



International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 3, May - June 2026

Impact Factor: 9.971



Agri AI: A Deep Learning-Based Potato Leaf Disease Classification System with Integrated Agricultural Decision Support

Avinash B N

Master of computer Applications, CMR Institute of Technology, Bengaluru, Karnataka, India

ABSTRACT: Crop diseases remain a critical cause of yield loss and financial hardship for farmers, despite agriculture forming the backbone of the Indian economy. Leaf diseases such as Early Blight (*Alternaria solani*) and Late Blight (*Phytophthora infestans*), which spread rapidly under favourable climatic conditions, are particularly devastating to potato crops. Conventional detection methods relying on expert consultation and manual inspection are expensive, time-consuming, and largely inaccessible to farmers in rural and remote areas. This paper presents Agri AI, a web-based Potato Disease Classification System that employs a Convolutional Neural Network (CNN) trained on labelled potato leaf image datasets to automatically classify uploaded images into three categories: Early Blight, Late Blight, or Healthy. Each prediction is accompanied by a confidence score to support informed user decision-making. Beyond disease classification, the system integrates an AI-powered chatbot for agricultural advisory (AgriBot), a fertilizer calculator for nutrient optimization, and real-time weather monitoring for disease risk assessment. Historical scan data is stored and visualised in a dashboard module to facilitate trend monitoring and analysis by agricultural extension workers. The application is implemented using TensorFlow/Keras for model inference, FastAPI for the backend server, SQLite for persistent storage, and a responsive HTML/CSS/JavaScript frontend for user interaction. System testing demonstrated a prediction response time of 1.4 seconds, satisfying the sub-2-second non-functional requirement. All unit, integration, system, and performance tests passed successfully, and User Acceptance Testing indicated strongly positive feed-back from pilot users. The findings confirm that Agri AI is a scalable, usable, and practically effective decision-support tool that promotes smart, sustainable, and data-driven agriculture.

KEYWORDS: Convolutional Neural Network, potato disease detection, deep learning, precision agriculture, Early Blight, Late Blight, FastAPI, TensorFlow, image classification, smart farming

I. INTRODUCTION

Agriculture is one of the most vital sectors of the Indian economy, employing approximately 58% of the rural workforce and contributing significantly to national food security. Despite decades of research and policy support, farmers continue to face numerous challenges including erratic rainfall, soil degradation, rising input costs, and most critically, the rapid spread of crop diseases. Among these, leaf diseases affecting potato crops pose one of the most immediate and economically damaging threats.

Potatoes (*Solanum tuberosum*) rank among the world's most widely cultivated food crops, prized for their nutritional value, rapid growth cycle, and versatility in food processing. In India, potatoes are grown across approximately 2.2 million hectares with an annual production exceeding 50 million metric tonnes. However, potato cultivation is persistently threatened by Early Blight, caused by the fungal pathogen *Alternaria solani*, and Late Blight, caused by the oomycete *Phytophthora infestans*. Early Blight typically manifests as dark concentric lesions on older leaves, while Late Blight spreads rapidly in cool, humid conditions, capable of destroying entire fields within a matter of days if untreated [1].

Traditional disease identification relies heavily on visual inspection and expert consultation. Experienced farmers can sometimes recognise external symptoms, but early-stage infections present subtle signs that are difficult to detect with the unaided eye. Furthermore, in rural India, agricultural specialists and laboratory diagnostic facilities are often unavailable or prohibitively expensive, leading to delayed diagnosis and ineffective treatment. This forces farmers to apply pesticides indiscriminately, raising production costs, reducing profitability, and causing substantial environmental harm [3].

The rapid advancement of Artificial Intelligence (AI) and Machine Learning (ML), particularly Convolutional Neural Networks (CNNs), has created transformative opportunities in agricultural disease detection. CNNs automatically extract hierarchical spatial features from raw images, eliminating the need for manual feature engineering and achieving state-of-the-art accuracy on complex visual classification tasks [4]. When deployed as web-based applications, CNN models can provide farmers with instant, expert-level diagnostics directly at the field level using a smartphone.

