Advanced Pest Identification: An Efficient Deep Learning Approach Using VGG Networks

A Project Report submitted in the partial fulfillment of
the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

D. Chandu Venkateswara Guptha (21471A05E7)

J. Sai (21471A05F9)

K. Rajesh (21471A05G2)

Under the esteemed guidance of
Dr. K. Lakshminadh M.Tech., Ph.D.
Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under Tyre -1 NIRF rank in the band of 201-300 and an ISO 9001:2015 Certified
Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "Advanced Pest Identification: An Efficient Deep Learning Approach Using VGG Networks" is a Bonafide work done by the team D. Chandu (21471A05E7), J. Sai (21471A05F9), K. Rajesh (21471A05G2) in partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

Dr. K. Lakshminadh, M.Tech., Ph.D.,

Professor

PROJECT-COORDINATOR

Dr. M. Sireesha, M.Tech., Ph.D.,

Assoc. Professor

HEAD OF THE DEPARTMENT

Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,

Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled " Advanced Pest Identification: An Efficient Deep Learning Approach Using VGG Networks" is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualification except as specified.

D. Chandu (21471A05E7)

J. Sai (21471A05F9)

K. Rajesh (21471A05G2)

ACKNOWLEDGEMENT

We wish to express my thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman Sri **M. V. Koteswara Rao, B.Sc.,** who took keen interest in us in every effort throughout this course. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of CSE department and also to our guide **Dr. K. Lakshminadh, M.Tech., Ph.D.,** of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dr. Sireesha Moturi, B.Tech, M.Tech., Ph.D.,** Associate Professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff in the department for their cooperation and encouragement during our B.Tech degree. We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who were involved in giving valuable suggestions clarified out doubts, which really helped us in successfully completing our project.

By

D. Chandu (21471A05E7)

J. Sai (21471A05F9)

K. Rajesh (21471A05G2)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research.

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills.

M3: Imbibe lifelong learning skills, entrepreneurial skills, and ethical values in students for addressing societal problems.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the industry.

M3: Inculcate teamwork and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the program are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry, and society.

PEO3: Work with ethical and moral values in multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in the software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes (PO'S)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and teamwork**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

Project Course Outcomes (CO'S)

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature.

CO421.4: Design and Modularize the project.

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Focuses on data preprocessing, machine learning, and algorithm implementation.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Utilizes natural language processing, sentiment analysis, and feature extraction techniques.	PO2, PO3
CC421.2, C2204.2, C22L3.3	Covers the integration of machine learning and deep learning models for solution development.	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Emphasizes feature engineering and optimization of algorithms.	PO1, PO5
CC421.4, C2204.4, C22L3.2	Highlights research methods and model evaluation techniques, ensuring effective communication.	PO10
CC421.5, C2204.2, C22L3.3	Explores real-world applications, project management, and ethical considerations.	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Integrates interdisciplinary knowledge for addressing societal and environmental impacts.	PO4, PO7
C32SC4.3	Applies advanced methodologies and tools for model deployment and evaluation.	PO5, PO6

s

ABSTRACT

Accurate pest identification is crucial for both effective pest management and crop protection. Pests must be found early in order to minimise damage and guarantee crop security. Conventional techniques typically entail visual examination and professional involvement, which might be time-consuming and susceptible to errors by humans. On the other hand, deep learning-powered high-performance systems can now more accurately identify pests thanks to developments in computer vision.

In this work, we employed the Keras-based deep learning models VGG16 and VGG19 to construct a passive pest detection system. We greatly improved the efficacy of these models in identifying pest species by using strategies such data augmentation, model optimisation, and modification of validated models. The VGG16 model produced an amazing accuracy rate of 99.8% and VGG19 model produced an accuracy of 96.8 % in our testing.

INDEX

LIST OF FIGURES

1. INTRODUCTION

The world economy depends heavily on agriculture, but pests pose a significant challenge to crop production and food security. Pests are responsible for the largest crop losses worldwide, hampering agricultural growth, especially in light of increasing global demand for food. The presence of pests not only reduces the quantity of agricultural produce but also affects its quality, leading to economic losses for farmers and a decline in market value.

Traditional pest control methods, including chemical pesticides, have been widely used, but their long-term sustainability is increasingly being questioned due to environmental concerns, pesticide resistance, and harmful effects on human health. In addition to environmental risks, the frequent use of pesticides has led to increased production costs for farmers, as stronger and more expensive chemicals are required over time. Expert techniques that rely on manual identification and flow visualization are typically laborious, error-prone, and time-consuming.

The complexity arises due to the vast diversity in pest species, which exhibit different forms, sizes, and behavioural patterns, making it challenging for even experienced entomologists to achieve precise identification in real-world agricultural settings. Some pests also evolve to adapt to their environments, making traditional identification methods even less effective. This necessitates the development of automated and efficient pest detection methods to support the agricultural industry.

Advances in computer vision and artificial intelligence (AI) have provided promising solutions for pest identification in recent years. Deep learning, a subset of machine learning, has emerged as a powerful approach for image-based classification tasks across various domains, including healthcare diagnostics, autonomous vehicles, and industrial quality control. The ability of AI-based systems to process large datasets and identify patterns that are difficult for humans to detect has opened new possibilities in precision agriculture.

In particular, Convolutional Neural Networks (CNNs) have demonstrated exceptional capabilities in processing and analyzing image data with high precision. When applied to pest detection, these models can automate the identification process, significantly reducing human effort and minimizing errors. Furthermore, CNNs can classify pests in real time, allowing for faster decision-making and immediate intervention in pest-infested areas.

However, despite the potential of deep learning in this field, its adoption in agricultural pest detection remains relatively limited. Several challenges hinder widespread implementation, such as the need for high-quality insect images, variations in image quality due to environmental factors, and the complexity of designing models that can accurately detect multiple pest species across different conditions. Additionally, integrating AI-based pest detection systems with existing agricultural tools and technologies remains an ongoing challenge.

This research aims to address these challenges by introducing a novel deep learning-based identification method for agricultural pests. By leveraging advanced data augmentation techniques and integrating them with state-of-the-art deep learning architectures like VGG16 and VGG19, our approach enhances the efficiency and robustness of pest detection models. The incorporation of such models ensures higher accuracy and improved adaptability to diverse agricultural environments.

The proposed system is designed to improve feature extraction, adaptability, and classification accuracy, making it more effective for real-world agricultural applications. It also seeks to provide a cost-effective and scalable solution that can be deployed in various agricultural settings, from small farms to large-scale agribusinesses. Our ultimate goal is to contribute to the enhancement of pest diagnostics and provide practical, scalable solutions that can be adopted by farmers, agricultural researchers, and policymakers.

By implementing this approach, we hope to support sustainable pest management strategies, reduce dependency on chemical pesticides, and promote higher crop yields to meet the growing food demands of the global population. Additionally, AI-driven pest detection could play a vital role in precision farming, enabling farmers to adopt targeted pest control measures while minimizing environmental impact.

