

Agr Guardian

PROTECTING CROPS, EMPOWERING FARMERS



INTRODUCTION

- Crop diseases pose a significant threat to global food security by reducing crop health and productivity.
- Conventional detection methods are manual and time-consuming, relying on visual inspection by experts.
- An innovative application leveraging deep learning to detect leaf diseases directly through a camera feed.
- Rice is a staple crop cultivated extensively across India, making it an ideal focus for impactful disease detection.

Our Objective : Develop a user-friendly application for accurately identifying rice leaf diseases using advanced deep learning techniques.

LITERATURE REVIEW

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
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

LITERATURE REVIEW

Paper Title	Dataset	Methodology	Results	Limitations
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 variants like R-CNN, Fast R-CNN, and Faster R-CNN.	Faster R-CNN emerged as a benchmark for high accuracy with significant speed improvements over its predecessors.	Real-time performance remains a challenge despite improvements.
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.
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.

LITERATURE REVIEW

Paper Title	Dataset	Methodology	Results	Limitations
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.

Datasets

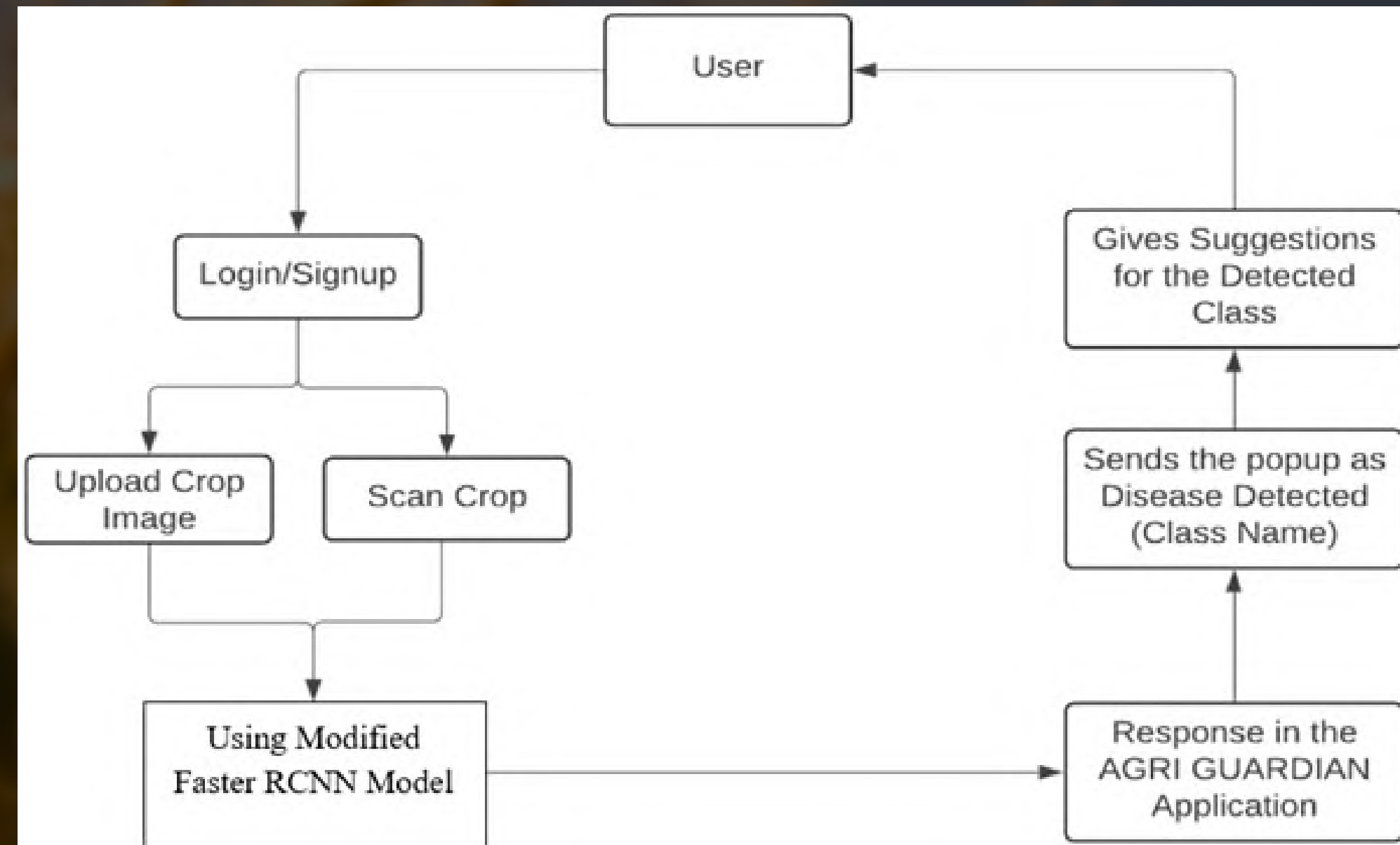
1. **Kaggle Dataset**: 3,678 rice leaf images across six classes. Images were sourced from diverse conditions, annotated using Roboflow for consistency.
 2. **Dataninja Dataset**: 373 rice leaf images across three classes. Pre-annotated but had class imbalance, requiring special handling.
- **Bacterial Leaf Blight**: Bacterial infection causing wilting and lesions, reducing grain yield.
 - **Brown Spot**: Fungal disease with brown lesions, affecting photosynthesis and grain quality.
 - **Healthy**: Non-infected rice leaves for comparison.
 - **Leaf Blast**: Fungal infection with oval lesions, significantly reducing yield.
 - **Leaf Scald**: Fungal disease with long, moist lesions, inhibiting growth.
 - **Narrow Brown Spot**: Fungal infection with dark, elongated lesions, lowering yield potential.



Categories of rice leaf diseases

METHODOLOGY

FLOW CHART



Workflow of AGRI GUARDIAN application.

