950) on the Kaggle dataset. On the Dataninja dataset, it achieved a mAP of 0.750, precision of 0.820, and recall of 0.840. Its robust feature extraction capabilities delivered balanced and superior detection performance, making it the most suitable choice.

Comparing the results of the three backbones, Faster R-CNN with a ResNet-50 backbone showed superior performance on both datasets. Its high mAP, precision, and recall highlighted its effectiveness in detecting and classifying rice leaf diseases, making it the chosen backbone for model deployment in the AGRI GUARDIAN application.

In object detection, the backbone is critical for extracting meaningful features, directly influencing performance. ResNet-50, DenseNet121, and MobileNet_v3_Large offer unique strengths, with ResNet-50 excelling in capturing complex patterns through residual connections, making it ideal for the rice leaf disease detection task. Changing the backbone yields more substantial improvements than hyperparameter tuning by enhancing feature extraction, improving generalization, and adapting to dataset complexity. For this task, ResNet-50 outperformed other backbones and YOLO v11, achieving higher mAP, precision, and recall, demonstrating that backbone optimization is a more effective approach for boosting model performance.

5.3 Optimizing Faster R-CNN with ResNet-50

In rice leaf disease detection, transfer learning enables leveraging pre-trained models, like Faster R-CNN with ResNet-50, to avoid training from scratch. By fine-tuning a pre-trained model, you can save time and resources, using knowledge learned from large datasets to improve performance on your task. Transfer learning allows the model to quickly adapt to detecting rice leaf diseases by retaining useful general features like textures and object shapes, making the training process more efficient and effective.

- **Freezing Layers:** Freezing specific layers during transfer learning focuses training on task-specific features while preserving general features from the pre-trained model. Freezing early layers (conv1, layer1 in ResNet-50) allows the model to retain essential information like edges and simple shapes, which are relevant across various tasks. This reduces training time, prevents overfitting on smaller datasets, and conserves computational resources by limiting the number of parameters that need updating.
- **Feature Pyramid Network (FPN) Adjustment:** The FPN is essential for detecting objects at multiple scales. By freezing the first few layers of the FPN, the model can reduce

computational complexity while still preserving the ability to detect rice leaf diseases of varying sizes. The initial layers of the FPN capture coarse features that may not be as beneficial for detecting rice leaves, while deeper layers that capture finer details are left unfrozen to maintain performance.

- **Region Proposal Network (RPN) Adjustment:** The RPN generates candidate bounding boxes for object detection, and adjusting its parameters, such as the number of anchors and Non-Maximum Suppression (NMS) threshold, is crucial for efficiency and accuracy. Reducing the number of anchors (6) helps the model focus on relevant object shapes and sizes, while adjusting the NMS threshold (0.6) balances detection accuracy and redundant proposals. Fine-tuning these parameters optimizes the model for rice leaf disease detection.

After determining that Faster R-CNN with a ResNet-50 backbone outperforms other architectures, we further optimized the model using transfer learning techniques. This approach leverages pre-trained models to save time, computational resources, and enhance model performance on the rice leaf disease detection task. By applying specific modifications, including freezing layers, adjusting the Feature Pyramid Network (FPN), and fine-tuning the Region Proposal Network (RPN), significant improvements in both performance and training efficiency were observed.

Time Efficiency Improvement: The modifications made to the Faster R-CNN with ResNet-50 backbone resulted in a notable reduction in training time. Initially, each epoch took a certain time of 9 minutes to complete. However, after the transfer learning adjustments, the time per epoch was reduced by 15 seconds to 8 minutes 45 seconds. With a total of 10 epochs, this resulted in a time savings of 150 seconds (2.5 minutes) per training session.

Table 5.3.1: Performance Metrics for ResNet-50 and Modified Model on Kaggle Dataset

| Model | mAP | Precision | Recall |
|--------------------|--------|-----------|--------|
| ResNet 50 | 0.87 | 0.86 | 0.95 |
| Modified ResNet 50 | 0.8835 | 0.877 | 0.935 |

Table 5.3.2: Performance Metrics for ResNet-50 and Modified Model on dataninja Dataset

| Dataset | mAP | Precision | Recall |
|--------------------|-------|-----------|--------|
| ResNet 50 | 0.75 | 0.82 | 0.84 |
| Modified ResNet 50 | 0.772 | 0.9 | 0.68 |

Performance Analysis by Backbone:

- 1) **Kaggle Dataset:** The modified model showed a slight improvement in mAP (0.8835) over the original ResNet-50 (0.870). Precision increased to 0.877, and recall rose to 0.935. These

improvements highlight the effectiveness of transfer learning modifications, including freezing layers and adjusting the FPN and RPN, in enhancing detection accuracy.