This paper presents **Agri AI**, a web-based Potato Disease Classification System designed to bridge the gap between cutting-edge AI research and practical agricultural deployment. The system addresses four specific research gaps identified in the literature: (1) the absence of integrated systems combining disease detection with advisory services; (2) limited real-world accessibility of AI-based solutions for rural farmers; (3) lack of real-time environmental data integration for contextual risk assessment; and (4) minimal focus on user-friendly interfaces suitable for users with limited technical backgrounds.

The primary contributions of this work are as follows:

- A deployable CNN-based system achieving reliable classification of potato leaf diseases into three classes with confidence scoring.
- Integration of real-time weather data (temperature, humidity) to compute dynamic disease risk indicators.
- An AI-powered agricultural chatbot (AgriBot) for contextual farmer advisory.
- A fertilizer calculator module that provides crop-specific nutrient recommendations based on land area.
- A historical analytics dashboard for disease trend monitoring by agricultural extension workers.
- A fully tested, deployment-ready web application validated through unit, integration, system, performance, and user acceptance testing.

The remainder of this paper is organised as follows. Section II reviews related literature and identifies research gaps. Section III describes the system requirements and design. Section IV details the implementation. Section V presents testing methodology and results. Section VI discusses limitations and future enhancements. Section VII concludes the paper.

II. LITERATURE SURVEY

A. Traditional and Manual Approaches

For decades, plant disease detection relied entirely on farmer experience, visual inspection, and expert consultation. While effective in controlled laboratory settings, manual inspection is highly subjective, prone to inter-observer variability, and incapable of detecting early-stage infections where disease symptoms are subtle. Laboratory microscopic analysis and chemical testing, though more accurate, are time-consuming and economically impractical for small-scale farmers who require rapid field-level decisions [5].

B. Image Processing-Based Approaches

With advancements in digital imaging and computer vision, researchers developed semi-automated disease detection pipelines involving image capture, preprocessing (noise removal, contrast enhancement, colour space conversion), segmentation of diseased regions from healthy tissue, and classification based on hand-crafted features encoding colour, texture, and shape attributes. Algorithms such as k-means clustering and region-growing were widely used for leaf segmentation. Although these approaches offered partial automation, their performance was highly sensitive to environmental factors including illumination variability, image background, and camera quality, significantly limiting their real-world applicability [2].

C. Machine Learning Approaches

To address the brittleness of rule-based image processing, supervised machine learning algorithms including Support Vector Machines (SVM), k-Nearest Neighbours (KNN), Naïve Bayes, and Random Forests were applied to plant disease classification. These methods recognised statistical patterns in labelled datasets and generally outperformed traditional image processing in controlled conditions. However, their effectiveness remained tightly coupled to the quality of manually engineered features extracted during preprocessing. This dependence constrained their scalability and robustness when applied to diverse real-world field images collected under varying conditions [3].

D. Deep Learning and CNN-Based Approaches

Recent progress in deep learning, particularly CNNs, has fundamentally transformed the accuracy ceiling for image-based plant disease identification. CNNs automatically derive multi-level feature representations directly from raw pixel data through successive convolutional, pooling, and normalisation operations, removing the bottleneck of manual feature engineering entirely.

Mohanty et al. [1] demonstrated the effectiveness of CNNs for multi-class plant disease classification using the large-scale PlantVillage dataset, achieving accuracy exceeding 99% under controlled imaging conditions. Sladojevic et al. [2] introduced a deep neural network model tailored for leaf disease recognition on a custom dataset, achieving 96% accuracy and emphasising robustness to image quality variation. Ferentinos [3] conducted a comparative evaluation of multiple CNN architectures on the PlantVillage dataset and reported up to 99.5% accuracy, consistently outperforming classical machine learning baselines. More recent studies (2021–2023) employing transfer learning with pretrained architectures (VGG16, ResNet50, InceptionV3, EfficientNet) on mixed datasets report classification accuracy in the 95–98% range, while highlighting poor UI integration and deployment usability as persistent gaps.