METHODOLOGY

Algorithm Discussion

YOLO v11

1. Backbone

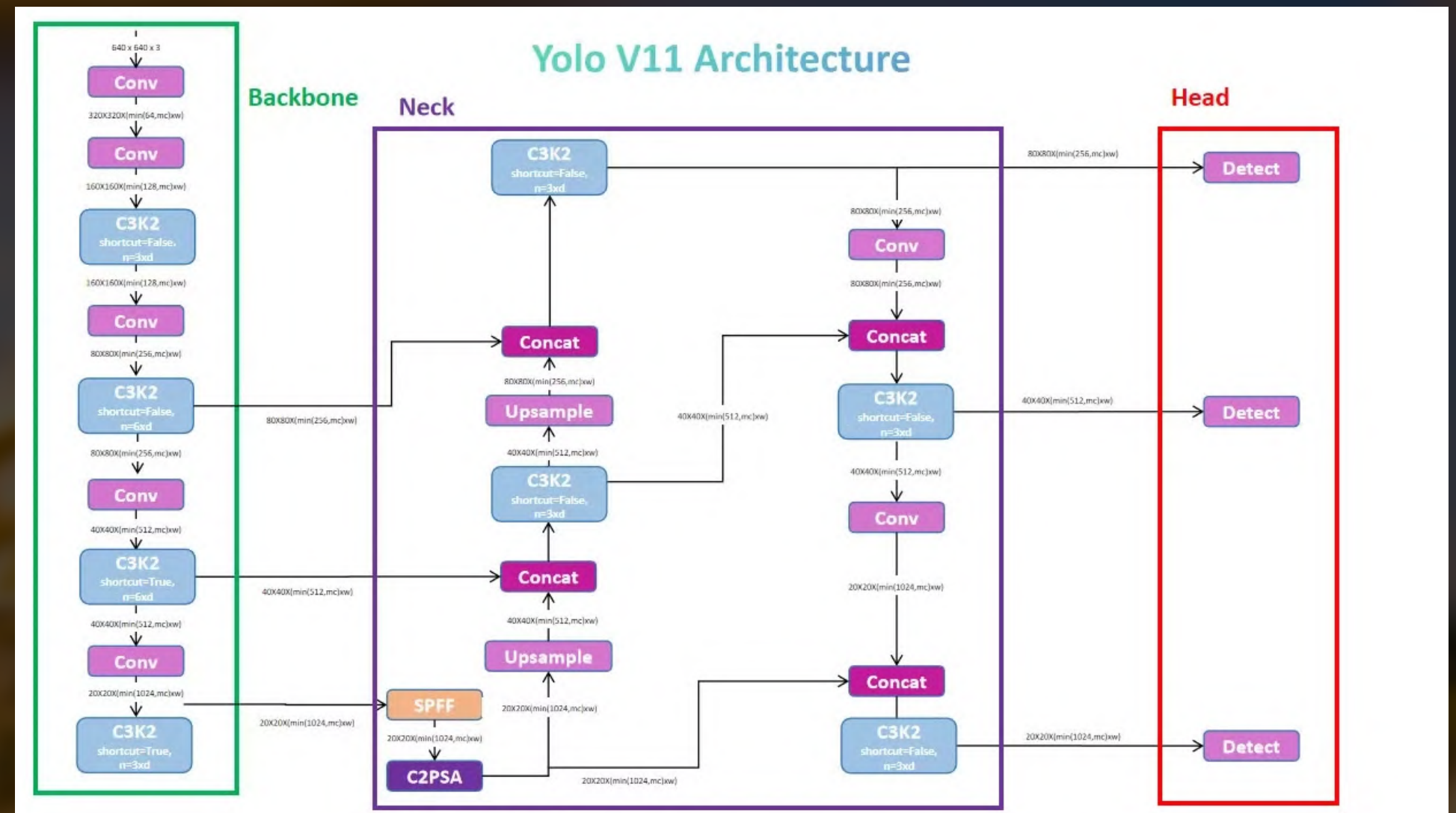
- Extracts features through convolutional layers and **C3k2** modules.
- C3k2 blocks:**
 - Use residual connections and bottlenecks for improved feature flow and gradient handling.
 - Capture patterns for detecting subtle rice leaf diseases.

2. Neck

- Aggregates multi-scale features to enhance detection.
- Key Modules:**
 - SPPF:** Captures multi-scale features for detecting varying lesion sizes.
 - C2PSA:** Focuses on critical features for subtle disease patterns.

3. Head

- Generates final predictions (bounding boxes, class probabilities, and confidence scores).
- Detect Module:** Ensures accurate disease localization and classification.



