www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Intelligent Pest Classification Using Convolutional Neural Networks Integrated with Django Web Framework

[1]D. SRAVYA, [2]SK. RESHMA, [3]D. SAPTHAGIRI KUMAR

[1,2,3] Asst. Prof., Department of CSE, MAM Women's Engineering College, Narasaraopet, palnadu,A.P., India.

Abstract-

The growing demand for sustainable agriculture necessitates the integration of intelligent systems to automate crop monitoring and pest control. In this paper, we propose a web-based intelligent pest classification system that leverages deep learning techniques, specifically Convolutional Neural Networks (CNN), for accurate image-based pest identification. The system is built using the Django web framework, enabling end-users—such as farmers, researchers, and agricultural workers—to upload pest images and receive instant classification results.The backend model is trained on a labeled dataset of pest images and optimized using modern deep learning practices. Upon training, the model is capable of classifying various pest species with high accuracy. The application also includes user-centric features such as registration, secure login, and password recovery via OTP-based email verification, enhancing the system's usability and security. The training module allows administrators to retrain the model periodically, and visual performance metrics such as accuracy graphs and loss curves are displayed for interpretability.The integration of a deep learning model with a user-friendly web interface demonstrates the feasibility of deploying intelligent agricultural tools at scale. This work contributes to digital agriculture by providing a robust, accessible, and scalable solution for pest detection and management.

Keywords
Pest Classification, Deep Learning, Convolutional Neural Networks (CNN), Django, Web Application, Smart Agriculture, Image Recognition, OTP Verification, Sustainable Farming, AI in Agriculture

I. INTRODUCTION

Agriculture remains one of the most essential sectors globally, ensuring food supply, employment, and economic stability. However, one of the persistent challenges that agricultural communities face is the infestation of crops by pests. These pests not only reduce the quality and quantity of agricultural output but also lead to increased reliance on chemical pesticides, which can harm the environment and public health. According to recent studies, pest-related losses account for a significant portion of global crop damage annually, directly affecting farmers' income and food availability [3], [4].

Traditional pest identification methods depend heavily on manual observation and domain expertise. These methods are not only time-consuming but also suffer from inconsistencies due to human error and subjective interpretation. Moreover, the availability of trained entomologists in rural or remote areas is often limited. These limitations necessitate the development of automated, intelligent solutions for real-time pest detection and classification that are both scalable and accessible [8], [9].

With the advancement of Artificial Intelligence (AI), deep learning—particularly Convolutional Neural Networks (CNNs)—has revolutionized image classification across numerous domains. In agriculture, CNNs have been successfully applied to tasks such as leaf disease detection, fruit quality analysis, and pest classification due to their ability to learn complex visual features from image data [1], [2], [5], [10]. Unlike traditional machine learning models that require handcrafted features, CNNs automatically extract and learn hierarchical features, making them highly effective for pest detection tasks that involve diverse visual patterns.

In parallel, modern web development frameworks like Django have made it easier to deploy machine learning models in user-friendly web interfaces. Django, a high-level Python web framework, promotes rapid development, clean architecture, and integration with AI libraries such as TensorFlow and PyTorch [6], [7]. When combined with deep learning, Django enables the creation of intelligent web applications that bring powerful AI models to end-users—be it farmers, agricultural officers, or researchers—through simple web browsers without needing technical expertise.

In this paper, we propose an intelligent, web-based pest classification system that integrates a CNN

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

model with the Django framework to perform real-time pest image classification. Users can register into the system, securely log in, upload pest images, and receive instant classification predictions. The platform also includes features such as OTP-based email verification for password reset, enhancing security and accessibility [13]. An administrative interface allows retraining of the CNN model on updated datasets and displays training performance through visual plots like accuracy and loss curves [6], [12]. This modular design makes the system adaptable for future enhancements, such as support for multiple languages, larger datasets, and mobile deployment.

