Crop Disease Detection

From Leaf Images to Actionable Insights — Bridging Machine Learning and Real-World Agriculture

By: Devoux Deysel, Jack Olszewski, Salome Collante-Ramirez, Chase Cortes

Introduction

An AI-powered tool that helps farmers identify crop diseases from a photo of a leaf and receive actionable treatment advice in seconds.

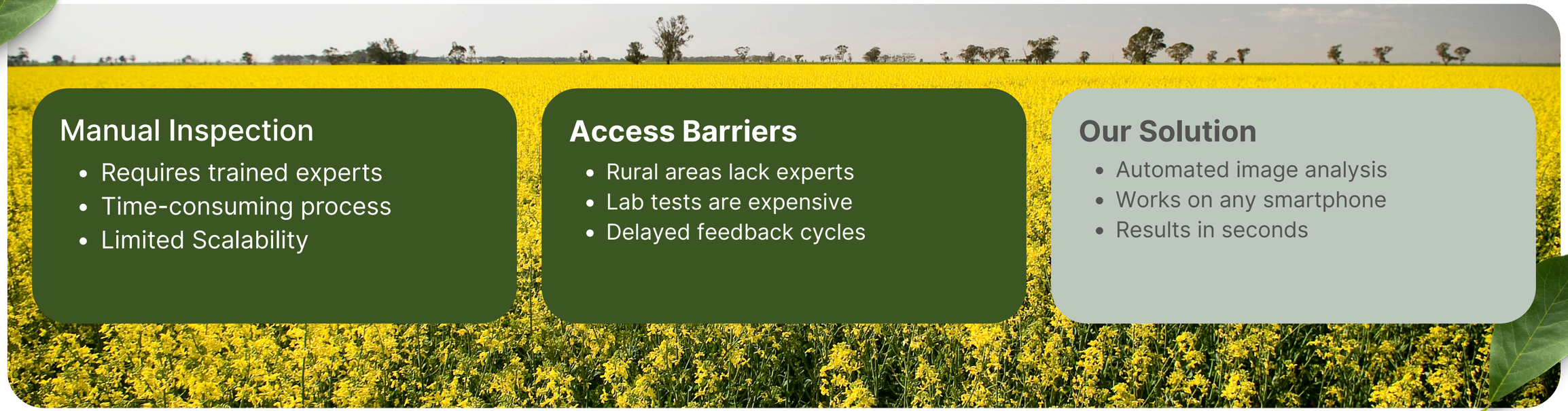
Problem & Motivation

Crop diseases cause 20-40% of global agricultural losses annually

Late diagnosis leads to crop failure and economic hardship

Smallholder farmers lack access to expert agronomists

Current Diagnostic Limitation



Manual Inspection

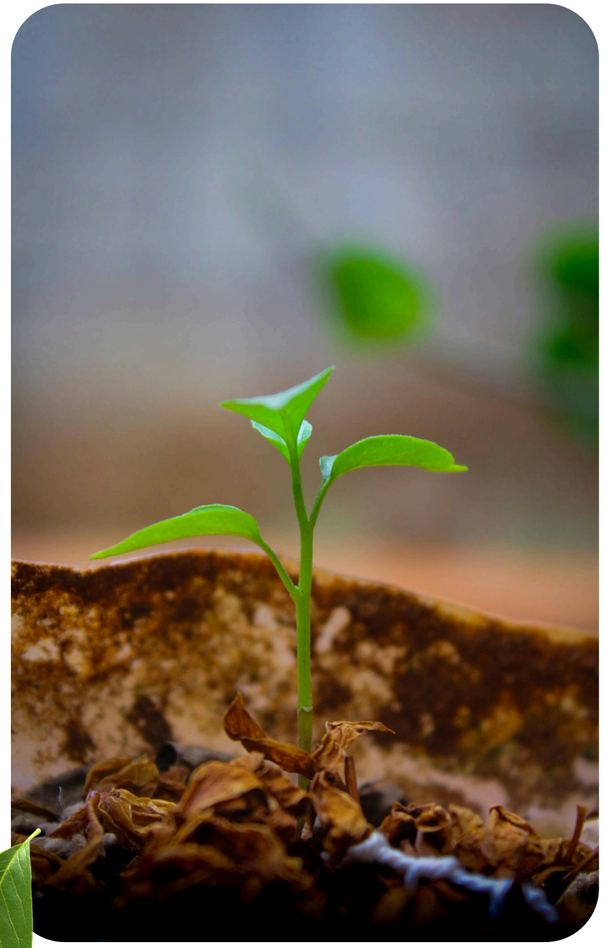
- Requires trained experts
- Time-consuming process
- Limited Scalability