Yolov11 Model architecture

METHODOLOGY

Algorithm Discussion

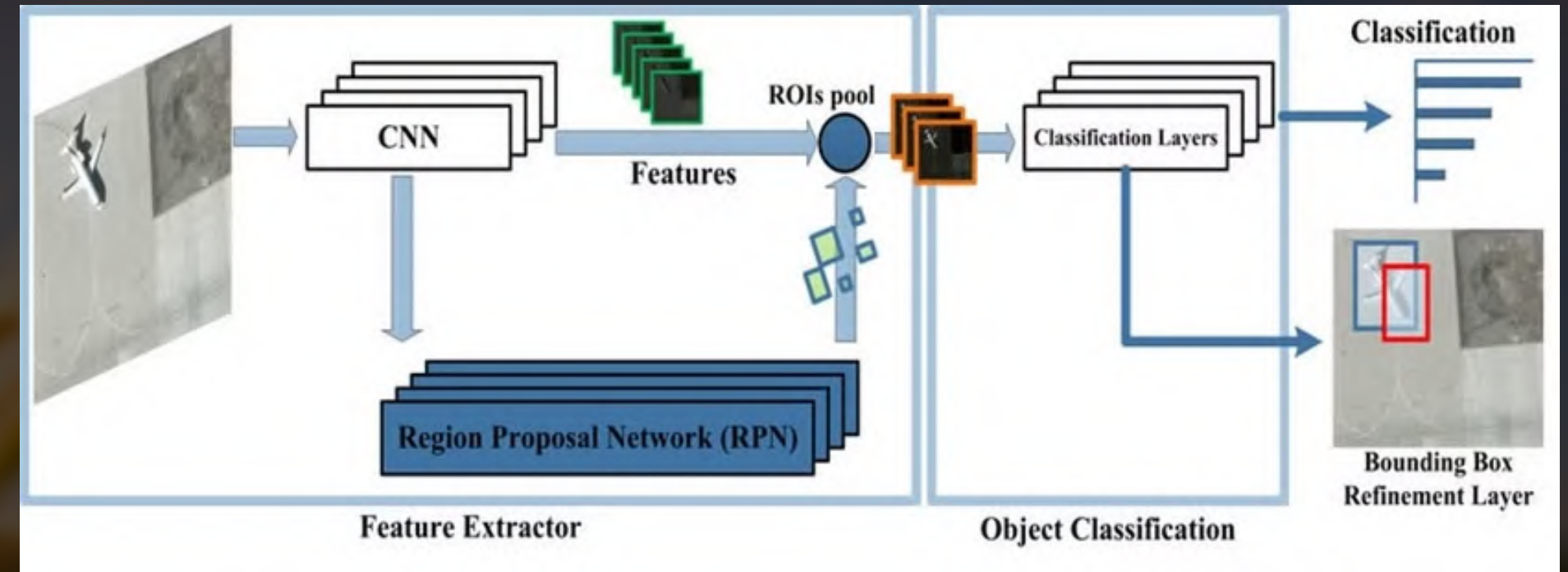
Faster RCNN

1) Backbone

- Extracts hierarchical features, from edges to high-level patterns.
- Focuses on disease-specific features like lesions & discoloration.

2) Region Proposal Network (RPN)

- Generates bounding box candidates using anchor boxes for scale variations.
- Key Layers:
 - **Classification:** Detects object presence.
 - **Regression:** Refines bounding box coordinates



Faster RCNN Model architecture

3) ROI Pooling

- Converts variable-sized proposals into fixed-size feature maps.
- **ROI Align** improves precision with reducing misalignment.

4) Detection Head

- Classifies objects and refines bounding box coordinates.
- Outputs:
 - **Class Scores:** Predict disease types.
 - **Bounding Box Regression:** Fine-tunes coordinates for accuracy.

METHODOLOGY

Algorithm Discussion

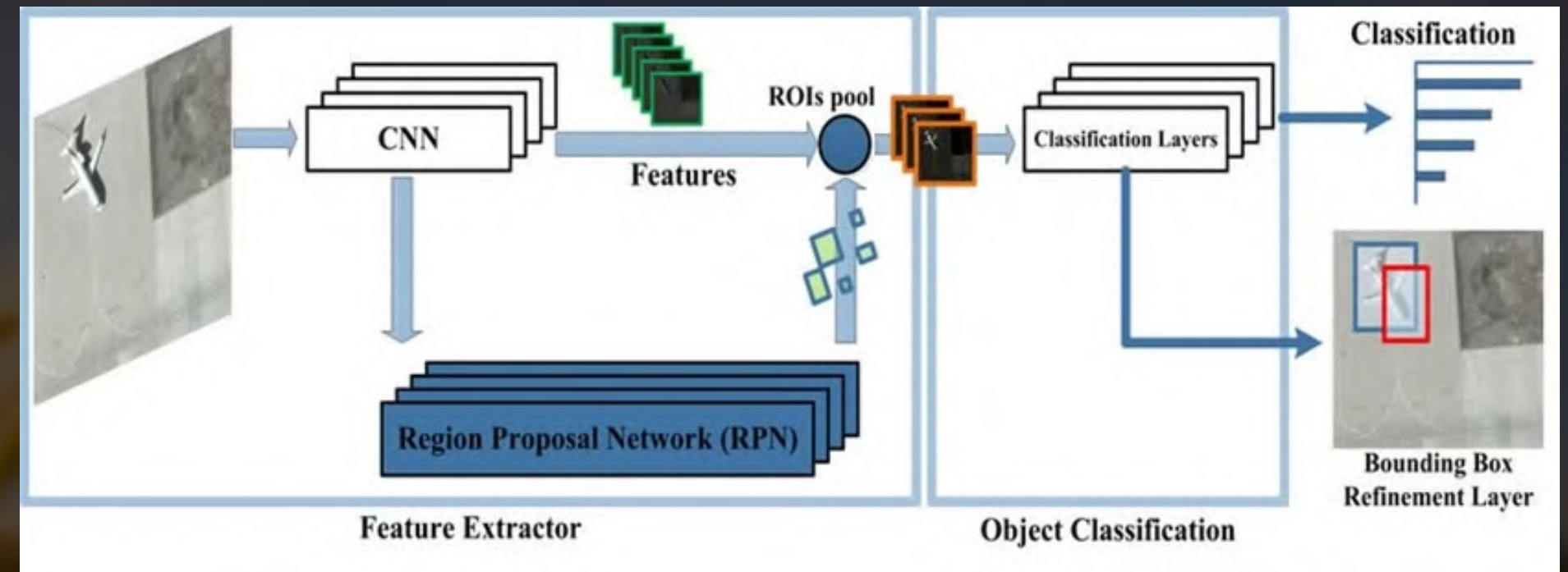
Faster RCNN

Why Faster R-CNN?

- High accuracy for detecting subtle disease symptoms.
- Pretrained backbones handle diverse conditions.
- Effective for multi-class identification and precise localization.

Optimization

- **Loss Functions:**
 - **Cross-Entropy Loss:** Ensures accurate object classification.
 - **Smooth L1 Loss:** Refines bounding box coordinates with robustness to outliers.
- **Optimizer:**
 - **Stochastic Gradient Descent (SGD):** Efficiently updates model parameters for balanced performance.
- **Learning Rate Scheduler:**
 - **StepLR:** Gradually reduces learning rate during training for stable convergence.



Faster RCNN Model architecture

METHODOLOGY

Algorithm Discussion

Mobilenet v3 Large as Backbone for Faster R-CNN

MobileNetV3-Large: Lightweight and efficient CNN for resource-constrained environments.

Backbone for Faster R-CNN: Effectively extracts features with low computational overhead, ideal for real-time object detection.