- 2) **Dataninja Dataset:** On the Dataninja dataset, the modified model achieved an mAP of 0.7724, up from 0.750. Precision saw a significant increase to 0.900, while recall dropped slightly to 0.680. The improved precision indicates a more accurate detection of rice leaf diseases, focusing on higher-quality proposals.

These results show that transfer learning, along with network adjustments, led to a more efficient and accurate Faster R-CNN model, improving both detection performance and training time for agricultural disease detection.

The trained Faster R-CNN model with a ResNet-50 backbone was saved as a .pt file and integrated into the **AGRI GUARDIAN** app via API calls. The app processes user-uploaded images, running them through the model to return disease predictions and bounding boxes. After successful integration and testing, the app was deployed on the **Indus Appstore**, ensuring accessibility for agricultural users.

5.4 Integration of Eigen-CAM for Explainability

To enhance the interpretability of the modified Faster R-CNN with ResNet-50 backbone, we integrated Eigen-CAM into the framework. Eigen-CAM is a gradient-based visualization technique that highlights the most influential regions in an image for a model's predictions. This step was essential for understanding the decision-making process of the model and ensuring its reliability for agricultural applications, particularly in critical tasks like rice leaf disease detection.

The Eigen-CAM integration provides a visual representation of the areas the model focuses on when detecting diseases, aiding in verifying the correctness of the predictions. By overlaying the Eigen-CAM heatmaps on the input images, we could assess whether the model was concentrating on the diseased areas of the leaves or being misled by irrelevant features.

Testing Results with Eigen-CAM:

The modified Faster R-CNN model was tested on both the Kaggle and Dataninja datasets, with Eigen-CAM visualizations generated for various predictions. These visualizations, displayed in Figures 5.5.2 and 5.5.3, showcase the regions of interest for disease detection. The results illustrate that the model consistently focused on diseased leaf areas, confirming the effectiveness of the modifications and the added explainability provided by Eigen-CAM.

This integration of Eigen-CAM not only validated the accuracy of the model's predictions but also enhanced the trustworthiness and usability of the system for agricultural experts and farmers. By providing clear visual feedback on its predictions, the system becomes more transparent and actionable in real-world scenarios.

5.5 Qualitative Analysis

This section provides a visual comparison of the models' performance, the predictions made by the modified Faster R-CNN, and the AGRI GUARDIAN application's interface. The visuals highlight the improvements in detection accuracy and model explainability.

This histogram (Fig 5.5.1) compares the mAP scores of YOLO v11, Faster R-CNN with different backbones, and the modified Faster R-CNN for both the Kaggle and Dataninja datasets. The visual illustrates the progression of model performance, showcasing the substantial improvement achieved by the modified Faster R-CNN.