E. Identified Research Gaps

Despite strong classification performance in research settings, a critical analysis of the surveyed literature reveals four persistent gaps:

- 1) **Integration gap:** Most existing systems offer only disease classification without agricultural advisory, weather-based risk context, or fertilizer guidance.
- 2) **Accessibility gap:** Solutions typically remain as research prototypes without accessible web or mobile deployment for rural farmers.
- 3) **Usability gap:** Interfaces are rarely designed for users with limited technical literacy, making adoption by smallholder farmers impractical.
- 4) **Environmental context gap:** Real-time environmental parameters such as temperature and humidity, which directly influence disease progression, are seldom integrated into prediction systems.

The proposed Agri AI system is specifically designed to address all four of these gaps within a single, cohesive platform.

III. SYSTEM DESIGN

A. Requirements Overview

The Agri AI system was designed to satisfy twelve functional requirements (FR1–FR12) and seven non-functional requirements (NFR1–NFR7) elicited through problem domain analysis and farmer-centric need assessment. Key functional requirements include image upload and validation (FR1–FR2), CNN-based disease classification with confidence scoring (FR3–FR5), PDF report generation (FR6–FR7), real-time weather fetching (FR8), fertilizer calculation (FR9), scan history storage and dashboard (FR10–FR11), and chatbot advisory (FR12). Critical non-functional requirements specify a prediction response time under 2 seconds (NFR1), 24/7 system availability (NFR2), secure image handling (NFR3), a responsive user interface (NFR4), and scalability for future expansion (NFR6).

B. Three-Tier Architecture

The system adopts a three-tier client–server architecture that enforces clean separation of concerns:

- **Presentation Layer:** A responsive web interface (HTML5, CSS3, JavaScript) enabling access on smartphones, tablets, and desktops without application installation.
- **Application Layer:** A Python FastAPI backend handling REST API routing, input validation, business logic, CNN model inference, and orchestration of external service calls.
- **Data Layer:** An SQLite database providing lightweight persistent storage of prediction results, disease metadata, and historical analytics data.

C. Functional Module Design

The system is decomposed into five loosely coupled functional modules, each encapsulating a specific agricultural concern:

1. Leaf Scanner Module handles image ingestion, format and size validation, preprocessing, and CNN model invocation. It enforces early input rejection to minimise unnecessary backend computation.

2. Advisory and Reporting Module generates structured diagnosis reports containing the predicted disease name, confidence percentage, causal pathogen description, and treatment recommendations. Reports are exportable as PDF

documents via a dedicated backend service endpoint.

3.Weather and Risk Module integrates the Open-Meteo API to fetch real-time temperature and relative humidity based on the user's GPS coordinates. A rule-based risk algorithm classifies blight risk as Low, Medium, or High based on epidemiological thresholds (e.g., humidity $\geq 85\%$ with temperature between 10–25°C triggers High Late Blight risk).

4.Fertilizer Calculator Module computes recommended quantities of Urea, Diammonium Phosphate (DAP), and Muri-ate of Potash (MOP) as a function of user-supplied land area(in acres), based on standard potato crop nutrient requirement ratios.

5.Dashboard and Analytics Module provides agricultural extension workers with paginated scan history, disease distribution donut charts, and temporal trend plots generated from accumulated database records.

D. Data Flow and Core Processing Pipeline

The core disease prediction data flow proceeds through six well-defined stages: image upload and validation → preprocessing (resize, normalise) → CNN inference → confidence threshold check → result persistence → JSON response to frontend. Results with confidence below the acceptance threshold are flagged as low-confidence before display, ensuring that unreliable predictions are clearly communicated to the user rather than silently accepted.