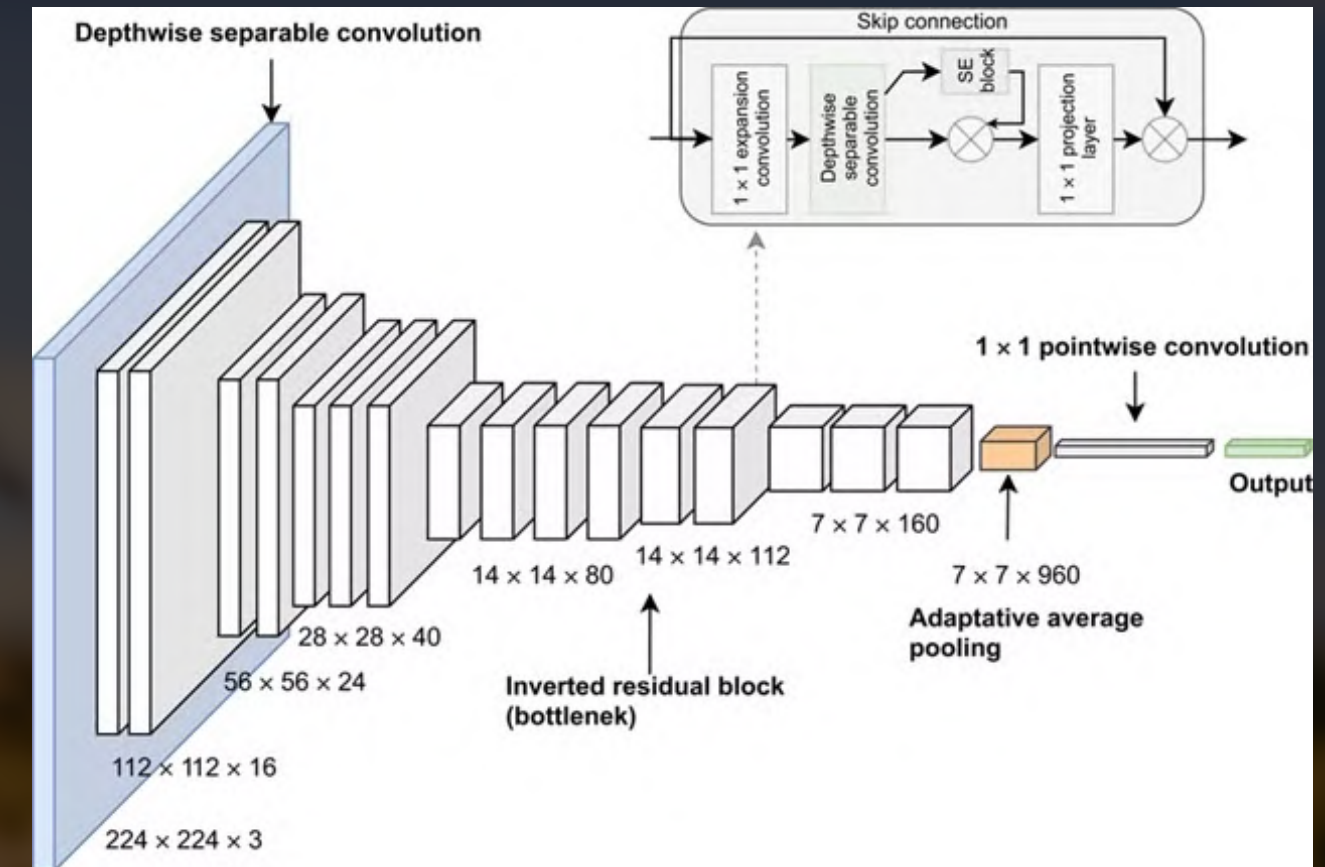
Key Components:

- **Depthwise Separable Convolutions:** Reduces computation by applying convolutions independently to each channel.
- **Squeeze-and-Excitation (SE) Blocks:** Perform channel-wise attention to focus on informative features.
- **Hard-Swish Activation:** Computationally efficient, enabling faster processing.

Advantages:

- Balance of **performance** and **efficiency**.
- Ideal for real-time, lightweight tasks like rice leaf disease detection.
- **Fast inference** for timely diagnosis in agricultural contexts.

Ideal for low-resource devices: Modular design allows scalability without sacrificing performance.



[MobileNet v3 Large Model architecture](#)