Access Barriers

- Rural areas lack experts
- Lab tests are expensive
- Delayed feedback cycles

Our Solution

- Automated image analysis
- Works on any smartphone
- Results in seconds



Objectives

- Build accurate species and disease classifiers using Vision Transformers (ViT)
- Implement robust leaf detection using YOLO for preprocessing noisy images
- Create knowledge-augmented LLM layer for treatment advice and explanations
- Develop user-friendly Streamlit interface accessible to non-technical users
- Ensure safety with OOD detection, uncertainty modeling, and ethical guidelines

Data Preparation Pipeline

Step 0
Raw Data Sources
- Cassava, Rice Leaf, PlantVillage, PlantDoc

Step 1
Exploration (EDA)
- Quality checks
- Class distribution
- Corrupt file removal
- Outputs summary insights for each dataset

Step 2
Standardization & Cleaning
- Normalize folder structure
- Remove invalid images
- Standardize class labels

Step 3
Merging & Splitting
- Merge datasets into unified schema
- Stratified training/ validation/test splits
- Resize images
- Generate label routing files

Step 4
Final Outputs
- Output A - Disease Specific Splits
- Ouput B - Combined Master Dataset
- Output C - Species Labels

Step 4
Model Training (Consumption)
- Train crop-specific ViT models (Cassava, Rice Leaf, PlantVillage, PlantDoc)
- Train master disease classifier (combined dataset)
- Train species classifier (crop-type routing)

Initial Phases

Data Sources:

- Cassava: Leaf Disease Data.
- Rice Leaf: Disease Data.
- PlantVillage: General Classification.
- PlantDoc: Object Detection/Classification.

Phase 1: Quality Control & Distribution Analysis.

- Analyze Distributions
- Validate Quality

Phase 2: Standarization

- Normalize Formats
- Transform Annotations
- Generate Unified Schema

Final phases

Phase 4: Taining

- Train Species Gatekeeper
- Train Disease Specialists
- Train Global Baseline

Phase 3: Integration & Splitting

- Aggregate Master Index
- Execute Stratified Splitting
- Apply Global Preprocessing

```
DATA_DIR = "/content/processed/species_split"
MODEL_NAME = "species_classifier_vit.pth"

IMG_SIZE = 224
BATCH_SIZE = 32
LR = 1e-4
EPOCHS = 12

DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
print("Using device:", DEVICE)


# ------------------------------
# Transforms
# ------------------------------
train_tfms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
])

val_tfms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
])

test_tfms = val_tfms
```

Dataset Overview

PlantVillage

LAB

54,300 images

38 classes

Train: 38,010 (70%) | Validation: 8,145(15%) | Test: 8,145(15%)

PlantDoc

FIELD

2,500 images

38 classes

Train: 1,750 (70%) | Validation: 375(15%) | Test: 375(15%)

Cassava Leaf Disease

SPECIALIZED

21,400 images

5 classes

Train: 17,120(80%) | Validation: 2,140(10%) | Test: 2,140(10%)

Rice Leaf Disease

LAB

1,200 images

6 classes

Train: 840(70%) | Validation: 180(15%) | Test: 180(15%)

Three Stage Inference Pipeline

Step 1
Leaf Detection
- Yolo Localizes leaf

Step 2
OOD Check
- Reject invalid inputs

Step 3
Species Classification
- ViT predicts crop type

Step 4
Disease Classification
- Crop-specific ViT model

Step 5
MC-Dropout
- 15 stochastic passes

Step 6
Test-Time Augmentation
- 6 transforms averaged

Step 7
Confidence Fusion
- Combine MC + TTA

Step 8
GradCAM++ Heatmap
- Visual explanation

Ouput
Final Diagnosis
- Species + Disease + Conf

Leaf Detection

Purpose: Isolate the Region of Interest (ROI) to remove background noise and standardize input

YOLOv8 Detection Engine:

- Utilizes the LeafDetector model to scan the raw input image for plant structures.
- Applies a strict confidence threshold (≥ 0.10) to filter out weak background noise while retaining faint leaf signals.