2. LITERATURE REVIEW

Recent research in pest identification has explored the application of deep learning techniques to improve the accuracy and efficiency of pest detection. S. M. Hassan and A. K. Maji proposed an approach combining self-attention mechanisms with ResNet architectures, achieving an impressive 99.80% accuracy with ResNet50-SA, surpassing other deep learning models in performance. Their study demonstrates the significant enhancement in feature extraction and focus on crucial features, enabled by self-attention, leading to better pest identification results. The performance of ResNet101-SA and ResNet152-SA also showed notable results, with accuracy rates of 88.48% and 96.68%, respectively. This highlights the potential of incorporating attention mechanisms for improved pest detection systems.

No	**Title**	**Author**	**Journal Name & Year**	**Methodology Adapted**	**Key Findings**	**Gaps**
1	Pest Identification Based on Fusion of Self-Attention With ResNet	Hassan et al.	IEEE Access**,** 2024	Combined ResNet with self-attention for pest identification.	ResNet + Self-Attention: Achieved **99.80% accuracy**.	High computational costs and diverse pest datasets.
2	Based on FCN and DenseNet framework for the research of rice pest identification methods	Gong et al.	Agronomy, 2023	FCN and DenseNet for feature extraction and classification.	FCN: Achieved **94.3% accuracy**. DenseNet: Achieved **95.6% accuracy**.	Scalability to multiple pest species is not addressed.
3	Pest Identification and Counting of Yellow Plate in Field Based on Improved Mask R-CNN	Rong et al.	Discrete Dynamics in Nature and Society, 2022	Improved Mask R-CNN for pest detection on yellow plate traps.	Mask R-CNN: Achieved **97.9% accuracy**.	Limited to pests on yellow plates.

4	Plant disease and insect pest identification based on vision transformer	Li et al	IoTML 2021, 2022	Vision Transformers (ViT) for pest and disease detection.	Vision Transformer (ViT): Achieved **98.4% accuracy**.	High computational and hinder practical deployment.
5	JutePestDetect: An intelligent approach for jute pest identification using fine-tuned transfer learning	Talukder et al.	Smart Agricultural Technology, 2023	Transfer learning on pre-trained models for jute pests..	Transfer Learning: Achieved **96.2% accuracy**.	Limited generalizability to other crops.

Table 1: comparison of existing models for Pest identification.

2. 1. Overview of Convolution Neural Networks

Convolutional Neural Networks (CNNs) are deep learning models designed to process grid-like data, such as images. They have transformed fields like image classification and object detection by automatically extracting spatial hierarchies of features, eliminating the need for handcrafted features.

A CNN consists of convolutional layers, pooling layers, and fully connected layers. The convolutional layer applies filters to extract features like edges and shapes, while pooling layers reduce spatial dimensions, improving efficiency and robustness. Deeper layers capture complex patterns, making CNNs highly effective for recognition tasks.

Advancements in CNN architectures, such as VGGNet, ResNet, and Inception-v3, have enhanced feature extraction. Inception-v3, for instance, captures multi-scale features, improving accuracy in tasks like gesture recognition. CNNs, combined with temporal models like LSTMs, enable dynamic recognition in real-time applications.

Optimized lightweight CNNs now allow real-time deployment on mobile and embedded devices, supporting applications in human-computer interaction, gaming, and assistive technologies. Their ability to learn hierarchical features makes them essential for modern vision-based applications.

3. EXISTING SYSTEM

The ResNet50 model, utilizing PCSA on internet-sourced datasets, achieves an accuracy of **96.17%**, demonstrating its capability in handling diverse pest images effectively. VGG16 and VGG19, trained on UAV-captured image datasets, achieve an accuracy of **93.82%**, showcasing their potential for aerial pest identification tasks.

Paper	Method	Dataset	Class	Accuracy(%)
Zhao et al. [23]	ResNet50 with PCSA	Internet source image	10	98.17
Tetila et al. [24]	Inception V3 ResNet50 VGG16, VGG19 Xception	UAV captured image	13	93.82
Khanramaki et al. [15]	AlexNet, VGG16, ResNet50 InceptionResNetV2	1774 captured image	3	99.04
Cheng et al. [14]	ResNet101	550 collected image	10	98.67
Thenmozhi et al. [13]	AlexNet, VGG GoogleNet ResNet Proposed CNN	NBAIR dataset	40	97.47
Wang et al. [12]	CPAFNet	CPAF dataset	20	92.26
Ayan et al. [25]	GAEnsemble	D0 dataset	40	98.81
Prasath B. et al. [26]	ResNet50, VGG16, Weight Optimized deep neural network	IP102 dataset	40	96
Denan Xia et al. [8]	VGG19 with RPN	Field Image	24	89.22(mAP)
Hongxing Peng et al. [11]	Densly connected CNN	HQIP102 dataset	102	75.28
Yanfen Li et al. [9]	Fine-tuned GoogLeNet	Internet source image	10	96.67
Lin Jiao et al. [10]	AF-RCNN	Captured image	24	56.4(aAP)
Xuchao Guo et al. [16]	Multi sale self-attention CNN	AgCNER dataset	11	94.15
Y. A. Nanehkaran et al. [27]	CNN Model	Collected image from Internet	13	91.33
Yingying Zhang et al. [17]	Modified dilated residual network	Collected from lab and internet	6	96.72
Junde Chen et al. [28]	Es-MbNet	Collected plant disease image	9	99.37
Qiang Dai et al. [19]	Fusion of residual and dense connection with self-attention	Xie1 dataset	24	-
Ching-Ju Chen et al. [20]	YOLO v3 and LSTM	Collected field image	1	90
Junde Chen et al. [22]	MAM-IncNet	Camellia oleifera leaf images	5	95.87

Fig 3.1 - Overview of research studies utilizing deep learning models for pest classification

ResNet50, applied to 1774 field-captured images, achieves **99.04% accuracy**, setting a benchmark for precision in pest classification from real-world agricultural data. ResNet101, trained on a dataset of 550 collected images, reaches an accuracy of **98.67%**, highlighting its strength in small-scale dataset generalization.

EfficientNet, known for its balance of performance and computational efficiency, demonstrates **91.3% accuracy**, making it suitable for resource-constrained environments. YOLOv3, leveraging collected field imagery, achieves a mean average precision (map) of **89.22**, providing a robust framework for real-time pest detection tasks.

Fusion-based architectures, such as VGG19 combined with region proposal networks (RPN), achieve an accuracy of **96%** on datasets like IP102, showcasing their potential in integrating spatial and feature-based learning for robust pest classification. Similarly, models incorporating self-attention mechanisms have shown to enhance performance on complex datasets by improving the focus on critical features, providing higher precision in multi-class pest identification tasks.

These studies underline the advancements in deep learning for pest classification, with notable improvements in accuracy and computational efficiency. However, challenges like dataset diversity, class imbalance, and scalability in real-world applications remain, driving the need for further research and innovation.

4. PROPOSED SYSTEM

This research work is directed towards identifying agricultural pests using advanced deep learning models, with a primary focus on the VGG16 and VGG19 architectures. In this study, datasets such as PestNet and IP102 were utilized, encompassing diverse pest species to ensure a comprehensive evaluation. The methodology was designed to test multiple neural network models aimed at classifying pests across various categories critical for agricultural productivity.