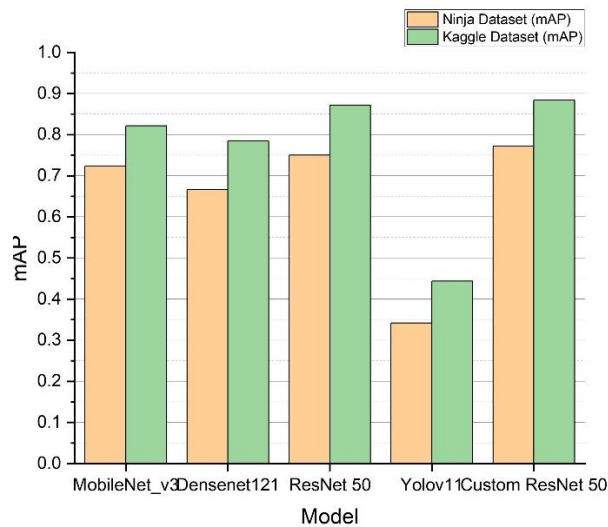
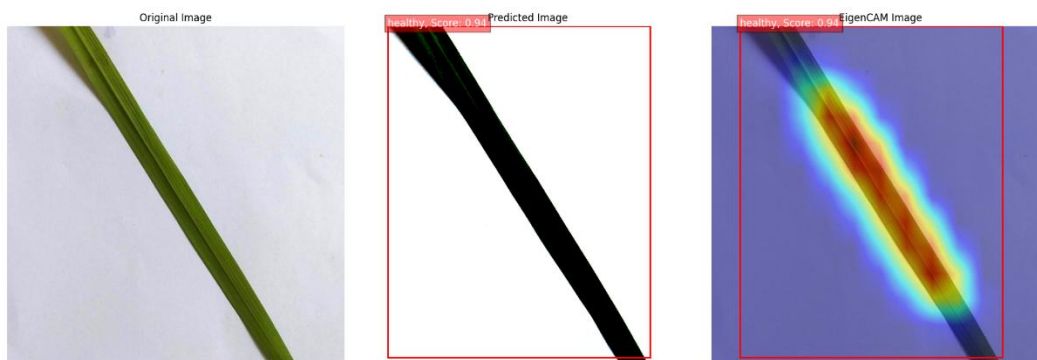


Figure 5.5.1: Comparison of mAP for Different Models Across Datasets

The Fig 5.5.2 displays side-by-side visualizations of the original image, the predicted bounding boxes from the modified Faster R-CNN, and the corresponding Eigen-CAM output for a sample image from the Kaggle dataset. The Eigen-CAM highlights areas critical to the model's predictions, providing insights into its decision-making process.



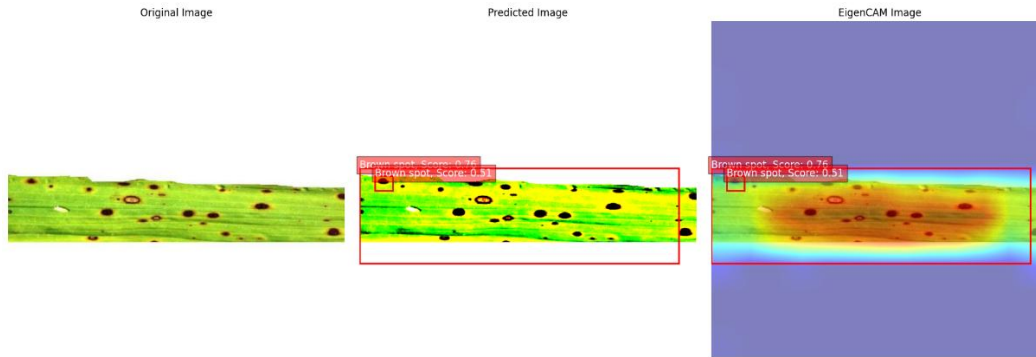


Figure 5.5.2: Results of Modified Faster R-CNN on Kaggle Dataset

This figure illustrates detection results on the Dataninja dataset, presenting the original image, predictions, and Eigen-CAM side by side. The visualization highlights the model's precision in detecting and localizing diseases.