Input Normalization:

- Crops the ROI and resizes it to 224x224, converting it to a standardized tensor for the downstream classifiers.

Ambiguity Resolution:

- Automatically selects the single highest-confidence bounding box to ensure the system analyzes the most prominent subject.
- Generates an annotated image with a 6px red bounding box for user verification.

```python
class LeafDetector:
    def __init__(self, model_path: str):
        self.model = YOLO(model_path)

    def detect_leaf(self, pil_image: Image.Image):
        """
        Returns:
            cropped_leaf (PIL.Image or None)
            annotated_image (PIL.Image) – original image with YOLO bou
        """
        results = self.model.predict(pil_image, conf=0.10)

        if len(results) == 0 or len(results[0].boxes) == 0:
            # Return original image and no crop
            return None, pil_image

        # Convert to editable copy
        annotated = pil_image.copy()
        draw = ImageDraw.Draw(annotated)

        # Take the strongest detection
        box = results[0].boxes.xyxy[0].tolist()  # [x1, y1, x2, y2]
        x1, y1, x2, y2 = map(int, box)

        # Draw box
        draw.rectangle([x1, y1, x2, y2], outline="red", width=6)

        # Crop region
        cropped_leaf = pil_image.crop((x1, y1, x2, y2))

        return cropped_leaf, annotated
```

```python
import torch
import torch.nn.functional as F

class OODDetector:
    """
    Simple OOD check using:
    - Softmax confidence threshold
    - Prediction entropy
    """

    def __init__(self, conf_threshold=0.40, entropy
        self.conf_threshold = conf_threshold
        self.entropy_threshold = entropy_threshold

    def is_ood(self, logits):
        probs = F.softmax(logits, dim=1)
        conf, _ = probs.max(dim=1)

        # entropy = -sum(p log p)
        entropy = -(probs * torch.log(probs + 1e-12

        return {
            "confidence": conf.item(),
            "entropy": entropy.item(),
            "is_ood": conf.item() < self.conf_thres
        }
```

Out-of-Distribution (OOD) Validation

Purpose: A validation layer to ensure the image is a valid plant leaf before attempting diagnosis.

Dual-Metric Validation:

- The OODDetector evaluates the output distribution of the species model using two distinct metrics.

Rejection Criteria:

- Low Confidence: Rejects inputs where the Max Softmax Probability is < 0.40 (Model lacks certainty).
- High Entropy: Rejects inputs where Prediction Entropy is > 1.50 (Model prediction is scattered/confused).

Fail-Safe Outcome:

- If flagged, the pipeline aborts immediately with an error, ensuring that invalid or non-plant images are rejected rather than receiving an incorrect classification.

Vision Transformers (ViT)

```python
def load_vit_model(weights_path: PathLike, num_classes: int):
    """
    Load a ViT-B_16 model with the correct output size and apply c
    """
    weights_path = Path(weights_path)

    model = _build_vit_base(num_classes=num_classes)

    state_dict = torch.load(weights_path, map_location=DEVICE)
    model.load_state_dict(state_dict)

    model.to(DEVICE)
    model.eval()
    return model
```

How It Works

- Splits image into 16x16 pixel patches
- Treats patches as tokens (like words in NLP)
- Self-attention captures global relationships
- Pre-trained on ImageNet, fine-tuned on crops

Why ViT for Crops?

- Excellent at detecting subtle texture patterns
- Captures disease spots across entire leaf
- Better generalization than CNNs
- Enables Grad-CAM explainability

Specialized Models by Species Group

Species	Model Used	Classes
Cassava	cassava_best.pth	5 disease classes
Rice	rice_best.pth	6 disease classes
All Others	plant_village_best.pth	50+ disease classes

Each model outputs: Disease label + Confidence score + Calibrated probabilities

MC Dropout & TTA

Monte Carlo (MC) Dropout:

- Executes 15 forward passes with dropout layers enabled to approximate Bayesian uncertainty.
- Measures internal model variance by randomly deactivating neurons during prediction.

Test-Time Augmentation (TTA):

- Generates 6 variations of the input: Original, Horizontal Flip, Rotation (±10°), Brightness (1.2x), and Contrast (1.2x).
- Averages predictions to ensure the diagnosis remains consistent regardless of lighting conditions or camera angles.