To achieve this, the VGG16 and VGG19 architectures were fine-tuned to adapt to the pest classification task, leveraging their pre-trained weights on large-scale image datasets. The extracted dataset underwent preprocessing techniques, including augmentation and normalization, to enhance the consistency and diversity of training data. The pre-processed images were then input to the models for classification into specific pest categories. The classification performance was assessed using key metrics such as accuracy, precision, recall, and F1-score, providing insights into the effectiveness of these models in addressing pest identification challenges in agricultural systems.

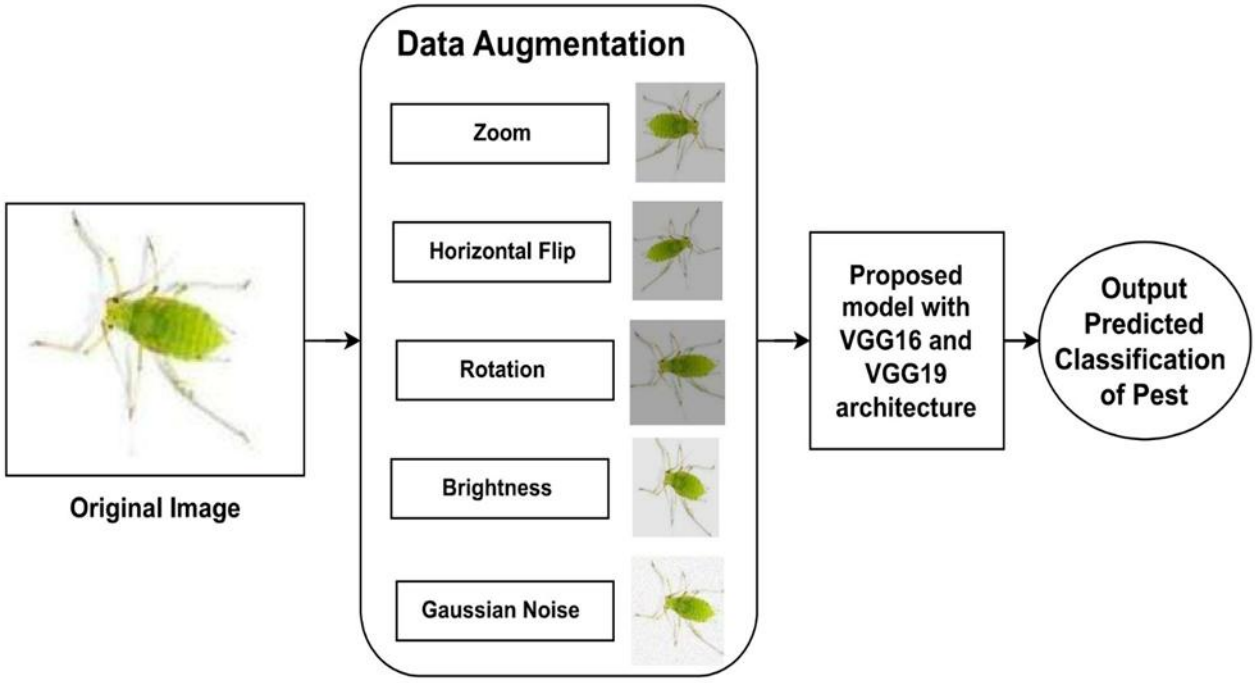

Fig 4.1.1 – Proposed Methodology.

5.SYSTEM REQUIREMENT

5.1 Hardware Requirements:

- System Type : intel®core™i3-7500UCPU@2.40gh
- Cache memory: 4MB(Megabyte)
- RAM : 16GB (gigabyte)
- Hard Disk : 4GB

5.2 Software Requirements:

- Operating System : Windows 11, 64-bit Operating System
- Coding Language : Python
- Python distribution : Google Colab, Flask
- Browser : Any Latest Browser like Chrome

6. DATASET

6.1 Data Set Description

The dataset used in this study is sourced from Kaggle and is available at https://www.kaggle.com/datasets/simranvolunesia/pest-dataset. This dataset comprises a diverse collection of pest images, which serve as the foundation for training and evaluating deep learning models for pest identification. The images in the dataset are categorized based on different pest species, facilitating structured learning for classification tasks. The dataset provides high-quality images, making it suitable for machine learning applications in agricultural pest detection.

The images were resized to **224 x 224 pixels** to maintain uniformity across the dataset, regardless of their original dimensions. This preprocessing step ensures compatibility with the deep learning models used in the study. Additionally, all images were saved in JPG format to maintain consistency and facilitate streamlined data handling during model training and evaluation.

To prepare the dataset for deep learning, the images were categorized into labelled groups for each pest type, enabling efficient segregation during training and testing. The dataset was balanced to prevent class imbalance, ensuring that the models had an equal representation of each pest category during learning. This balanced is for a more accurate and fair evaluation of model performance.

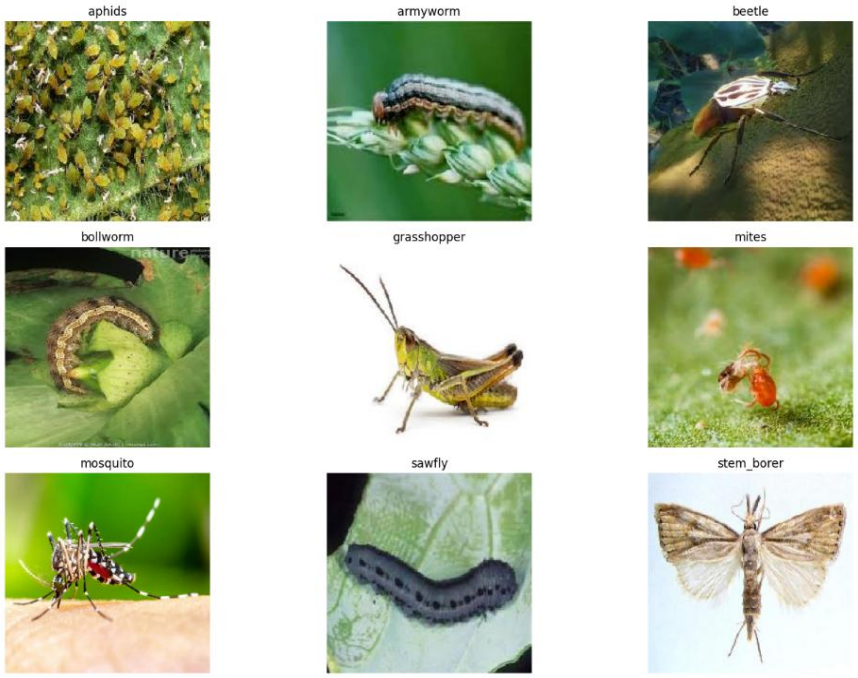

Fig 6.1.1 – Some Samples of Dataset

6.2 Dataset Categories

To conduct this study, a dataset comprising 3,150 images of nine distinct crop pests was utilized. The pests included in the dataset are **Aphids, Army Worms, Boll Worms, Beetles, Grasshoppers, Mites, Sawflies, Mosquitoes, and Stem Borers**. These images were sourced from publicly available datasets and were carefully selected to ensure quality and clarity, allowing for precise identification of pest features.