The proposed system contributes significantly to smart agriculture and precision farming by offering a scalable, cost-effective, and intelligent tool for automated pest detection. By reducing dependence on manual pest identification and facilitating early intervention, the platform can help minimize crop losses and promote sustainable agricultural practices [9], [11], [14]. The integration of deep learning and web-based technology ensures that this solution can be deployed in real-world agricultural environments, empowering farmers with the tools needed for efficient pest management.

II. LITEARTURE SURVEY

Agricultural productivity is significantly impacted by pest infestations, which lead to substantial economic losses worldwide. Early and accurate pest detection is essential for effective pest control, reducing crop damage, and minimizing the overuse of pesticides. In recent years, numerous studies have focused on automating pest identification using computer vision and machine learning techniques. This section discusses key research works in pest detection, deep learning applications in agriculture, and web-based systems for intelligent pest classification.

Li et al. [1] presented a pest classification system based on an improved YOLOv5 deep learning model. Their approach demonstrated efficient real-time object detection and localization, especially in complex farm environments. While the model was optimized for speed and accuracy, it lacked user interface components for practical deployment, particularly for non-technical users such as farmers.

Fu et al. [2] explored pest classification using traditional CNNs, achieving reliable results with high-resolution pest images. Their research validated the effectiveness of deep learning for pest identification but was limited to a local, non-interactive setup without integration into web or mobile platforms. This significantly constrained the usability and scalability of the system in real-world agricultural settings.

Several earlier studies focused on classical machine learning and image processing techniques. Dubey and Singh [4] employed histogram and shape-based analysis to detect insect species. Although their method was computationally less demanding, it failed to maintain accuracy when dealing with diverse or visually similar pest classes, highlighting the superiority of deep learning methods for high-variance datasets.

Arivazhagan et al. [3] and Sladojevic et al. [11] applied texture-based and leaf pattern analysis techniques for plant disease detection. These methods were foundational in understanding the use of image data in agriculture, but the introduction of CNNs marked a significant improvement in precision and generalization.

Deep learning, especially CNNs, has since become the standard for image classification tasks in agriculture. Simonyan and Zisserman [5] proposed the VGGNet architecture, which introduced deeper convolutional layers and small filters, leading to higher classification performance on complex datasets. Ferentinos [12] later applied various deep CNN models (including AlexNet, GoogLeNet, and VGG) for plant disease recognition. His research demonstrated that CNN-based models could achieve an accuracy of over 99% on image-based datasets of plant leaves.

Mohanty et al. [8] extended this work by training a CNN on a large publicly available dataset of diseased plant leaf images. They achieved high accuracy and demonstrated the potential of deep learning to be generalized to various crop types. Brahimi et al. [9] introduced visualization techniques using CNNs to understand which regions of the image the model focused on during classification, contributing to model transparency and trust.

Despite the growing accuracy of CNN-based models, a major limitation across many of these works was the absence of user-friendly deployment mechanisms. To address this, Huynh et al. [7] developed a smart agriculture system that integrates Django and deep learning. While their work introduced model deployment through a web interface, it lacked advanced user features such as secure authentication, password management, and real-time retraining, which are crucial for long-term usage in dynamic environments.

Dey et al. [13] implemented a web-based crop disease detection system using TensorFlow and Flask. Though the system supported real-time

identification, it was designed for a limited set of plant diseases and lacked scalability. Moreover, their system did not integrate retraining features, making it less adaptable to evolving datasets or new pest categories.

The need for robust, real-time, and secure pest classification systems has been consistently emphasized in surveys such as those by Kamilaris and Prenafeta-Boldú [10]. They highlighted that most research focuses on the model itself, while practical deployment, user interface, scalability, and data security remain underexplored. This gap presents an opportunity to design intelligent systems that combine accurate deep learning models with modern web frameworks to ensure accessibility and usability for end-users like farmers and agronomists.