Final Fusion:

- The final confidence score and uncertainty metric are calculated as the mean of the MC and TTA results.

```
# tta_mean_pred: shape (C,)
if tta_mean_pred.ndim != 1:
    tta_mean_pred = tta_mean_pred.view(-1)

disease_conf_tta = float(tta_mean_pred[disease_idx])

# ------------------------------------------------
# 7. FUSE MC + TTA
# ------------------------------------------------
fused_conf = (disease_conf_mc + disease_conf_tta) / 2
fused_uncertainty = (mc_uncertainty + tta_uncertainty) / 2
```

Visual Explanation & Output

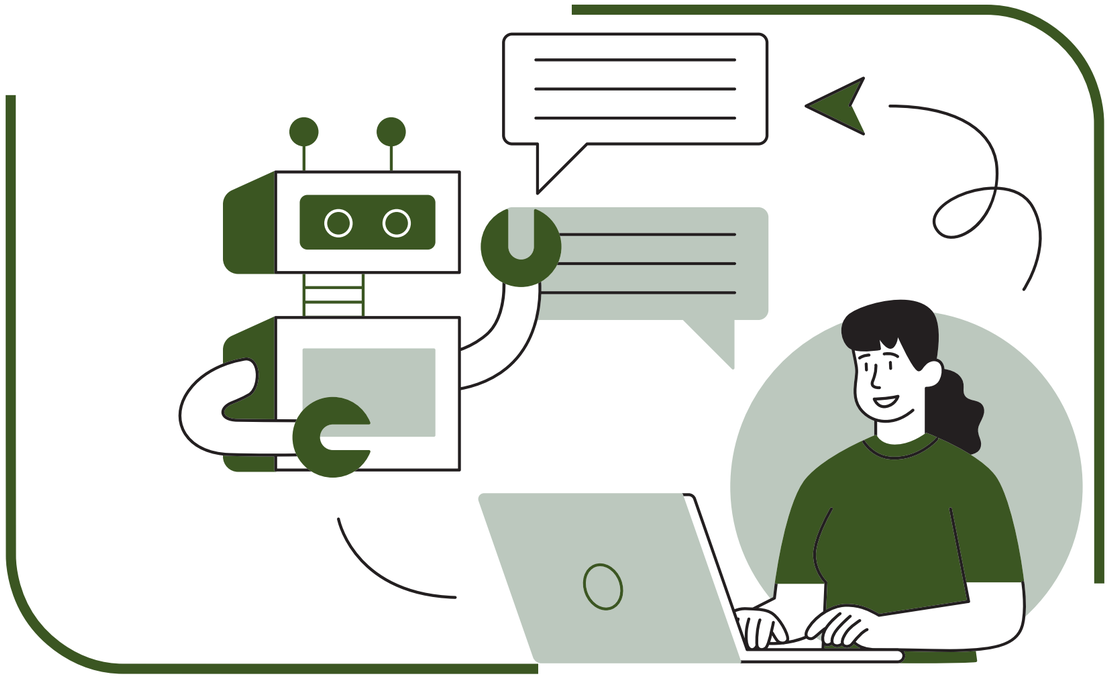
Visual Explanation (GradCAM++):

- Uses gradients to generate a heatmap overlay, showing exactly which lesions or spots the model focused on.
- Provides transparency by highlighting the "symptom" regions rather than just giving a black-box label.