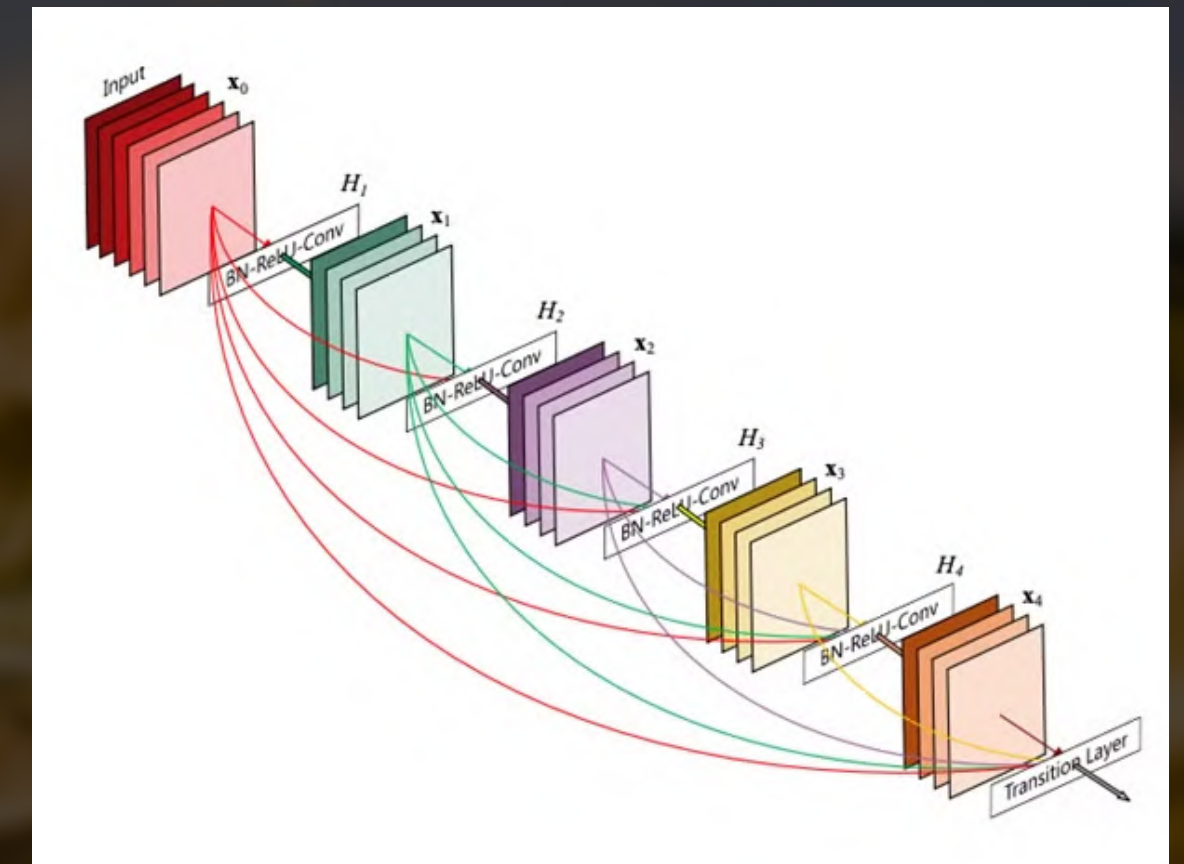
METHODOLOGY

Algorithm Discussion

Densenet121 as Backbone for Faster R-CNN

DenseNet121 Architecture: Enhances feature propagation by connecting each layer to every other layer, promoting feature reuse.

- **Backbone for Faster R-CNN:** Used in Faster R-CNN for feature extraction in the RPN and detection head.
- **Structure:** Comprises four dense blocks with convolutional layers, 1x1 and 3x3 convolutions, and transition layers for down sampling.
- **Global Average Pooling:** Used for classification but omitted in Faster R-CNN.
- **Advantages for Object Detection:**
 - Efficient feature reuse for both low- and high-level extraction.
 - Helps detect fine details (e.g., lesions or spots) in complex backgrounds.
 - Memory efficient and mitigates the vanishing gradient problem.
 - Enables faster convergence and robust performance in medium-sized datasets.



[DenseNet121 Model architecture](#)

METHODOLOGY

Algorithm Discussion

ResNet-50 as Backbone for Faster R-CNN

ResNet-50 Backbone for Faster R-CNN:

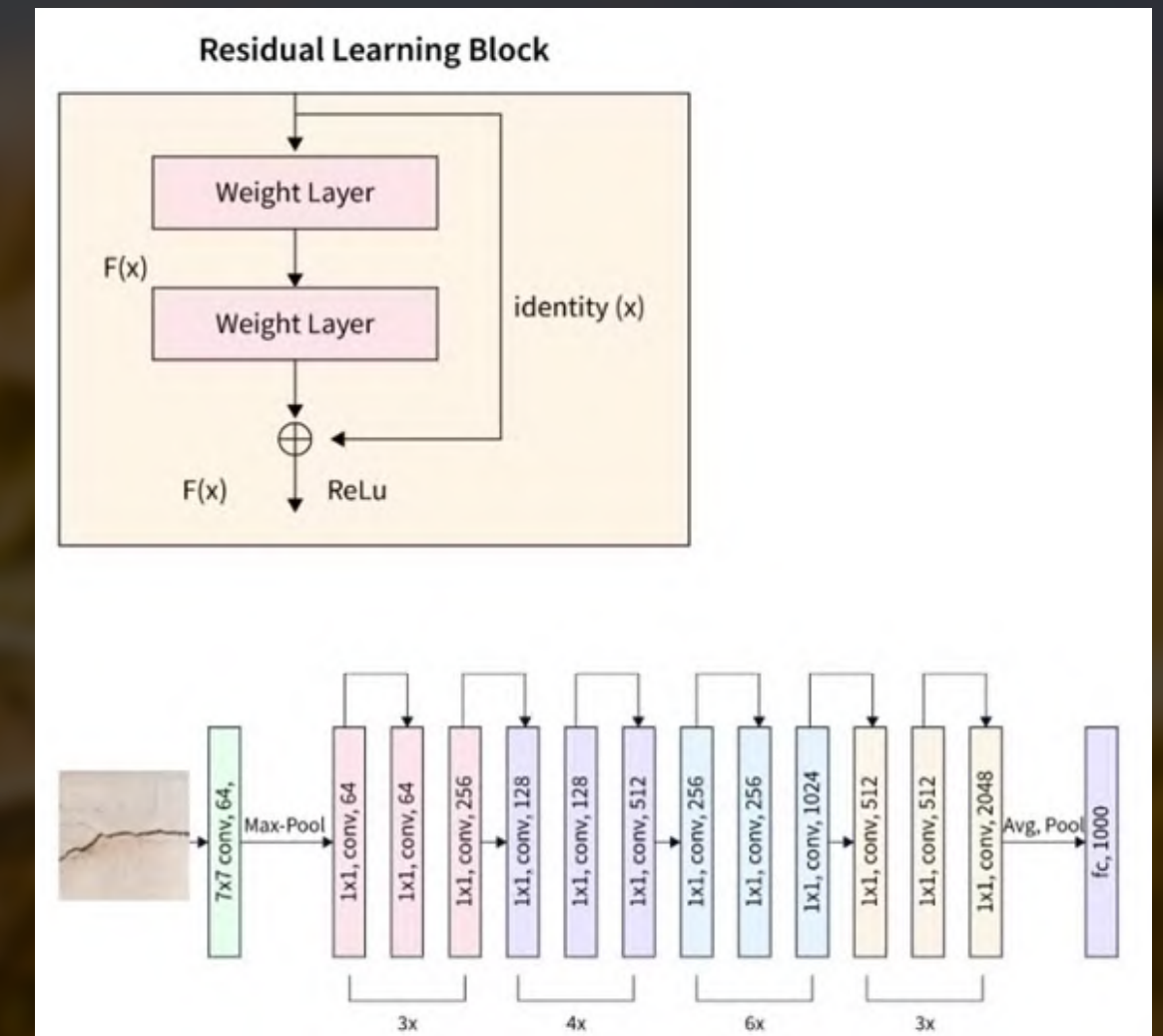
- Overcomes vanishing gradient issues, ideal for object detection tasks like rice leaf disease.
- Extracts hierarchical features for detecting small and large objects.