E. Database Schema

Two entities are maintained in the SQLite schema. ANALYSIS_RESULT (id PK, disease, confidence, timestamp, latitude, longitude) stores dynamic per-scan prediction records. DISEASE_INFO (disease_name PK, cause, remedy) stores static disease knowledge. A many-to-one classified_as association between the two entities eliminates data redundancy and allows advisory content to be updated independently of prediction records.

IV. IMPLEMENTATION

A. Technology Stack

Table I summarises the technology selections made for each layer of the system. All components are open-source, actively maintained, and selected for performance, ecosystem maturity, and ease of integration.

TABLE I TECHNOLOGY STACK SUMMARY

Component	Technology / Version
Operating System	Windows / Linux / <u>macOS</u>
Backend Framework	<u>FastAPI</u>
ML Framework	<u>TensorFlow 2.x</u> / <u>Keras</u>
Database	SQLite
Frontend	HTML5, CSS3, JavaScript
Weather API	<u>Open-Meteo</u>
Maps Integration	<u>Leaflet.js</u>
Charts & Analytics	Chart.js
Web Browser (Client)	Chrome / Firefox / Edge

B. CNN Model Architecture and Training

The CNN model was trained offline on the publicly available Plant Village dataset, filtered to three target classes: Early Blight, Late Blight, and Healthy. The architecture consists of four convolutional blocks, each comprising a Conv2D layer (with ReLU activation), Batch Normalization, MaxPooling2D, and a Dropout layer to mitigate overfitting. The convolutional blocks are followed by a GlobalAveragePooling2D layer, a fully connected Dense layer (512 units, ReLU), and a final three-class softmax output layer.

Input images are resized to 256×256×3 before being fed to the model. Data augmentation strategies applied during training include horizontal flip, random rotation ($\pm 15^\circ$), zoom ($\pm 10\%$), and brightness adjustment to improve generalisation across diverse field imaging conditions. The model was compiled with the Adam optimiser (learning rate = 10^{-4}), categorical cross-entropy loss, and accuracy as the evaluation

metric. The trained model weights are serialised in TensorFlow Saved Model format and loaded into server memory at application startup, eliminating per-request model loading overhead and ensuring consistently low inference latency.

C. AI Inference Pipeline

The complete six-stage AI inference pipeline processes each uploaded leaf image as follows: weather panel with blight risk indicator; an inline fertilizer calculator with real-time output; a paginated scan history table; and a floating AgriBot chat widget. The responsive layout is implemented using CSS flexbox and grid, ensuring consistent usability across devices ranging from low-end smartphones to desktop browsers.

D. Security and Configuration

Security measures implemented include file type and size validation before server-side processing, parameterised database queries to prevent SQL injection, restricted access to backend endpoints, and management of sensitive configuration (API keys, runtime settings) via environment variables. Uploaded images are processed in-memory and not persisted permanently unless required for analytics, minimising privacy exposure. These measures ensure compliance with the data privacy non-functional requirements defined in the SRS.

V. TESTING AND RESULTS

A. Testing Strategy

$$I_{\text{raw}} \longrightarrow I_v \longrightarrow I_p \longrightarrow \vec{p} \longrightarrow c \longrightarrow R$$

where I_{raw} is the raw uploaded image, I_v is the validated image, I_p is the preprocessed tensor, $\vec{p} \in \mathbb{R}^3$ is the class probability vector produced by the soft max layer, c is the predicted class index, and R is the structured result object containing the disease label, confidence score ($\max(\vec{p}) \times 100\%$), and advisory metadata retrieved from the DISEASE_INFO table.