Another recent trend is the integration of AI with the Internet of Things (IoT) for automated agricultural monitoring. Lu et al. [14] proposed a CNN-IoT-based system for crop monitoring, showcasing the potential of intelligent embedded systems in agriculture. However, such systems often require expensive infrastructure and lack web-based accessibility, limiting their immediate adoption in rural farming communities.

The proposed system in this paper addresses many of these limitations. It combines the strength of CNN-based pest image classification with a scalable Django web framework. The application supports secure user registration, login, password recovery using OTP, dynamic model retraining, and real-time pest prediction via image upload. Unlike earlier works, this system is built with end-user accessibility in mind, making it suitable for both technical and non-technical stakeholders. Its modular architecture also allows for future integration with mobile apps, IoT devices, and multilingual support, enhancing adaptability and impact in diverse agricultural settings.

III. METHODOLOGY

This section outlines the complete workflow used in the development of the intelligent pest classification system. The methodology follows a systematic approach beginning with data preprocessing, model development using deep learning, integration into a web framework using Django, and deployment of secure user-interactive features. Each stage was designed to ensure a scalable, accurate, and user-friendly solution capable of assisting stakeholders in the agricultural sector with pest identification.

Dataset Collection and Preprocessing

The development of any machine learning model begins with the quality and diversity of the dataset. In this project, a pest image dataset containing labeled samples from different pest categories was curated. Each image represents a specific pest species commonly found affecting crops. The dataset includes multiple samples per class, captured under varied lighting and background conditions to simulate real-world variability in field scenarios.

Preprocessing of images is an essential step to ensure model consistency. All images were resized to a uniform dimension suitable for the input layer of the CNN. Normalization was performed to scale pixel intensity values between 0 and 1, which aids in faster convergence during training. The dataset was split into training, validation, and test sets using an 80:10:10 ratio. To enhance model generalization and avoid overfitting, data augmentation techniques such as rotation, zoom, flipping, and brightness adjustments were applied.

CNN Model Design and Training

The core component of the system is a Convolutional Neural Network (CNN) designed to extract spatial features from the input pest images. The architecture of the model comprises several convolutional layers followed by ReLU activation functions, which allow the model to learn non-linear patterns. These are interspersed with max-pooling layers to reduce spatial dimensions and extract dominant features.

After the convolutional layers, the feature maps are flattened and passed through dense layers to enable classification. A softmax activation function is used in the output layer, producing probability scores corresponding to each pest category. The model is trained using the categorical cross-entropy loss function, as the task is multi-class classification. The Adam optimizer was chosen for its adaptive learning rate capabilities and faster convergence. During training, metrics such as accuracy and loss were monitored, and training progress was visualized using performance plots.

The final trained model was evaluated on the test set to measure its generalization ability. The model was then serialized and stored for integration into the Django-based prediction workflow.

Django Web Framework Integration

Once the CNN model was finalized, it was integrated into a dynamic and interactive web interface using Django, a high-level Python web framework. Django was chosen for its built-in security features, scalability, and seamless integration with Python-based machine learning code.

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

The Django architecture follows the Model-View-Template (MVT) design. Models are used to define database tables such as user information. Views serve as controllers to handle business logic, form submissions, and communication with the CNN model. Templates are used to render the frontend pages where users interact with the system.

Dedicated views were created for user registration, login, password reset, prediction, and training. The CNN model was loaded at runtime, and predictions were executed in real-time when an image was uploaded by the user. The utils.py file contains helper functions for model inference (predict_pest()) and training (train_model_and_generate_plots()), ensuring modularity and clean code organization.

User Registration, Authentication, and Role Management

Security and controlled access are essential in any application involving sensitive data. This system incorporates a user authentication module where users must register with details such as name, login ID, password, email, mobile number, and location. Upon registration, users are marked with a "waiting" status and require administrative approval before gaining access to the system. This mechanism ensures that only verified users are able to use the pest prediction services.