Structure:

- 5 stages with residual blocks: 1x1 (dimensionality reduction), 3x3 (feature extraction), 1x1 (restoration).
- Shortcut connections for efficient training.

Advantages:

- Pretrained on ImageNet, balancing accuracy and efficiency.
- Detects subtle symptoms like bacterial blight and sheath blight.
- Ensures precise localization and classification of symptoms.



[ResNet 50 Model architecture](#)

METHODOLOGY

Algorithm Discussion

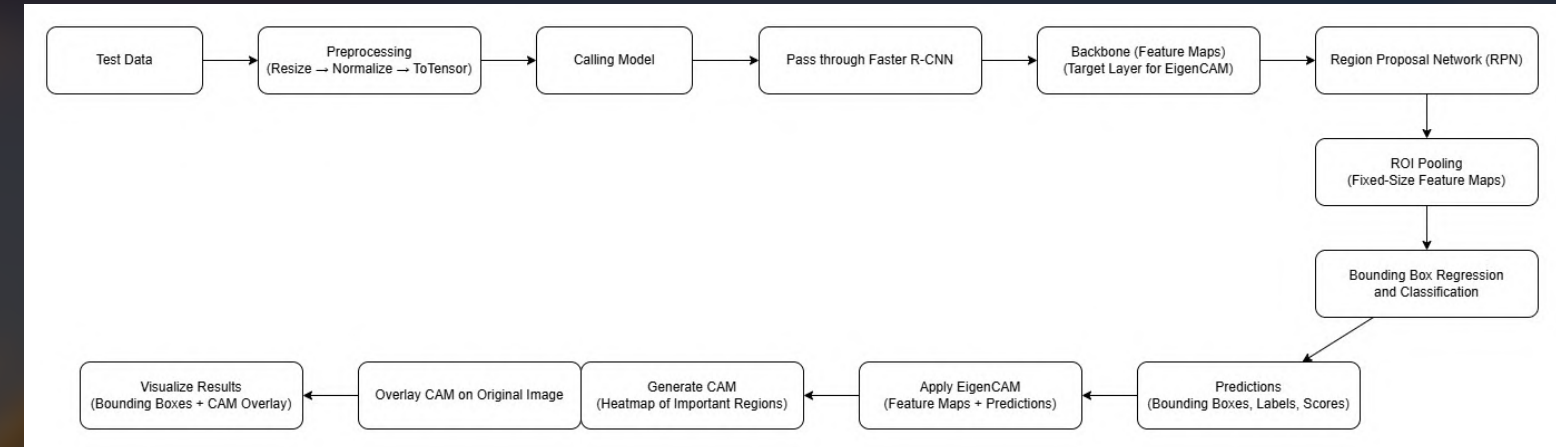
Eigen-CAM for Explainability

Explainability in AI:

- Refers to making models decisions understandable to humans.
- Helps build trust and transparency in AI models, especially in critical applications like healthcare or agriculture.
- Eigen-CAM (Eigen Class Activation Mapping) is a technique that highlights areas in an image that contribute most to a model's prediction.
- It uses the model's internal features to create a visual map of important regions in the image.

Use Eigen-CAM:

- Improves model transparency.
- Helps detect errors or biases in model predictions.
- Ensures the model is focusing on meaningful features, leading to better and more reliable outcomes.



Flowchart for the EigenCAM Implementation

Experimental Results

Performance Comparison of YOLO v11 and Faster R-CNN

YOLO v11

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

Dataset Results:

- **Kaggle:** Low mAP, precision, and recall.
- **Dataninja:** Performance below expectations.

Faster R-CNN with MobileNet_v3_Large

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

Achieved **significant improvement** in detection metrics.

Faster R-CNN significantly outperformed YOLO v11 in mAP, precision, and recall across both datasets. Faster R-CNN was chosen for optimization, offering higher reliability and accuracy for rice leaf disease detection.

Experimental Results

Optimization of Faster R-CNN Using Different Backbones

Enhance Faster R-CNN performance for rice leaf disease detection by testing different backbone architectures.

Performance on Kaggle Dataset

Backbone	mAP	Precision	Recall
MobileNet v3 Large	0.821	0.928	0.53
DenseNet 121	0.78	0.75	0.64

Performance on Dataninja Dataset

Backbone	mAP	Precision	Recall
MobileNet v3 Large	0.723	0.79	0.76
DenseNet 121	0.66	0.73	0.56

- **MobileNet_v3_Large**: Strong precision, moderate recall.
- **DenseNet121**: Limited adaptability, lower performance.
- **ResNet-50**: Best mAP, precision, and recall, ideal for task.

Experimental Results

Optimizing Faster R-CNN with ResNet-50

- **Transfer Learning:**
 - Leverages pre-trained Faster R-CNN with ResNet-50, saving time and resources.
 - Quickly adapts to rice leaf disease detection by retaining general features (textures, shapes).
- **Freezing Layers:**
 - Freezes early layers (conv1, layer1 in ResNet-50) to preserve general features.
 - Reduces training time, prevents overfitting, and conserves resources.
- **Feature Pyramid Network (FPN) Adjustment:**
 - Freezes initial layers to reduce complexity while retaining detection capability for different leaf sizes.
- **Region Proposal Network (RPN) Adjustment:**
 - Adjusts RPN parameters (anchors, NMS threshold) for better efficiency.
 - Reducing the number of anchors to 6 and setting NMS threshold to 0.6 to balance accuracy and redundancy.