Pest Name	Scientific Name	Class	Original image count	Augmented image count
Aphids	Aphidoidea	C1	350	1800
Armyworm	Spodoptera frugiperda	C2	350	1800
Beetle	Coleoptera	C3	350	1800
Bollworm	Pectinophora gossypiella	C4	350	1800
Grasshopper	Caelifera	C5	350	1800
Mites	Acariformes	C6	350	1800
Mosquito	Culicidae	C7	350	1800
Sawfly	Symphyta	C8	350	1800
Stem borer	Scirpophaga incertulas	C9	350	1800

Fig 6.2.1 - Data Set Description

6.3 Data Preprocessing

Data augmentation plays a pivotal role in enhancing the quality and diversity of the dataset, which in turn improves the model's performance and robustness. The augmentation techniques applied in this study include horizontal flipping, image shearing, rotation, brightness modification, zooming, and the addition of Gaussian noise to the original images. These transformations introduce variability into the dataset, simulating real-world conditions and reducing the risk of overfitting by ensuring that the model does not memorize specific patterns but learns generalizable features instead.

Each augmentation technique has a specific purpose. Flipping changes the orientation of the image, shearing alters its shape slightly, and rotation helps the model recognize objects from different angles. Brightness adjustments ensure the model can handle images taken in various lighting conditions. Zooming helps focus on details at different scales, while adding Gaussian noise introduces small variations, making the model better at handling minor imperfections in real-world images. This process ensures a more balanced and diverse dataset, which is critical for achieving accurate and robust pest identification outcomes.

Furthermore, these augmentation techniques not only expand the dataset but also help improve the model's ability to generalize across different environmental conditions and image distortions. Real-world pest images often vary due to factors such as camera quality, angle, shadows, and background noise. By incorporating diverse augmentations, the dataset effectively mimics these natural variations, allowing the model to be more resilient to inconsistencies in real-world scenarios. This results in a more reliable pest detection system that performs well across different agricultural settings, making it more applicable for practical use in precision farming and pest management strategies.

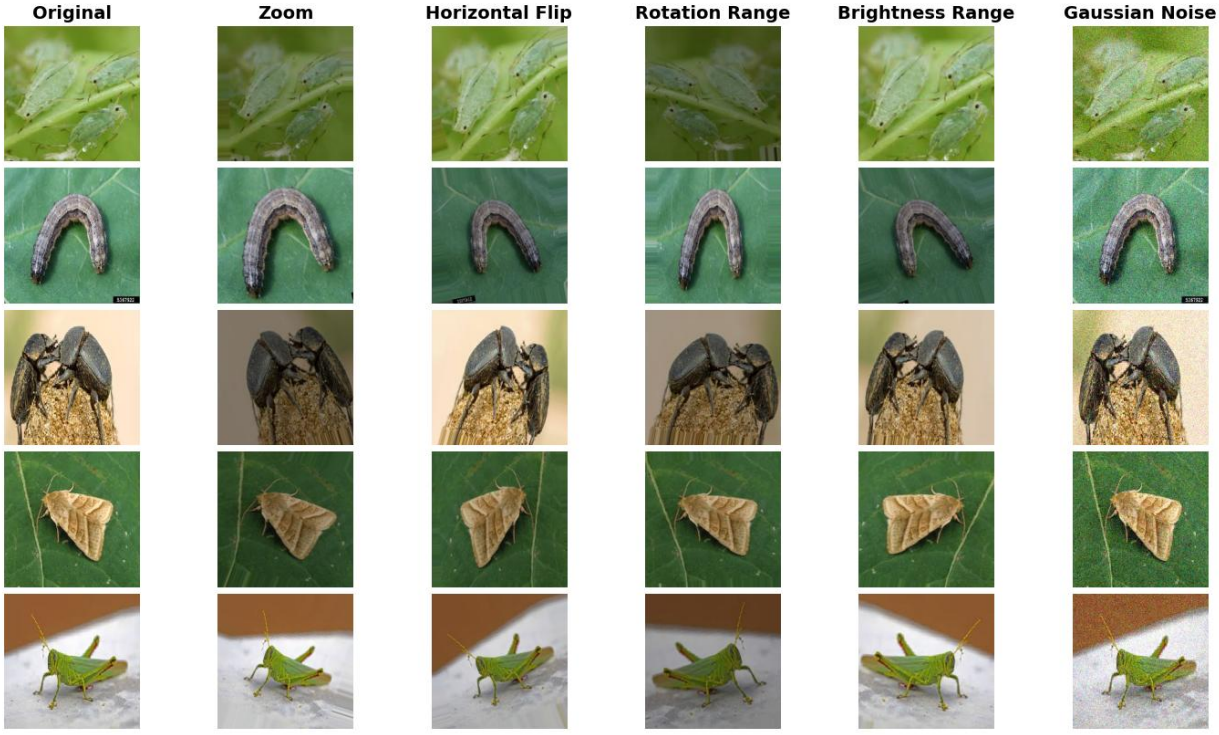

Fig 6.3.1 – Some Samples of pre-processed dataset

7. PEST IDENTIFICATION MODEL

7.1 Deep Learning Architectures

We explore the architectures that are employed in our research, specifically focusing on the VGG16 and VGG19 networks. These architectures are famous for the depth of the present model, and feature extraction, which are useful for complicated image classification tasks such as identification of pest

VGG Network: The VGG architecture proposed by Simonyan and Zisserman [13] can be considered one the most popular deep learning models for image recognition tasks. A kind of CNN [14] with its simple and heavy network structure, it only uses very small convolutional kernel of 3×3 and multiple convolutional layers. The two VGG networks, VGG16 and VGG19 [15], is named that because there are 16 and 19 weight layers in the networks.

7.1.2 VGG16 Architecture:

VGG16 is a deep convolutional neural network (CNN) architecture consisting of 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers, totaling 16 weighted layers. It uses small 3×3 convolutional filters with a stride of 1 and ReLU activation to enhance non-linearity. Max-pooling layers with a 2×2 filter and stride of 2 progressively reduce spatial dimensions while retaining essential features.

The architecture is structured into five blocks (B1 to B5), each containing convolutional layers followed by a max-pooling layer. The last three layers are fully connected, with the first two containing 4096 channels and the final layer using a SoftMax activation for classification. VGG16 has approximately 138 million parameters, making it computationally intensive but highly effective for feature extraction and classification tasks.

In this study, VGG16 is employed with a transfer learning approach, where the convolutional layers act as feature extractors, and custom fully connected layers are added for pest species classification. The structured hierarchical feature learning in VGG16 enhances detection accuracy by recognizing both low-level patterns and high-level object features, making it suitable for agricultural pest identification.