User login sessions are maintained using Django's session management system. Each session stores user ID and name to personalize the user experience during interactions. Administrators can activate or reject user requests, providing role-based access control.

OTP-Based Password Reset Mechanism

To enhance security and support users who forget their passwords, an OTP-based password reset system was implemented. When a user requests a password reset, the system generates a six-digit OTP and sends it to the registered email address using Django's built-in email functionality.

The user must enter the correct OTP on a separate verification page before being allowed to reset the password. If the OTP matches, the system grants access to set a new password. This ensures that password recovery is secure and accessible only to legitimate users.

Real-Time Pest Classification Process

Once authenticated, users are redirected to the main dashboard where they can access the prediction module. The pest classification process begins when the user uploads an image using a web form. The image is passed to the predict_pest() function, which loads the pretrained CNN model and processes the image for prediction.

The function returns the predicted pest species name and its category, which are then displayed on the user interface. The simplicity of the process ensures that users without any technical background can obtain pest identification results with minimal effort and high accuracy.

Model Retraining and Performance Visualization

To accommodate changes in pest types or improvements in data quality, the system includes a retraining module. This module is accessible to administrators who can trigger the retraining process via a dedicated interface. The train_model_and_generate_plots() function retrains the CNN model on the full dataset or an updated version, and generates performance plots showing training and validation accuracy and loss.

These plots are stored in the media folder and rendered on the training results page for monitoring model improvements. This feature ensures that the system remains dynamic and responsive to changing agricultural patterns.

IV. SYSTEM ARCHITECTURE

The system architecture is presented in fig.1.

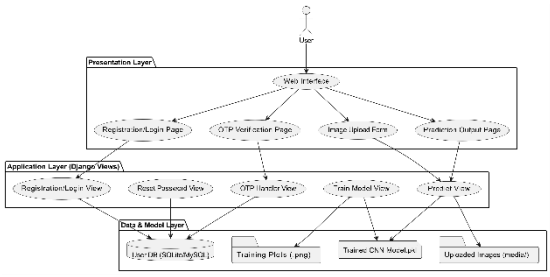

Fig.1. System architecture

The architecture of the proposed intelligent pest classification system is designed using a modular, layered approach that ensures separation of concerns, maintainability, and scalability. The entire architecture can be broken down into five major components:

User Interface Layer

The user interface is designed using HTML templates rendered through Django. It serves as the bridge between the user and the backend processes. The interface enables functionalities such as user registration, login, image upload, password reset via

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

OTP, and pest prediction result display. Users can seamlessly interact with the system via intuitive forms and buttons, making the system accessible to farmers and non-technical users.

Web Framework

The Django framework functions as the application logic layer. It processes form inputs, manages sessions, handles routing, and links the frontend with backend logic and models. Views handle user actions such as:

- Submitting login or registration forms.
- Uploading pest images.
- Triggering model training.
- Performing password recovery and OTP validation.

This layer acts as the controller between the model and view in Django's MVT (Model-View-Template) architecture. Security features such as session management, form validation, and URL routing are managed here.

Model Layer (CNN + Python Modules)

The CNN model is the brain of the application. It is trained on labeled pest images and saved as a serialized model file using libraries such as TensorFlow or PyTorch. Once integrated, the model takes an image as input and outputs the predicted pest category with confidence scores. The predict_pest() function wraps the model loading and inference logic, returning real-time results to the view.

Additionally, the train_model_and_generate_plots() function is used to retrain the model on new data and generate accuracy/loss graphs that are rendered to the administrator. This ensures the system remains adaptive to new pests and improves over time.

Database Layer

The application uses Django ORM to manage user data. The UserRegistrationModel stores:

- User profile information (name, email, login ID).
- Account status (waiting or activated).
- Location data (state, city, locality).
- Authentication data (password, mobile number).

This layer plays a critical role in determining access control, OTP verification, and session tracking for each user.