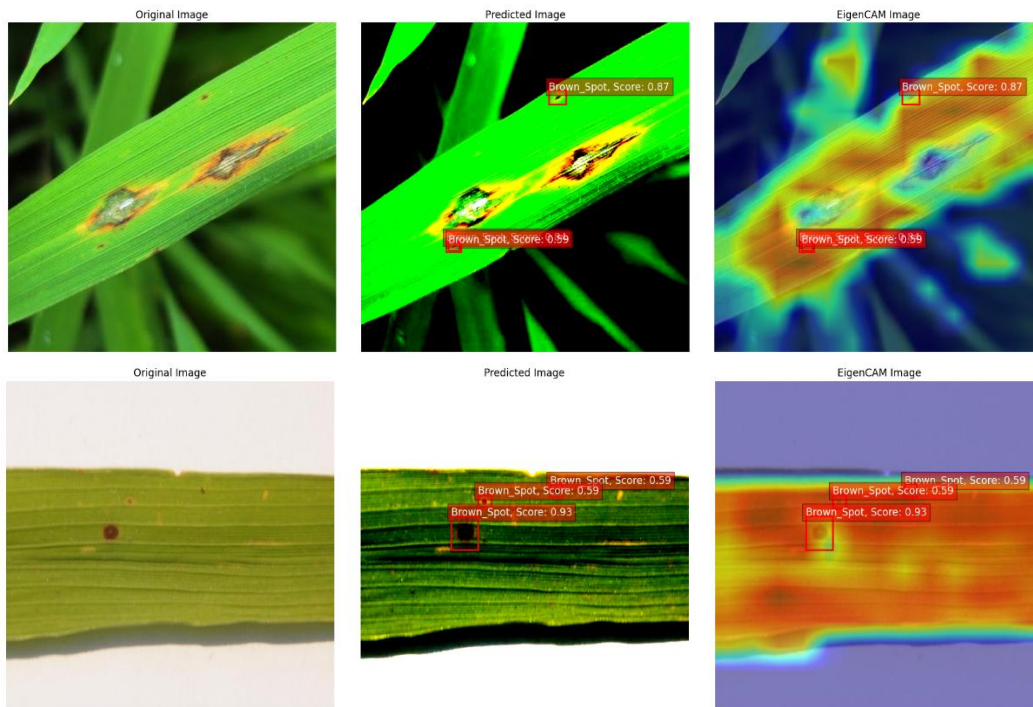


Figure 5.5.3: Detection Results of Modified Faster R-CNN on Dataninja Dataset

The screenshot Fig 5.5.4 showcases the AGRI GUARDIAN application's user-friendly home screen, featuring disease detection tools, prevention guidelines, and resources for purchasing agricultural products.

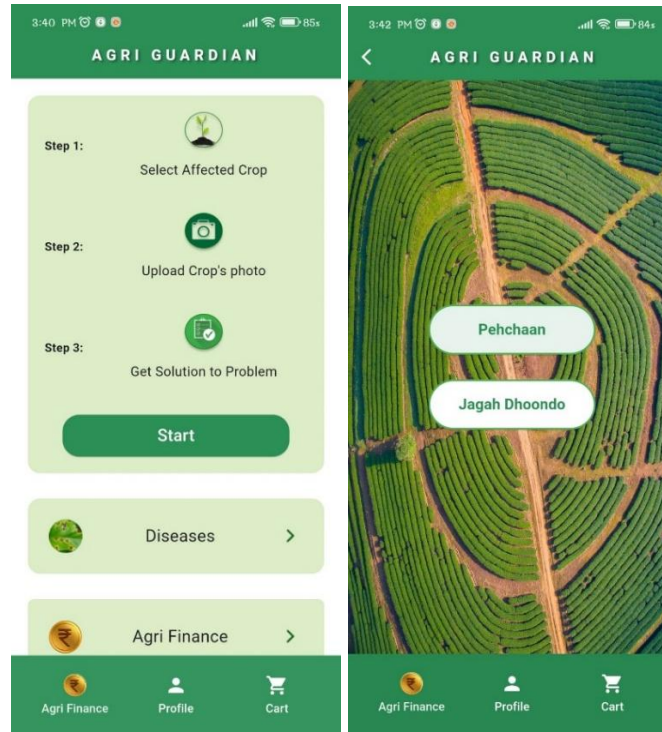


Fig 5.5.4: AGRI GUARDIAN Application Interface – Home Screen

The screenshot Fig 5.5.5 illustrates the disease detection results interface, where users can view detected diseases along with detailed prevention measures.