B. Backend REST API Design

The Fast API backend follows RESTful design principles with stateless communication and JSON payloads. Every incoming request passes through a five-stage processing chain: Fast API endpoint $\rightarrow$ input validation $\rightarrow$ business logic $\rightarrow$

model inference / DB access $\rightarrow$ JSON response. The five primary API endpoints are:

- POST /predict — accepts a multipart/form-data image; returns predicted disease class, confidence score, cause, and remedy.
- GET /weather — accepts latitude/longitude query parameters; returns temperature, humidity, and computed blight risk level.
- POST /fertilizer — accepts land area (acres); returns recommended Urea, DAP, and MOP quantities in kilograms.
- POST /chat — accepts a natural-language query string; returns an advisory response from the Agri Bot service.
- GET /dashboard — returns aggregated historical scan statistics and disease distribution data for the analytics view.

C. Frontend Architecture

The frontend is implemented as a single-page, event-driven web application. All backend communication is performed via asynchronous fetch() calls, preventing page reloads and maintaining a fluid user experience. Core UI components include: an image drag-and-drop upload zone with client-side format validation; a prediction result card displaying disease name, confidence percentage, and colour-coded severity.

A comprehensive, layered testing strategy was adopted to validate the Agri AI system at multiple levels of abstraction. Testing was conducted in five phases: Unit Testing, Integration Testing, System Testing, Performance Testing, and User Acceptance Testing (UAT). Test cases were derived directly from the functional and non-functional requirements specified in the SRS, ensuring full requirement traceability.

D. Unit Testing

Unit tests were developed and executed for all backend modules in isolation, including the image upload and validation module, the image preprocessing function, the disease prediction logic, the fertilizer calculation service, and the weather risk computation algorithm. All unit test cases passed. Representative unit test results are summarised in Table II.

TABLE II UNIT TEST RESULTS (SELECTED)

ID	Module	Input	Status
UT01	Image Upload	Valid JPG	Pass
UT02	Image Upload	Invalid file type	Pass
UT03	Preprocessing	Large image	Pass
UT04	Prediction	Clear leaf image	Pass
UT05	Fertilizer Calc	Land = 2 acres	Pass

E. Integration Testing

Integration tests validated the four critical inter-module communication paths: (1) frontend image upload to Fast API backend, (2) backend to CNN model prediction invocation,

(3) backend to SQLite database persistence, and (4) backend to Open-Meteo weather API retrieval. All integration test scenarios passed, confirming correct data flow and interoperability between all system layers. Table III summarises the integration test outcomes.

TABLE III

INTEGRATION TEST RESULTS		TEST
ID	Integrated Modules	Status
IT01	Frontend – Backend	Pass
IT02	Backend – CNN Model	Pass
IT03	Backend – SQLite Database	Pass
IT04	Backend – Weather API	Pass

F. System and Performance Testing

System testing validated the complete end-to-end disease diagnosis workflow, including handling of invalid image uploads, dashboard data retrieval, fertilizer calculation, and chat-bot interaction. All system test scenarios passed. Performance testing evaluated prediction response time and system stability under concurrent user load. As shown in Table IV, the observed prediction latency of 1.4 seconds satisfies the SRS NFR1 requirement of under 2 seconds, and the system remained stable under a simulated concurrent load of 20 simultaneous users.

TABLE IV PERFORMANCE TEST RESULTS

ID	Parameter	Target	Observed	Status
PT01	Prediction latency	<2 s	1.4 s	Pass
PT02	Concurrent users	20	Stable	Pass

G. User Acceptance Testing

The system was evaluated by a pilot group of end-users using real potato leaf images captured under authentic field conditions. Participants performed three key tasks: uploading a leaf image and viewing the diagnosis result, interpreting the weather risk indicator, and using the fertilizer calculator. As shown in Table V, all three tasks received

positive feedback, validating both the functional correctness and usability of the system.