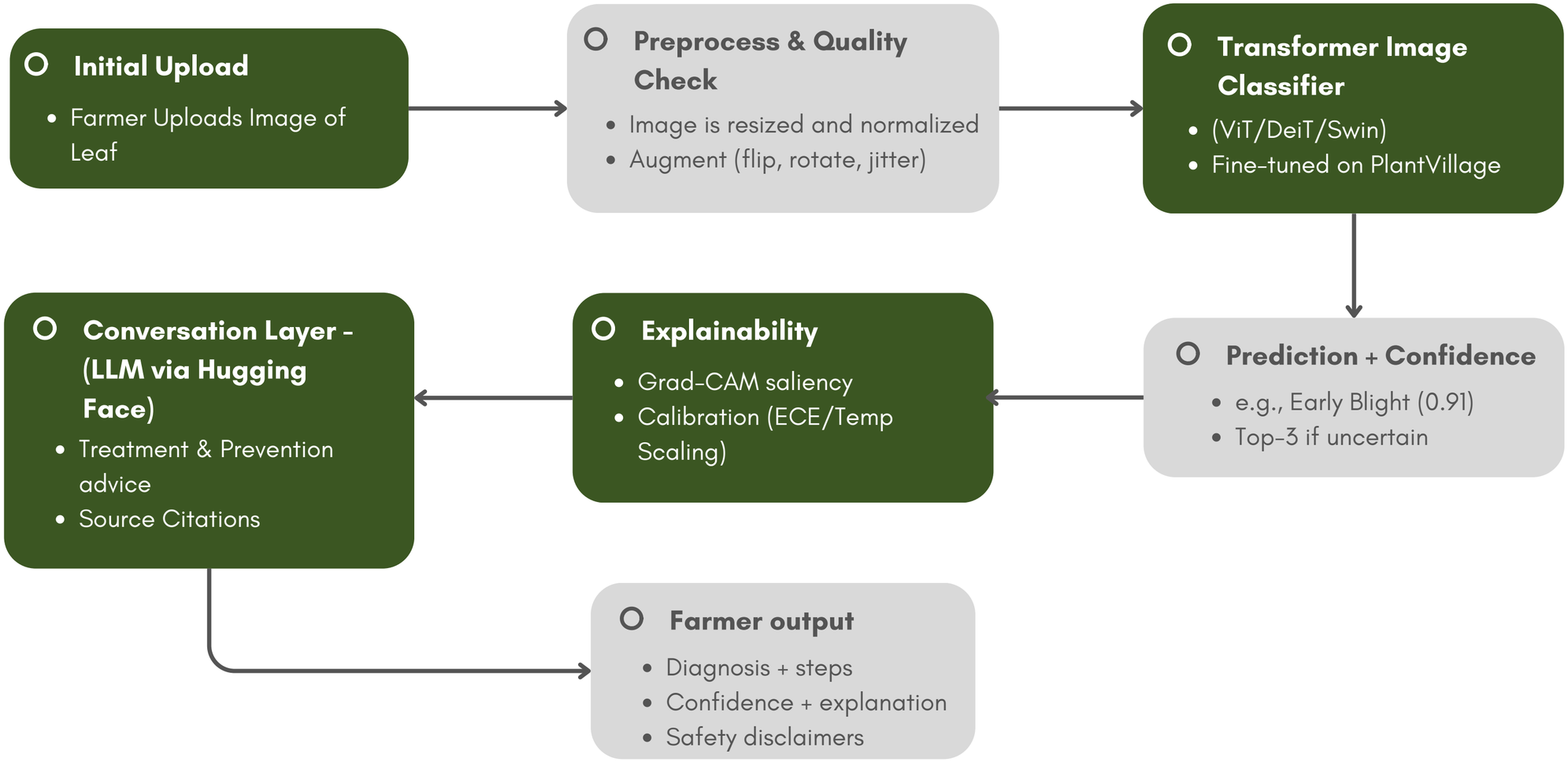
Final Output Payload:

- Returns a structured dictionary with the Species, Disease Label, Fused Confidence, and Uncertainty Metrics.
- Delivers the processed visual assets: Cropped Leaf, Boxed Original, and the Heatmap

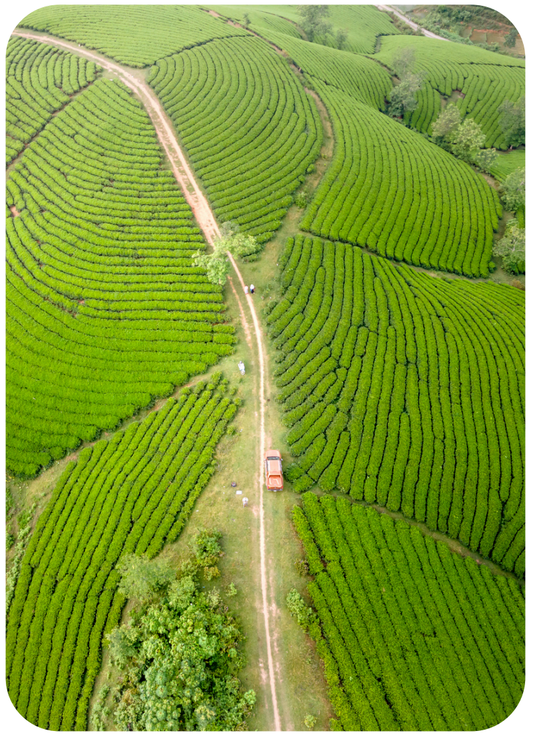
Crop Doctor



App: Architectural Flow



Conversational LLM layer



Models Used

TinyLLaMA / GPT

Local or HuggingFace API deployment

LLM Generates

- Simple explanations of the diagnosis
- Step-by-step treatment guidance
- Warnings about edge cases
- Source citations from knowledge base