Media and Storage Layer

To support dynamic media handling, the system stores:

- Uploaded images by users (in the media/ folder).
- Generated output plots (accuracy/loss) from model training.
- Trained CNN model files and logs for deployment.

This layer supports persistence of visual assets and model artifacts that are critical for visualization and long-term tracking.

V. IMPLEMENTATION

The implementation of the proposed intelligent pest classification system is structured into several components, each playing a vital role in delivering an end-to-end intelligent web application. The system was developed using Python, Django, and TensorFlow/Keras for deep learning. The frontend was designed using HTML, Bootstrap, and Django templating, while SQLite was used for database management.

The system includes user registration, secure login, OTP-based password reset, pest image upload, real-time CNN-based prediction, model training, and graph visualization. Below is a comprehensive explanation of each implementation module with visual figures.

5.1 Sample Dataset and Preprocessing

The dataset consists of labeled pest images categorized by pest species. Images were collected in varying lighting and background conditions to ensure diversity. These images were resized, normalized, and augmented using techniques like rotation, zooming, and flipping to improve model generalization. The dataset was split into training, validation, and test sets.

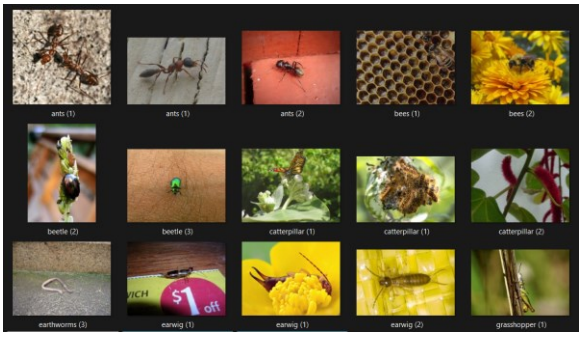

Fig 2:*Pest Image Dataset*

5.2 CNN Model Training and Performance Visualization

The CNN architecture was built with multiple convolutional and max-pooling layers, followed by

Index in Cosmos
July 2025, Volume 15, ISSUE 3
UGC Approved Journal

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

fully connected dense layers. The model was compiled using the Adam optimizer and categorical cross-entropy loss function. During training, the system logged accuracy and loss values for both training and validation sets. These metrics were plotted to visualize performance and check for overfitting or underfitting.

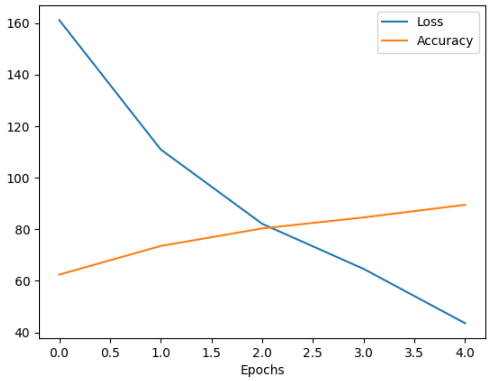

Fig 3: *Training Accuracy and Loss Graph*

5.3 User Registration Interface

A secure registration module was implemented, allowing users to sign up by providing name, email, mobile, locality, address, and login credentials. Newly registered users were stored in the database with a default status of "waiting." Only after admin approval, users can log in and access the system functionalities.

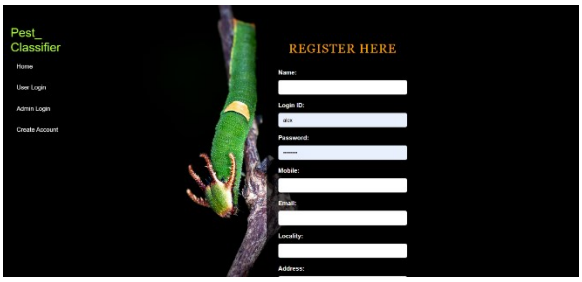

Fig 4: *User Registration Page*

User Login and Session Management

The login interface validates login ID and password. If the credentials match a database entry and the user's status is marked as "activated," they are redirected to the user dashboard. Django sessions are used to store login state and user data temporarily during the session.