TABLE V USER ACCEPTANCE TEST RESULTS

ID	Task	Feedback
UAT01	Upload image and view result	Positive
UAT02	Interpret weather risk indicator	Positive
UAT03	Use fertilizer calculator	Positive

F. Comparative Analysis

Table VI benchmarks Agri AI against representative exist-ing systems across six key evaluation dimensions. Agri AI demonstrates clear advantages in automation, response time, decision support integration, and accessibility compared to prior approaches.

TABLE VI AGRI AI VS. EXISTING SYSTEMS

Parameter	Existing Systems	Agri AI
Automation	Partial / manual	Fully automated
Accuracy	Medium to High	High
Response Time	Slow	Real-time (<2 s)
Scalability	Limited	High (modular)
Decision Support	Not available	Fully integrated
Accessibility	Expert dependent	Web-based, any device

G. Boundary and Negative Testing

Boundary value testing verified system behaviour at the edges of valid input ranges, including maximum image size limits, minimum land area values for fertilizer calculation, and extreme weather parameter values. Negative testing confirmed that the system responds with clear, descriptive error messages and does not crash or produce undefined behaviour when presented with invalid inputs such as non-image files, corrupted images, unsupported MIME types, or empty request payloads. These tests validated the robustness of the input validation layer and the graceful degradation behaviour of the system under adverse conditions.

VI. DISCUSSION AND FUTURE ENHANCEMENTS

A. System Limitations

Despite its demonstrated capabilities, the current version of Agri AI has several acknowledged limitations. Prediction accuracy is influenced by image quality and lighting conditions; blurry, overexposed, or shadowed images may produce lower-confidence or incorrect classifications. The system currently supports only two potato leaf diseases (Early Blight and Late Blight) and requires continuous internet connectivity for weather monitoring and chatbot features. The SQLite database, while adequate for prototype-scale usage, may require migration to a more robust database engine (e.g., PostgreSQL) for high-concurrency production deployment.

B. Planned Enhancements

A four-phase development roadmap is planned to evolve Agri AI from a focused potato disease classifier into a comprehensive multi-crop smart agriculture platform:

Phase 1 — Multi-crop Support: Extending the CNN model to cover additional crops including tomato, maize, rice, and cotton by retraining with expanded PlantVillage subsets and applying transfer learning with EfficientNetB3 or ResNet50V2 as pretrained feature extractors.

Phase 2 — Mobile Application: Developing a standalone Progressive Web App (PWA) or cross-platform mobile application (React Native / Flutter) to enable direct in-field image capture via device camera, offline disease detection using a TensorFlow Lite quantised model, and push notifications for disease risk alerts.

future enhancements including multi-crop support, mobile deployment, IoT integration, and predictive disease forecasting. By making expert-level agricultural knowledge accessible through a simple web interface at no cost to

the farmer, Agri AI contributes to reducing crop losses, optimising pesticide usage, and improving the livelihoods of smallholder farmers. The system aligns with India's national digital agriculture initiatives and demonstrates that responsible AI deployment in agriculture can drive inclusive economic growth and long-term ecological sustainability.

REFERENCES

1. S. P. Mohanty, D. P. Hughes, and M. Salathe', "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, Sep. 2016.
2. S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational Intelligence and Neuroscience*, vol. 2016, p. 3289801, 2016.
3. K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, Feb. 2018.
4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
5. R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 8th ed. New York, NY, USA: McGraw-Hill, 2014.
6. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
7. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.
8. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
9. S. Ramirez, "FastAPI: Modern, fast (high-performance), web framework for building APIs with Python," [Online]. Available: <https://fastapi.tiangolo.com/>. [Accessed: Dec. 2024].
10. M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," [Online]. Available: <https://www.tensorflow.org/>. [Accessed: Dec. 2024].
11. Open-Meteo, "Open-Meteo free weather API," [Online]. Available: <https://open-meteo.com/>. [Accessed: Dec. 2024].
12. D. P. Hughes and M. Salathe', "An open access repository of images on plant health to enable the development of mobile disease diagnostics", 2015.



International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 9.971