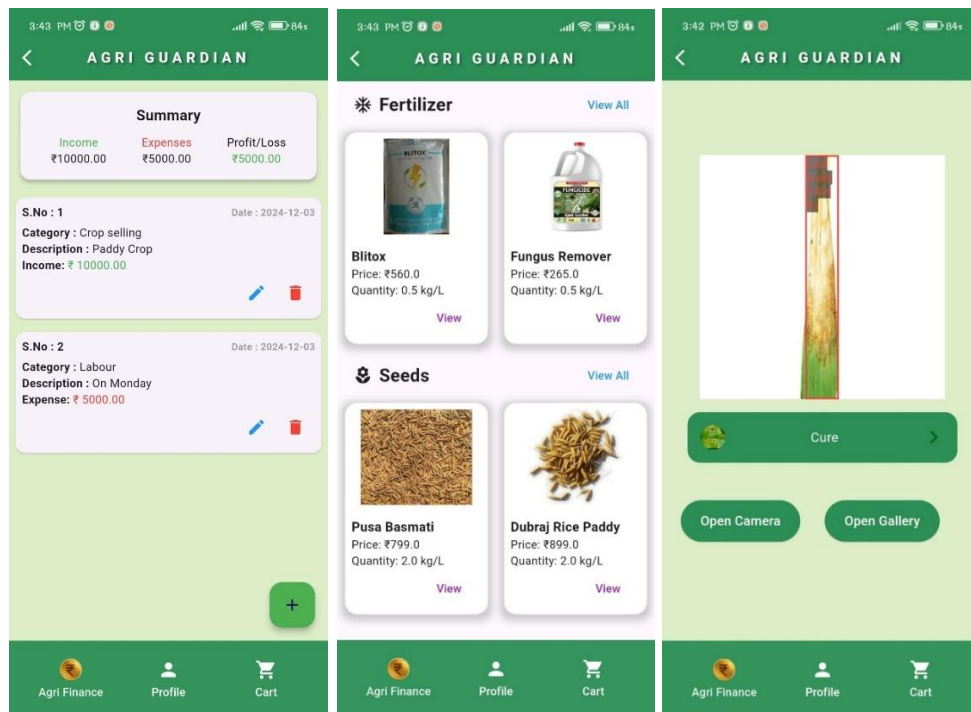


Fig 5.5.5: AGRI GUARDIAN Application Interface – Disease Detection Results

6. Conclusion and Future Scope

6.1 Conclusion

The AGRI GUARDIAN mobile application has effectively tackled the challenge of rice leaf disease detection, providing farmers with a comprehensive solution for identifying diseases, accessing preventive measures, and managing agricultural finances. The combination of Faster R-CNN with a ResNet-50 backbone and transfer learning has resulted in a high-performing model capable of accurate disease detection, with a mAP of 88.35. Additionally, the use of Eigen-CAM has enhanced the interpretability of model predictions, offering transparency in disease identification.

The application goes beyond disease detection by incorporating features like a Agri Finance system and a store option for purchasing insecticides and pesticides. These features equip farmers with the necessary tools to make informed decisions on disease management and resource procurement. With a user-friendly interface and advanced machine learning capabilities, the application provides significant value for rice farmers and agriculture officers alike, promoting early disease detection, informed treatment strategies, and improved crop health and yield.

6.2 Future Scope

1. Developed a custom model combining parts from various architectures, but faced issues with the loss functions. These will be addressed to enhance performance and stability.
2. Incorporating real-time data like humidity, soil health, and temperature will improve disease prediction accuracy, while adding MSP insights can aid farmers in financial decisions.
3. Adding multilingual support will make the app accessible to farmers globally, breaking language barriers.
4. Adding multilingual support will make the application accessible to farmers globally.
5. A community feature will allow farmers to share disease reports and treatment results, fostering knowledge exchange.

References:

- [1]. Q. Cao, D. Zhao, J. Li, *et al.*, "Pyramid-YOLOv8: a detection algorithm for precise detection of rice leaf blast," *Plant Methods*, vol. 20, p. 149, 2024. doi: 10.1186/s13007-024-01275-3.
- [2]. Zenino, E., M. Maretanio, M. Zen, S. Hadi, P. Kristalina, and A. Pratiarso, "Estimation of rice field area using YOLO method to support smart agriculture system," *Proc. 2024 International Electronics Symposium (IES)*, pp. 562–568, 2024. doi: 10.1109/IES63037.2024.10665762.
- [3]. Q. Cao, D. Zhao, J. Li, J. Li, G. Li, S. Feng, and T. Xu, "Pyramid-YOLOv8: A detection algorithm for precise detection of rice leaf blast," *Plant Methods*, vol. 20, no. 1, 2024. doi: 10.1186/s13007-024-01275-3.
- [4]. R. Sapkota and M. Karkee, "Comprehensive Performance Evaluation of YOLO11, YOLOv10, YOLOv9, and YOLOv8 on Detecting and Counting Fruitlet in Complex Orchard Environments," 2024. doi: 10.32388/e9y7xi.
- [5]. "Rice Leaf Disease Classification and Detection Using YOLOv5," 2022. doi: 10.48550/arxiv.2209.01579.
- [6]. A. K. and A. S. Singh, "Detection of Paddy Crops Diseases and Early Diagnosis Using Faster Regional Convolutional Neural Networks," 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2021, pp. 898-902, doi: 10.1109/ICACITE51222.2021.9404759.
- [7]. D. Dalmar, A. Dakari, and C. Daniel, "A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs. SSD vs. Faster R-CNN," *Information Technology Engineering Journals (ITEJ)*, vol. 8, no. 2, pp. 96-106, 2023, doi: 10.24235/itej.v8i2.123.
- [8]. R. Saieshan, N. Pillay, and N. Singh, "Comparative Study of Convolutional Neural Network Object Detection Algorithms for Image Processing," 2023 IEEE International Conference on Electronics, Computing and Engineering Technologies (ICECET), 2023, pp. 1-5, doi: 10.1109/icecet58911.2023.10389186.
- [9]. W. Dong, "Faster R-CNN and YOLOv3: A General Analysis Between Popular Object Detection Networks," *Journal of Physics: Conference Series*, vol. 2580, no. 1, pp. 012016, 2023, doi: 10.1088/1742-6596/2580/1/012016.
- [10]. W. Sheng, X. Yu, and X. Chen, "Faster RCNN Target Detection Algorithm Integrating CBAM and FPN," *Applied Sciences*, vol. 13, no. 12, p. 6913, 2023, doi: 10.3390/app13126913.
- [11]. W. Li, "Analysis of Object Detection Performance Based on Faster R-CNN," *Journal of Physics: Conference Series*, vol. 1827, no. 1, pp. 012085, Mar. 2021, doi: 10.1088/1742-6596/1827/1/012085.
- [12]. D. Avola, L. Cinque, A. Diko, A. Fagioli, G. L. Foresti, A. Mecca, D. Pannone, and C. Piciarelli, "MS-Faster R-CNN: Multi-Stream Backbone for Improved Faster R-CNN

- Object Detection and Aerial Tracking from UAV Images," *Remote Sensing*, vol. 13, no. 9, p. 1670, 2021. doi: [10.3390/rs13091670](https://doi.org/10.3390/rs13091670).
- [13]. O. Hmidani and E. M. I. Alaoui, "A comprehensive survey of the R-CNN family for object detection," in *2022 International Conference on Communications and Networking (CommNet)*, 2022, pp. 1-6. doi: 10.1109/CommNet56067.2022.9993862.
- [14]. R. A. Boukabouya, A. Moussaoui, and M. Berrimi, "Vision Transformer Based Models for Plant Disease Detection and Diagnosis," in *2022 5th International Symposium on Informatics and its Applications (ISIA)*, Nov. 2022, pp. 1-6.
- [15]. <https://www.analyticsvidhya.com/blog/2024/10/yolov11-object-detection/>
- [16]. Karim, Shahid & Zhang, Ye & Yin, Shoulin & Bibi, Irfana. (2020). A brief review and challenges of object detection in optical remote sensing imagery. *Multiagent and Grid Systems*. 16. 227-243. 10.3233/MGS-200330.
- [17]. Evolution toward intelligent communications: Impact of deep learning applications on the future of 6G technology - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-MobileNetV3-architecture-and-its-core-components_fig4_375462137 [accessed 4 Dec 2024]
- [18]. <https://medium.com/the-advantages-of-densenet/the-advantages-of-densenet-98de3019cdac>
- [19]. <https://www.scaler.com/topics/residual-networks-resnet-deep-learning/>

Appendix:

1) Usage of ViT Classification

In the previous semester, we utilized the Vision Transformer (ViT) model for image classification tasks, particularly in rice leaf disease detection. The ViT model leverages self-attention mechanisms that allow it to capture complex spatial dependencies within images, which is essential for distinguishing various disease types on rice leaves. By dividing the image into non-overlapping patches and processing them through multiple layers of attention, the ViT model learns hierarchical representations that contribute to high accuracy in classification tasks. This model was integrated into the AGRI GUARDIAN project to classify rice leaf diseases, providing an effective baseline before integrating object detection models like Faster R-CNN for more precise localization and classification of diseases.