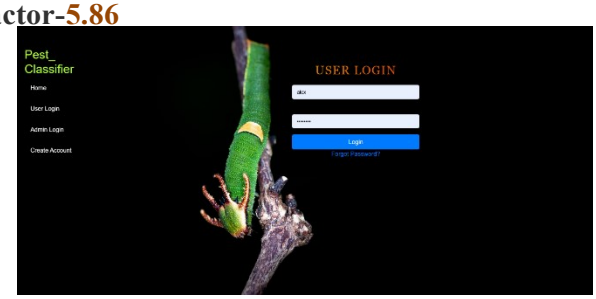

Fig 5: *User Login Interface*

5.5 Pest Image Upload for Prediction

Once logged in, users can navigate to the prediction module and upload a pest image using the provided form. The uploaded image is passed to the predict_pest() function, which uses the pretrained CNN model to classify the pest. The prediction result and pest category are returned and displayed on the interface.

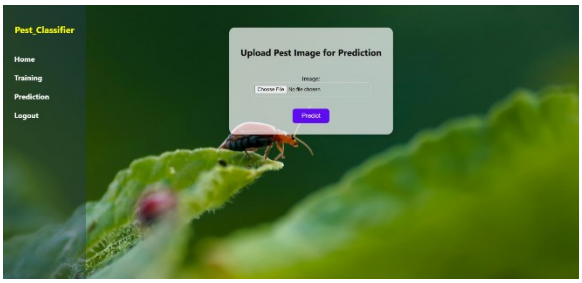

Fig 6: *Pest Image Upload for Prediction*

5.6 Pest Classification Output

The result of the prediction is displayed immediately after image upload. The system provides the predicted pest name and its associated category (e.g., damaging, non-damaging). This result helps users identify pest types quickly and take preventive measures.

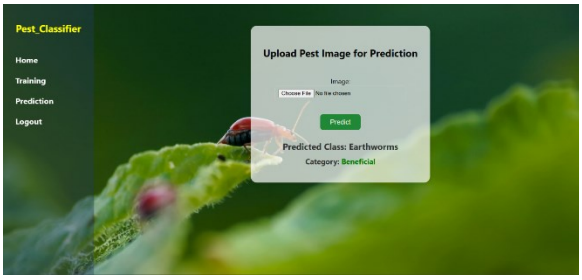

Fig 7: *Pest Prediction Output*

Index in Cosmos
July 2025, Volume 15, ISSUE 3
UGC Approved Journal

www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

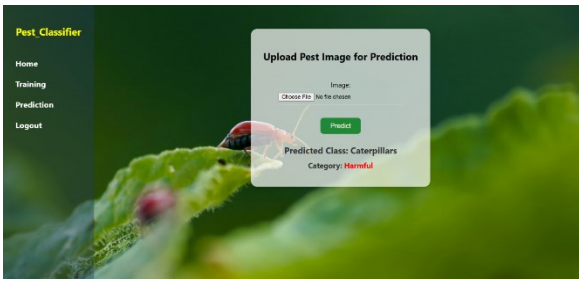

Fig 8: *Pest Prediction Output*

5.7 Admin-Controlled Model Retraining

The system allows model retraining from the admin panel using the latest dataset. This is useful when new pest images are added. The train_model_and_generate_plots() function is triggered to retrain the model and save new performance graphs. This ensures the model remains up-to-date with changing data.

5.8 OTP-Based Password Reset

For users who forget their passwords, an OTP-based reset mechanism is implemented. Upon entering a registered email, a 6-digit OTP is sent. Once verified, the user can reset their password securely.