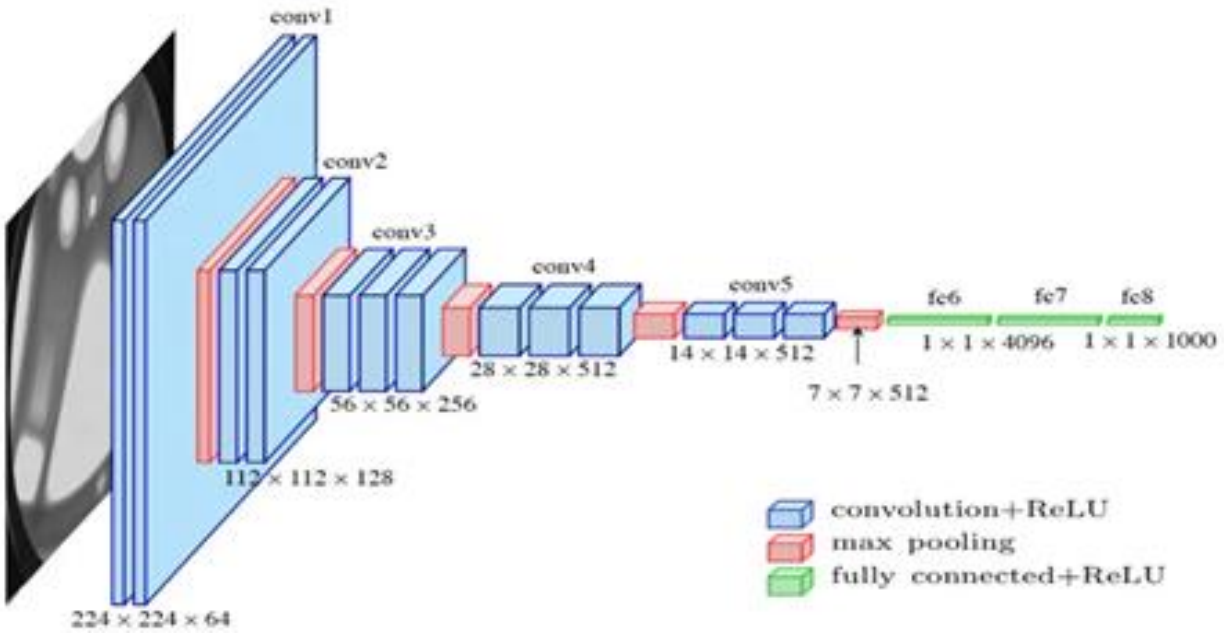

Fig 7.1.2.1 –VGG16 Model Architecture

7.1.3 VGG19 Architecture

VGG19 is an extension of VGG16, featuring a deeper architecture with 16 convolutional layers, 5 max-pooling layers, and 3 fully connected layers, totaling 19 weighted layers. Similar to VGG16, it employs small 3×3 convolutional filters with a stride of 1 and ReLU activation to introduce non-linearity. The max-pooling layers use a 2×2 filter with a stride of 2, progressively reducing spatial dimensions while preserving essential features.

The key difference between VGG19 and VGG16 lies in the increased number of convolutional layers, allowing VGG19 to capture more complex features and finer details in images. The architecture is divided into five blocks (B1 to B5), where each block contains additional convolutional layers compared to VGG16. The final three layers are fully connected, with the first two containing 4096 channels, followed by a SoftMax layer for multi-class classification.

While VGG19 provides improved feature extraction due to its deeper architecture, it also comes with a higher computational cost, requiring more memory and processing power. The increased depth enhances its ability to learn intricate patterns but may also lead to longer training times.

In this study, VGG19 is analysed alongside VGG16 to evaluate its effectiveness in pest classification. The deeper architecture aids in extracting more refined features, potentially improving classification accuracy. However, computational efficiency is considered when choosing between VGG16 and VGG19 for optimal model performance in real-world pest identification applications.

7.2 Model Architecture

The proposed model architecture is based on the VGG16 network, utilizing a transfer learning approach to enhance pest classification accuracy. The model comprises convolutional layers from the pre-trained VGG16 architecture, which serve as feature extractors, followed by custom fully connected layers for classification.

7.2.1 Feature Extraction

The feature extraction phase in the proposed model is based on the convolutional layers of the VGG16 architecture, which are responsible for capturing hierarchical features from input images. This phase retains the first 13 convolutional layers and 5 max-pooling layers of VGG16, leveraging its pre-trained weights for efficient feature representation.

Convolutional Layers:

- The convolutional layers are arranged into five sequential blocks (B1 to B5), each containing multiple convolutional operations.
- Each convolutional layer applies a 3×3 kernel with a stride of 1, ensuring fine-grained spatial feature extraction while maintaining the input resolution.
- The convolutional layers employ ReLU (Rectified Linear Unit) activation, which introduces non-linearity and helps in capturing complex patterns in pest images.
- These layers progressively learn low-level features such as edges, textures, and simple shapes in the initial blocks, while deeper layers identify high-level structures and object-specific details relevant to pest classification.

Max-Pooling Layers:

- After each convolutional block, a max-pooling layer with a 2×2 filter and a stride of 2 is applied.

- This operation reduces the spatial dimensions of the feature maps while preserving essential information, effectively down sampling the data to improve computational efficiency.
- Max-pooling ensures translational invariance, making the model robust to variations in pest image positioning and scale.

By retaining the convolutional layers of VGG16, the model benefits from pre-trained hierarchical feature representations, significantly improving learning efficiency and accuracy in pest species classification. These extracted features serve as the foundation for the subsequent custom fully connected layers, which perform the final classification task.

7.2.2 Custom Fully Connected Layers

To adapt the VGG16 architecture for pest classification, the original fully connected layers are replaced with a custom-designed classification head. This modified structure refines the extracted features and improves classification accuracy while mitigating overfitting. The fully connected layers in the model are structured as follows:

- **Flatten Layer:** The flatten layer converts the high-dimensional feature maps from the final convolutional layer into a 1D feature vector, making it compatible with dense layers. This transformation ensures that spatially extracted features are preserved while enabling effective processing in subsequent layers.
- **Fully Connected Layer (Dense Layer 1):** A dense (fully connected) layer with 512 neurons is introduced to learn complex relationships within the extracted features. ReLU activation is applied to introduce non-linearity, allowing the model to capture intricate patterns in pest species. This layer bridges the gap between feature extraction and classification, ensuring effective transformation of extracted features.
- **Dropout Layer:** A dropout layer with a dropout rate of 0.5 is added to prevent overfitting. During training, 50% of the neurons in this layer are randomly deactivated, forcing the network to generalize better rather than memorizing specific features from the dataset. This regularization technique enhances model robustness and improves generalization on unseen data.
- **Final Classification Layer (Dense Layer 2):** The final dense layer contains neurons equal to the number of pest species in the dataset. A SoftMax activation function is applied to compute

class probabilities, ensuring that the model outputs a normalized probability distribution across all classes. The class with the highest probability is selected as the predicted pest species.