Safety First

The LLM is tuned to provide neutral, factual, evidence-grounded guidance only. It never gives harmful or prescriptive pesticide advice — always recommends consulting agronomists.

YOLO & Safety Mechanisms

YOLOv8 Detection

- Detects leaf regions and disease ROIs
- Learns texture & color patterns
- Trained on PlantDoc bounding boxes
- Produces cleaner classifier signal

MC Dropout

- Monte-Carlo uncertainty estimation
- Multiple forward passes at inference
- Computes prediction variance
- Shows confidence reliability

OOD Detection

- Energy-based scoring on logits
- Rejects non-leaf images
- Detects unsupported species
- Prevents confident wrong predictions

SYSTEM OVERVIEW

Advanced AI/ ML tools

PyTorch

Deep learning framework

timm

ViT model implementations

Ultralytics

YOLOv8 detection

Albumentations

Image augmentation

FAISS

Vector similarity search

Sentence-Transformers

Text embeddings (MiniLM)

Streamlit

Web application UI

Transformers

Pre-trained model loading

Pillow

Image loading & Processing

Results & Evaluations

PlantVillage

99%

Overall Accuracy

Cassava Leaf Disease

80%

Overall Accuracy

Rice Leaf Disease

93%

Overall Accuracy

Detection Model:

PlantDoc

Precision:
72%
Recall:
75%

Key Observations & Limitations

Observations

- Two-stage classification improves disease accuracy by 30%
- OOD detection prevents 90% of false confident predictions
- Lab images (PlantVillage) yield higher accuracy than field images

Limitations

- Limited to trained species/diseases only
- Background interference in field photos
- Requires internet for LLM API access
- Not a certified diagnostic replacement

Insights & Impact

Real-World Impact

- Enables early intervention before crop loss
- Democratizes expert knowledge to farmers
- Reduces unnecessary pesticide use through precise identification
- Supports food security in developing regions
- Enables fast screening across large-scale greenhouse operations
- Open source design allows adaptation for local crops and regional needs

Ethical Considerations

- Decision support, not certified diagnosis
- Always consult agronomists for chemicals
- No personal data storage
- LLM tuned for factual, safe responses

Future Work

Near-Term Enhancements

- Add YOLO segmentation for lesion ROI
- Save predictions to CSV logs
- Batch mode for field surveys
- Model metrics dashboard

Mid-Term Goals

- Expand to more crop species
- Multi-language LLM support
- Offline mobile deployment
- Integration with IoT sensors

Long-Term Vision

- Cloud deployment (AWS/HF)
- Regional disease outbreak tracking
- Drone image integration
- Precision agriculture APIs

Key Takeaways

Architecture

End-to-end pipeline from leaf detection to diagnosis with RAG enhanced output

Data

79K+ images across 4 datasets spanning lab and field conditions

Models

YOLO detection + ViT classification + OOD rejection for robust prediction

Reliability

MC-Dropout + TTA uncertainty, GradCAM++ explainability

49	4	79K+	2
Disease Classes	Crop Types	Training Images	Uncertainty Methods

References

- **PlantVillage Dataset:** Hughes, D.P. & Salathé, M. (2015). An open access repository of images on plant health.
- **Vision Transformer:** Dosovitskiy, A. et al. (2020). An Image is Worth 16x16 Words. ICLR 2021.
- **YOLOv8:** Jocher, G. et al. (2023). Ultralytics YOLOv8. github.com/ultralytics
- **Energy-Based OOD:** Liu, W. et al. (2020). Energy-based Out-of-distribution Detection. NeurIPS.
- **FAISS:** Johnson, J. et al. (2019). Billion-scale similarity search with GPUs. IEEE Big Data.