Experimental Results

Training Efficiency & Performance Analysis

- **Training Time Efficiency:**
 - Epoch time reduced from 9 minutes to 8 minutes 45 seconds (15 seconds saved per epoch).
 - 2.5 minutes saved per session (10 epochs).

Performance on Kaggle Dataset

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

Performance on Dataninja Dataset

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

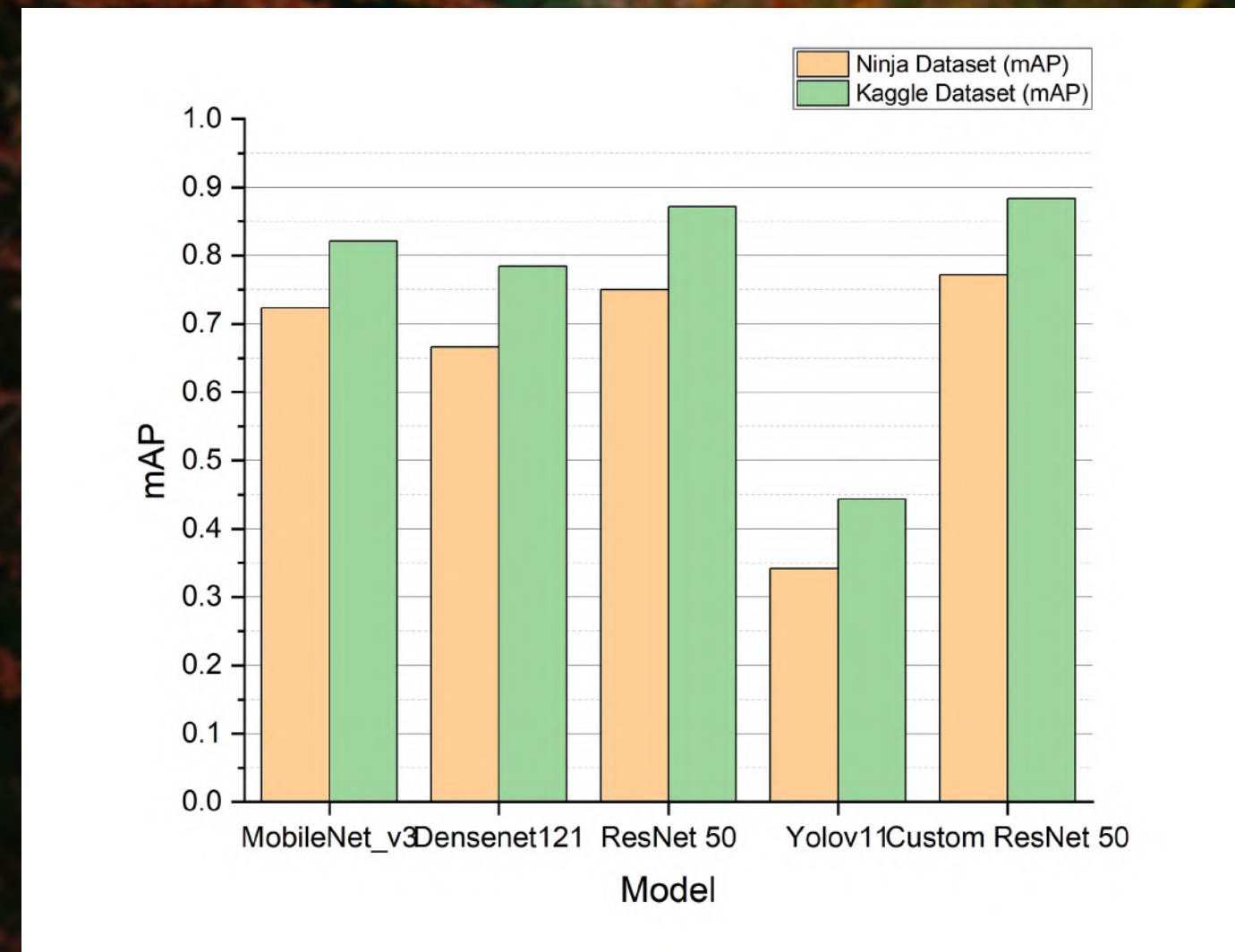
Model Integration:

- Faster R-CNN model saved and integrated into AGRI GUARDIAN app for disease detection.
- Deployed on the Indus Appstore for user access.

Qualitative Analysis

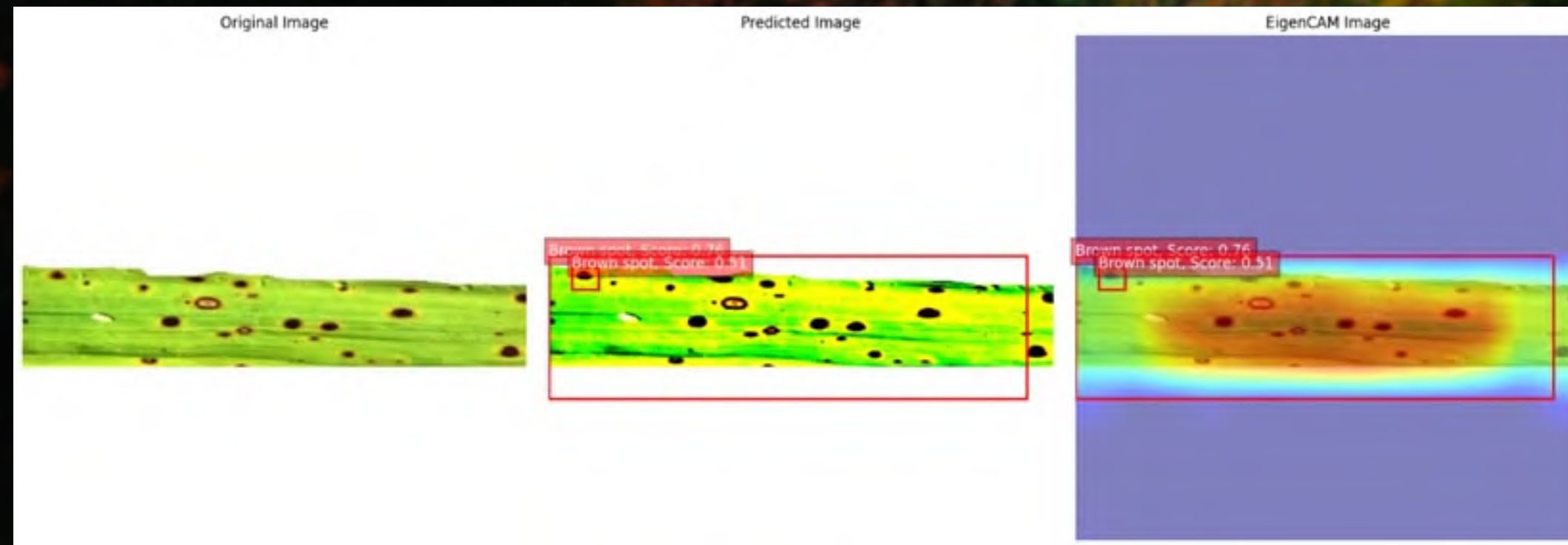
- **Model Performance Comparison**
 - Improvement in detection accuracy with modified Faster R-CNN.
 - Visual comparison of mAP scores across YOLO v11, Faster R-CNN with different backbones, and the modified model.