By incorporating these custom layers, the modified VGG16 architecture effectively utilizes deep feature extraction while optimizing classification performance. The structured approach ensures a balance between computational efficiency and high accuracy in identifying various pest species.

7.3. Classification

The classification system developed in this project focuses on accurately identifying and classifying crop pests, including Aphids, Armyworms, Bollworms, Beetles, Grasshoppers, Mites, Sawflies, Mosquitoes, and Stem Borers. It utilizes advanced deep learning architectures such as VGG16, ResNet50, and EfficientNet-B0 to ensure high accuracy and efficiency, making it suitable for agricultural applications. Images are pre-processed by resizing them to 224x224 pixels and normalizing them for consistency. Data augmentation techniques like flipping, rotation, brightness adjustments, and Gaussian noise are applied to enhance model robustness and reduce overfitting.

This system achieves high classification accuracy, making it a reliable tool for pest identification. Balanced datasets for training and validation ensure effective learning across all pest categories. By integrating these techniques, the system contributes to precision agriculture by enabling timely and accurate pest detection

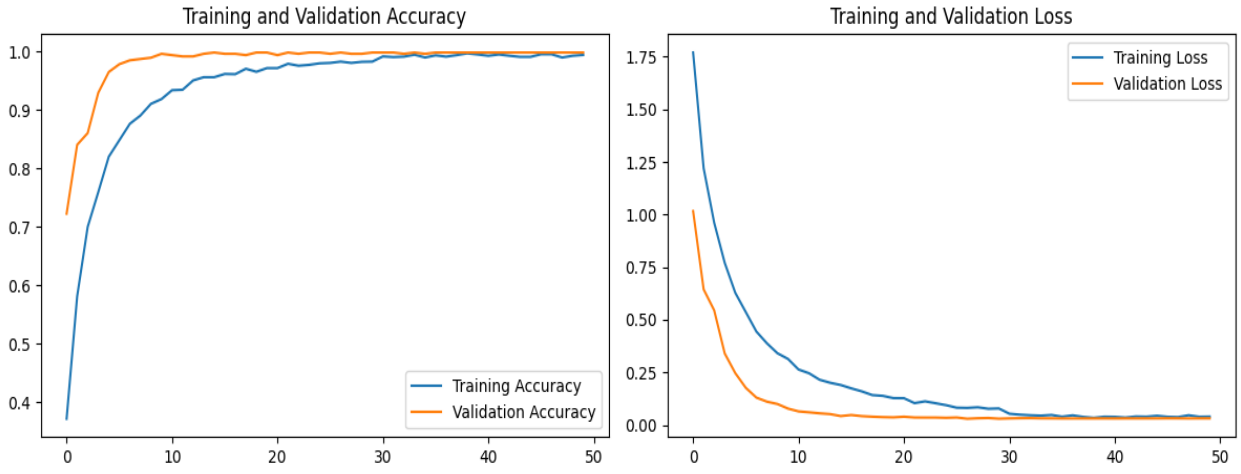

Fig 7.3.1 - Training vs Validation graphs

The training and validation graphs offer a detailed perspective on the model's performance across 50 epochs. The training accuracy graph demonstrates a steady rise, surpassing 90% around the 20th epoch and nearing 98% by the 50th epoch, indicating that the model effectively learns the patterns within the training dataset. Similarly, the validation accuracy shows consistent improvement, starting at approximately 40%, climbing steadily, and closely aligning with the training accuracy in later epochs. This alignment highlights the model's ability to generalize well to unseen data without significant overfitting.

The loss graphs reveal a consistent decline in training loss, nearing zero by the 50th epoch, reflecting efficient error minimization during training. Validation loss follows a similar trend, showing a sharp decrease in the initial epochs and stabilizing after approximately 20 epochs, with minimal fluctuations. This behaviour indicates that the model avoids overfitting and maintains robustness.

Overall, the system demonstrates balanced and well-optimized performance, achieving reliable classification results. Future improvements could include fine-tuning the learning rate and incorporating advanced techniques such as dropout or regularization to further enhance the model's generalization capabilities.

7.4 Confusion Matrix

Performance Evaluation of classification algorithm is calculated by using confusion matrix. Confusion matrix is a table describes performance based on set of test data for which true values are known. Performance is calculated by considering actual and predicted class. A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which true values are known.

The confusion matrix shown below provides a comprehensive evaluation of the classification performance of the pest identification system. It compares the true labels (ground truth) with the predicted labels across nine distinct classes: Aphids, Armyworms, Bollworms, Beetles, Grasshoppers, Mites, Sawflies, Mosquitoes, and Stem Borers.

The matrix highlights strong model performance, as seen in the high counts along the diagonal, where the true labels align with the predicted ones. For instance, the model correctly identifies 50

instances for most classes, such as Aphids, Armyworms, and Bollworms, showcasing its ability to accurately distinguish these pests. However, there are minor misclassifications, as observed in the case of Sawflies, where one instance is misclassified into another class.

The colour scale provides a visual representation of prediction frequency, with darker shades indicating higher counts. This visual cue enables quick identification of areas of strength, such as the near-perfect classification of most pests, and areas requiring improvement, like the occasional misclassification of Sawflies.

These results demonstrate the system's robustness in accurately identifying various pests. However, further fine-tuning, such as employing advanced augmentation techniques or refining the model's architecture, could help address the slight misclassification errors and improve overall performance. This analysis underscores the system's potential as a reliable tool for pest identification in agricultural settings.