2) Custom Model Architecture

For the rice leaf disease detection task, we designed a custom model that combines several advanced blocks, aiming to improve both accuracy and efficiency in detection. The architecture is composed of the following components:

1. **Backbone (ResNet-50):** We chose ResNet-50 as the backbone due to its deep architecture, which excels at extracting hierarchical features from images. The residual connections in ResNet-50 help mitigate the vanishing gradient problem, enabling effective training even with deep networks. This backbone helps the model learn complex patterns and generalize better, making it a robust feature extractor for our task.
2. **Spatial Attention Block:** The Spatial Attention block was incorporated to enable the model to focus on specific regions of the image that are important for detecting rice leaf diseases. This block generates a spatial attention map, which highlights critical areas and suppresses irrelevant parts of the image. This allows the model to allocate resources more effectively, improving both accuracy and computational efficiency.
3. **Channel Attention Block:** The Channel Attention block enhances the model's ability to focus on important feature channels by re-weighting the channels according to their relevance. This block helps the model to adaptively prioritize feature channels that are critical for disease detection, contributing to improved performance by ensuring that more informative features are emphasized.
4. **Transformer Attention Block:** We introduced a Transformer Attention mechanism to capture long-range dependencies in the image. The Transformer Encoder layer processes the output from the backbone, allowing the model to focus on relationships between distant pixels in the image. This attention mechanism is particularly useful for detecting diseases

where contextual information from different regions of the image is necessary for accurate classification and localization.

5. **YOLO Style Detection Head:** To perform object detection, we used a YOLO style detection head. This head is responsible for generating bounding box predictions and classifying the objects within those boxes. The detection head includes several convolutional layers that predict both the coordinates of the bounding boxes and the corresponding class labels. This architecture is designed to output a set of potential disease locations along with their predicted class, directly addressing the task of object detection in rice leaf disease images.

Model Summary

- **Total Parameters:** 43,810,949
- **Multiplies and Adds (G):** 67.61
- **Input Size (MB):** 9.83
- **Forward/Backward Pass Size (MB):** 2906.89
- **Parameters Size (MB):** 150.03
- **Total Size (MB):** 3066.75

Loss Function and Implementation Challenges: For training the custom model, we used the YOLO loss function, which combines classification and bounding box regression losses to detect and localize diseases. While implementing the loss, we faced challenges balancing the loss components, handling class imbalance, fine-tuning anchors, and adjusting non-maximum suppression (NMS), which impacted model convergence and accuracy.

We are currently working on refining the YOLO loss function, focusing on dynamic anchor adjustments and advanced NMS techniques to enhance detection accuracy and reduce false positives. Once resolved, we expect these improvements to significantly boost the model's performance in real-world applications, particularly in detecting rice leaf diseases with higher accuracy and fewer computational resources.

Plagiarism Check Report:

Similarity

Submission date: 04-Dec-2024 09:35PM (UTC+0530)

Submission ID: 2540547970

File name: P4REPORT_1_.pdf (1.38M)

Word count: 10214

Character count: 55918

P4REPORT (1).pdf

ORIGINALITY REPORT

5%

SIMILARITY INDEX

2%

INTERNET SOURCES

4%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to Canterbury Christ Church
University

Student Paper

<1%

2

Submitted to University of New Haven

Student Paper

<1%

3

Submitted to Liverpool John Moores
University

Student Paper

<1%

4

de Almeida Henriques Pereira, Francisco
Jorge. "Forehead Detection in Thermal
Camera Using Deep Learning", Universidade
do Porto (Portugal), 2024



Publication

<1%

AI Check

Manisha Saini

P4REPORT (1).pdf

 nikhil_report
 Gen AI and LLMs
 BML Munjal University

Document Details

Submission ID
trn:oid::1:3102056697

Submission Date
Dec 4, 2024, 9:35 PM GMT+5:30

Download Date
Dec 4, 2024, 9:37 PM GMT+5:30

File Name
P4REPORT_1_.pdf

File Size
1.4 MB

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.