mAP Comparison

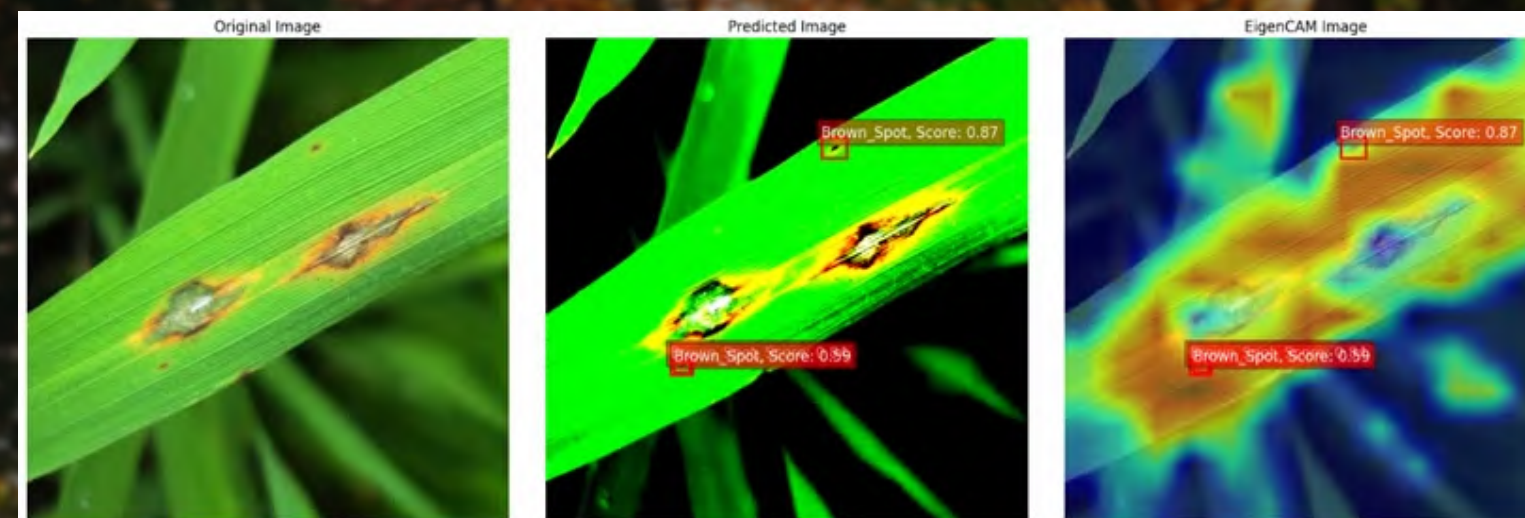


Modified Faster R-CNN shows significant improvement on Kaggle and Dataninja datasets.

Qualitative Analysis



Detection Results of Modified Faster R-CNN on Kaggle Dataset



Detection Results of Modified Faster R-CNN on Dataninja Dataset

Appendix

1) Custom Model Architecture:

- **Backbone (ResNet-50):** Deep architecture with **residual connections**.
- **Spatial Attention Block:** Focuses on relevant image regions.
- **Channel Attention Block:** Re-weights important feature channels.
- **Transformer Attention Block:** Captures **long-range dependencies**.
- **YOLO Style Detection Head:** Predicts bounding boxes and class labels.


Model Summary:

- **Parameters:** 43.81M | **Input Size:** 9.83MB | **Total Size:** 3066.75MB

Loss Function & Challenges:

- **YOLO Loss:** Combines classification and bounding box regression losses.
- **Challenges:** **Class imbalance**, **anchor tuning**, **NMS adjustments**.
- Refining loss function to improve **accuracy** and reduce **false positives**.

DEPLOYMENT



List New App

Home

Dev Tools

Developer Page


Support

Policy Center

Home >

Home

Search by app or package name


Published		Version	Updated on	Installs (30 Days)	Status	Action
	Agri Guardian: Protecting Crops, Empowering Farmers com.example.project_2	1.0.0	12/4/2024		Published	View →

12:45

< > ⌵ ⌵ ⌵ ⋮

Your app - Agri Guardian: Protecting Crops, Empowering Farmers will be live on Indus Appstore in 24 hours. [Inbox](#)

 Indus Appstore 5:06 PM
to me ▾



App Review Successful

Hello D Veera Harsha Vardhan,

We are pleased to inform you that your app, Agri Guardian: Protecting Crops, Empowering Farmers, has successfully passed our review process.

Your app is now live on the Indus Appstore Developer Platform.

You can access your account here:

Indus Appstore Developer Platform

If you have any questions or need assistance, please reach out to our support team at:
support@indusappstore.com

DEMO

RESULTS

THANK YOU





Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)