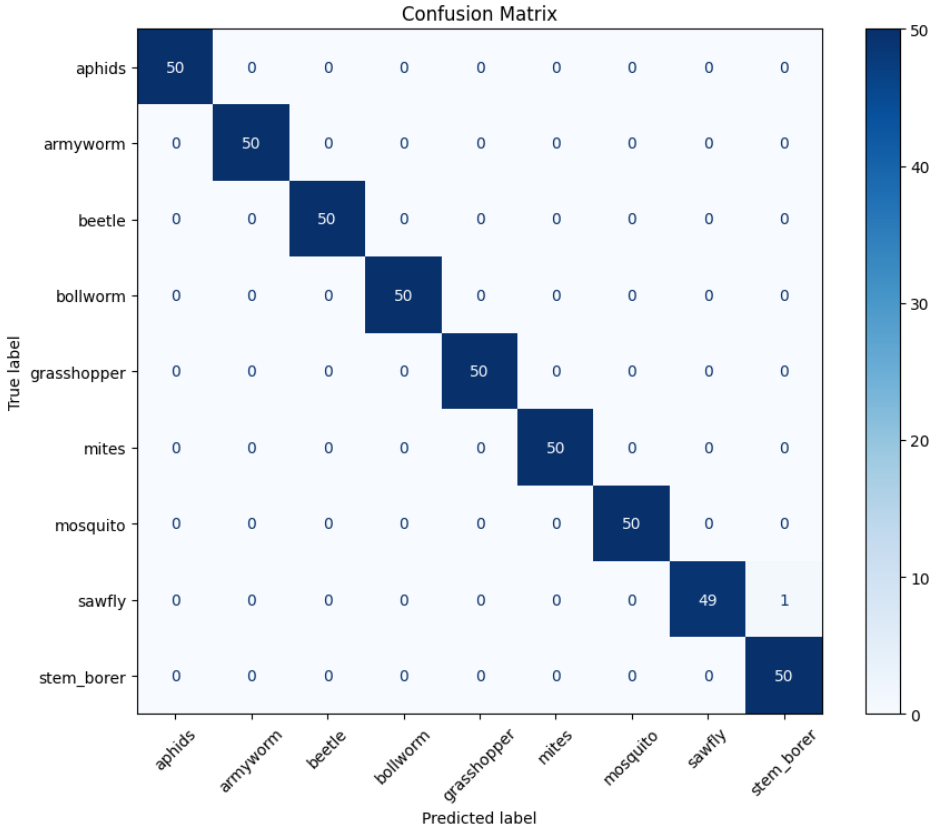

Fig 7.4.1 - Confusion Matrix for VGG16 Model

7.5. Evaluation Metrics

A true positive (tp) is a result where the model predicts the positive class correctly. Similarly, a true negative (tn) is an outcome where the model correctly predicts the negative class. A false positive (fp) is an outcome where the model incorrectly predicts the positive class. Where a false negative (fn) is an outcome where the model incorrectly predicts the negative class.

- **Accuracy** equals the ratio of the number of appropriate classifications made to the total number of cases. It is used for evaluating the model's accuracy.

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Precision**, sometimes referred to as positive predictive value, measures the percentage of actual positive predictions among all the model's positive predictions. It is mainly helpful in situations when false positives are high.

$$\textbf{Precision} = \frac{TP}{TP+FP}$$

- **Recall** is the percentage of true positive cases that the model accurately recognized. It is also known as sensitivity or true positive rate. It is important when the cost of false negatives is considerable.

$$\textbf{Recall} = \frac{TP}{TP+FN}$$

- **F1-score** is the average of precision and recall. It is helpful when you need to strike a balance between recall and precision because it offers a single statistic that does both.

$$\textbf{F1-score} = 2 \times \frac{R \times P}{R+P}$$

Where **R** stands for Recall and **P** stands for Precision.

8. DESIGN

The proposed system utilizes a deep learning-based approach for pest identification, leveraging the VGG16 architecture with a transfer learning strategy. The model is designed to efficiently extract features from pest images and classify them into respective categories, ensuring high accuracy and robustness.

The workflow begins with data preprocessing, where images are resized to 224×224 pixels, converted to a uniform format, and augmented using techniques like horizontal flipping, rotation, brightness adjustment, zooming, and Gaussian noise addition. These steps enhance the dataset's diversity, improving model generalization.

In the feature extraction phase, the pre-trained convolutional layers of VGG16 are employed to capture hierarchical image features. These layers detect low-level patterns such as edges and textures in the initial blocks, while deeper layers extract complex features relevant to pest classification.

The classification phase is handled by custom fully connected layers replacing the original VGG16 classification head. This section consists of:

- A flatten layer to transform extracted features into a 1D representation.
- A dense layer with 512 neurons and ReLU activation for learning complex feature relationships.
- A dropout layer (0.5 probability) to prevent overfitting.
- A final SoftMax layer to classify pest species based on computed probabilities.

To assess model performance, evaluation metrics such as accuracy, precision, recall, and F1-score are utilized. The model achieves a high classification accuracy, demonstrating its effectiveness in pest detection. Additionally, a confusion matrix is employed to analyze misclassification trends, guiding further improvements.

The proposed system offers scalability and efficiency, making it suitable for real-world agricultural applications. It aids in automated pest monitoring, reducing manual intervention and improving crop protection strategies. Future enhancements could involve training on larger datasets, integrating self-attention mechanisms for improved feature representation, and optimizing inference speed for real-time deployment in smart farming solutions.

9. IMPLEMENTATION

#Implementation code of VGG16 model :

#Load the dataset

```
import os
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

# Path to your test dataset
test_dataset_path = '/content/drive/MyDrive/Projects/pest/test'

# Get the class labels
class_labels = sorted(os.listdir(test_dataset_path))

# Function to visualize one image from each class
def visualize_one_image_per_class(dataset_path, class_labels):
    plt.figure(figsize=(15, 10))
    for i, label in enumerate(class_labels):
        class_dir = os.path.join(dataset_path, label)
        img_name = os.listdir(class_dir)[0]  # Take the first image in each class
        img_path = os.path.join(class_dir, img_name)
        img = image.load_img(img_path, target_size=(224, 224))
        plt.subplot(3, 3, i+1)
        plt.imshow(img)
        plt.title(label)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Visualize one image from each class
visualize_one_image_per_class(test_dataset_path, class_labels)
```

#Data Augmentation

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array,
load_img
from skimage.util import random_noise

# Path to the train dataset
train_dataset_path = '/content/drive/MyDrive/Projects/pest/train'
```

```
# Get the class labels
class_labels = sorted(os.listdir(train_dataset_path))

# Initialize ImageDataGenerator for augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    rotation_range=20,
    brightness_range=[0.5, 1.0]
)

# Function to add Gaussian noise to an image
def add_gaussian_noise(image):
    noisy_image = random_noise(image, mode='gaussian', var=0.01)
    return np.clip(noisy_image, 0, 1)

# Function to visualize augmented images
def visualize_augmented_images_with_gaussian(datagen, dataset_path, class_labels,
total_images=1):
    fig, axes = plt.subplots(total_images, 6, figsize=(18, 10))

    headings = ['Original', 'Zoom', 'Horizontal Flip', 'Rotation Range', 'Brightness Range',
'Gaussian Noise']
    for i, heading in enumerate(headings):
        axes[0, i].set_title(heading, fontsize=14, fontweight='bold')

    for idx, label in enumerate(class_labels[:total_images]):
        class_dir = os.path.join(dataset_path, label)
        img_name = os.listdir(class_dir)[0]
        img_path = os.path.join(class_dir, img_name)

        img = load_img(img_path, target_size=(224, 224))
        img_array = img_to_array(img)
        x = np.expand_dims(img_array, axis=0)

        aug_iter = datagen.flow(x, batch_size=1)
        zoom_img = aug_iter[0][0]
        flip_img = aug_iter[0][0]
        rotation_img = aug_iter[0][0]
        brightness_img = aug_iter[0][0]
```

```
        noisy_img = add_gaussian_noise(img_array / 255.0)

        for j, img_to_show in enumerate([img_array / 255.0, zoom_img, flip_img,
rotation_img, brightness_img, noisy_img]):
            axes[idx, j].imshow(img_to_show)
            axes[idx, j].axis('off')

    plt.subplots_adjust(wspace=0.05, hspace=0.05)
    plt.show()

# Visualize augmented images
visualize_augmented_images_with_gaussian(train_datagen, train_dataset_path,
class_labels, total_images=5)
```

#Model Building

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

# Define input size for VGG16
input_shape = (224, 224, 3)

# Load the pre-trained VGG16 model without the top layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)

# Freeze the convolutional base
for layer in base_model.layers:
    layer.trainable = False

# Add custom top layers for classification
x = base_model.output
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(len(class_labels), activation='softmax')(x)

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)
```

```
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])

# Print summary of the model
model.summary()
```