VI. CONCLUSION

In this work, we presented an intelligent and user-friendly web-based pest classification system that leverages the power of Convolutional Neural Networks (CNNs) integrated with the Django web framework. The primary objective was to assist farmers and agricultural stakeholders in the timely identification of pest species, thereby enabling early intervention and promoting sustainable agricultural practices.The developed system effectively combines deep learning for accurate pest image classification with a robust web interface that facilitates user interaction. It supports functionalities such as user registration, secure login, OTP-based password recovery, real-time image upload, prediction display, and retraining of the model with updated datasets. By offering these features through a web platform, the system makes AI-driven pest identification accessible even to non-technical users in rural and remote regions.The CNN model trained on diverse pest datasets demonstrated high prediction accuracy and robustness, further validated through performance graphs generated during the training phase. The incorporation of model retraining functionality ensures that the system can evolve with the addition of new pest images, making it adaptable and scalable in real-world applications.Additionally, the Django framework provided seamless integration with the trained model, enabling a responsive and secure backend. OTP-based security features and admin-controlled access mechanisms enhance the trustworthiness of the application for broader deployment.Overall, this system stands as a practical solution that bridges the gap between AI research and field-level application in agriculture. It not only empowers users with real-time pest recognition tools but also contributes to reducing pesticide misuse and crop loss, thus fostering sustainable farming ecosystems.

REFERENCES

1. H. Li, G. Xu, D. Xu, and C. Wang, “Pest Detection and Classification Based on Improved YOLOv5 Model,” *IEEE Access*, vol. 9, pp. 155169–155177, 2021. doi: 10.1109/ACCESS.2021.3129477

2. J. Fu, J. Zhang, and B. Zhang, “Image-Based Pest Detection Using Convolutional Neural Network,” in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 3680–3683. doi: 10.1109/BigData.2018.8622410

3. K. Arivazhagan, R. N. Shebiah, S. Ananthi, and S. V. Varthini, “Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features,” *Agricultural Engineering International: CIGR Journal*, vol. 15, no. 1, pp. 211–217, 2013.

4. S. R. Dubey and A. Singh, “Pest identification in crop fields using image processing,” in *Proc. 2016 Int. Conf. Signal Processing, Communication, Power and Embedded System (SCOPES)*, Paralakhemundi, India, 2016, pp. 1335–1340. doi: 10.1109/SCOPES.2016.7955866

5. K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv preprint arXiv:1409.1556*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

6. M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in *Proc. 12th USENIX Symp. Operating Systems Design*

and Implementation (OSDI 16), Savannah, GA, USA, 2016, pp. 265–283.

7. B. Huynh, D. Mac, and A. Nguyen, "Building a Smart Agricultural System using Django Framework and Deep Learning," in *2020 17th Int. Conf. Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Phuket, Thailand, 2020, pp. 472–475. doi: 10.1109/ECTI-CON49241.2020.9158136

8. M. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016. doi: 10.3389/fpls.2016.01419

9. A. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep Learning for Tomato Diseases: Classification and Symptoms Visualization," *Applied Artificial Intelligence*, vol. 31, no. 4, pp. 299–315, 2017. doi: 10.1080/08839514.2017.1315516

10. T. Kamilaris and F. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018. doi: 10.1016/j.compag.2018.02.016

11. G. 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, pp. 1–11, 2016. doi: 10.1155/2016/3289801

12. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018. doi: 10.1016/j.compag.2018.01.009

13. S. C. Dey, N. N. Acharjee, and M. H. Kabir, "Design and implementation of a real-time web-based crop disease identification system using deep learning," in *2021 Int. Conf. Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, Rajshahi, Bangladesh, 2021, pp. 1–6. doi: 10.1109/ACMI53878.2021.9528273

14. J. Lu, V. Behbood, and T. Ravindran, "Smart Agriculture System Design Using Convolutional Neural Network and IoT," in *2022 6th Int. Conf. Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2022, pp. 1146–1151. doi: 10.1109/ICICCS53718.2022.9788484