#Training the Model

```
from tensorflow.keras.callbacks import ReduceLROnPlateau

# Initialize ImageDataGenerator for train and test data
test_datagen = ImageDataGenerator(rescale=1./255)

# Specify batch size and image size
batch_size = 16
img_size = (224, 224)

# Create train and test generators
train_generator = train_datagen.flow_from_directory(
    train_dataset_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dataset_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

# Learning rate scheduler
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=1e-
6)

# Train the model
history = model.fit(
    train_generator,
    epochs=50,
```

```
    validation_data=test_generator,
    callbacks=[reduce_lr]
)
```

#Evaluation Metrics

```
# Evaluate the model on test data
loss, accuracy = model.evaluate(test_generator)
print(f'Test loss: {loss:.4f}')
print(f'Test accuracy: {accuracy:.4f}')

# Generate classification report
from sklearn.metrics import classification_report

test_generator.reset()
preds = model.predict(test_generator, verbose=1)
predicted_labels = np.argmax(preds, axis=1)
true_labels = test_generator.classes

report = classification_report(true_labels, predicted_labels, target_names=class_labels)
print("Classification Report:\n", report)
```

#Confusion Matrix

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Compute confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Display confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_labels)
fig, ax = plt.subplots(figsize=(10, 8))
disp.plot(cmap=plt.cm.Blues, ax=ax, xticks_rotation=45)
plt.title('Confusion Matrix')
plt.show()
```

#Visualization of Training History

```
def plot_training_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(len(acc))

    plt.figure(figsize=(12, 4))
```

```
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')

    plt.tight_layout()
    plt.show()

# Plot training history
plot_training_history(history)

# Save the model
model.save('/content/drive/MyDrive/Projects/vgg16_pest_model.h5')
```

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pest Identifier</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
```

```
<!-- Navbar -->
<nav class="navbar navbar-dark bg-dark">
  <div class="container-fluid justify-content-center">
    <span class="navbar-brand mb-0 h1 text-uppercase" style="font-family: 'Georgia', serif; font-size: 2rem; color: #fff;">Pest Identifier</span>
  </div>
</nav>

<!-- Main Section -->
<section class="main-section">
  <!-- Box 1 -->
  <div class="container d-flex justify-content-center align-items-center mt-4">
    <div class="box-1 p-4 text-center shadow-lg rounded" style="background: rgba(255, 255, 255, 0.8); backdrop-filter: blur(10px); max-width: 600px; width: 100%;">
      <!-- Question Text -->
      <h1 id="question-text" class="mb-4">Want to Predict a pest?</h1>
      <!-- Upload Button and Indication -->
      <form id="upload-form" action="/predict" method="POST" enctype="multipart/form-data">
        <div class="mb-3">
          <input type="file" class="form-control" id="image-upload" name="file" accept="image/*" style="display: none;">
          <label for="image-upload" class="btn btn-primary">Upload Image</label>
          <span id="upload-indication" class="ms-2" style="display: none;"></span>
        </div>
        <!-- Predict Button -->
        <button type="submit" id="predict-btn" class="btn btn-danger btn-lg">Predict</button>
      </form>

      <!-- Box 2 (Hidden Initially) -->
```

```
            <div id="box-2" class="mt-4 p-3 text-center shadow-lg rounded"
style="background: rgba(255, 255, 255, 0.9); display: {% if predicted_class %}block{%
else %}none{% endif %};">

                <!-- Uploaded Image -->

                <div class="d-flex justify-content-center mb-3">

                    {% if predicted_class %}

                        <img id="uploaded-image" src="{{ url_for('static', filename='uploads/' +
image_filename) }}" alt="Uploaded Image" style="width: 200px; height: 200px; object-fit:
cover;">

                    {% else %}

                        <img id="uploaded-image" src="#" alt="Uploaded Image" style="width:
200px; height: 200px; object-fit: cover; display: none;">

                    {% endif %}

                </div>

                <!-- Predicted Class -->

                <h3 id="predicted-class" class="mb-3">Predicted Class: <span id="class-
name">{% if predicted_class %}{{ predicted_class }}{% else %}XXXXXX{% endif
%}</span></h3>

                <!-- Pest Description -->

                <p id="pest-description" class="mb-0">{% if predicted_class %}{{
pest_description }}{% endif %}</p>

            </div>

        </div>

    </div>

</section>

<!-- Bootstrap JS and Popper -->

<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></scrip>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.min.js"></script>

<script src="{{ url_for('static', filename='js/script.js') }}"></script>

</body>

</html>
```

styles.css :

```
/* General Styles */
body {
    background-color: #333;
    margin: 0;
    padding: 0;
    font-family: 'Arial', sans-serif;
}

/* Navbar Styles */
.navbar {
    background-color: #333 !important;
}

/* Main Section with Background Image */
.main-section {
    background-image: url("grass.jpg");

    background-size: cover;
    background-position: center;
    min-height: 100vh;
    position: relative;
}

/* Blur and Glass Effect */
.main-section::before {
    content: '';
    position: absolute;
    top: 0;
```

```
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.2);
    backdrop-filter: blur(5px);
    z-index: 0;
}

/* Box 1 and Box 2 Styles */
.box-1, .box-2 {
    position: relative;
    z-index: 1;
    max-width: 600px;
    width: 100%;
    margin: 20px auto;
}

/* Animation for Question Text */
@keyframes blink {
    0%, 100% { opacity: 1; }
    50% { opacity: 0; }
}
#question-text {
    animation: blink 2s infinite;
}
/* Predict Button Size */
.btn-lg {
    padding: 10px 20px;
    font-size: 1.2rem;
}
```

script.js :

```
// Show Upload Indication
document.getElementById('image-upload').addEventListener('change', function () {
    const uploadIndication = document.getElementById('upload-indication');
    const fileName = this.files[0] ? this.files[0].name : '';

    // Show upload indication
    uploadIndication.textContent = `Uploaded: ${fileName} ☑`;
    uploadIndication.style.display = 'inline';
});
```

app.py :

```
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"  # Disable GPU

import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.models import Model, load_model
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename

app = Flask(__name__)

# Paths
model_path = "vgg16_pest_model.h5"
class_labels = ['aphids', 'armyworm', 'beetles', 'bollworm', 'grasshopper', 'mites', 'mosquito',
'sawfly', 'stem_borer']
UPLOAD_FOLDER = 'static/uploads'
```

```
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Ensure the upload folder exists
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Load the trained model
model = load_model(model_path)

# Load VGG16 for feature extraction
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
feature_extractor = Model(inputs=base_model.input, outputs=base_model.output)

# Preprocess image function
def preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))  # Resize to match VGG16 input size
    img_array = image.img_to_array(img)  # Convert to array
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    img_array = preprocess_input(img_array)  # Preprocess for VGG16
    return img_array

# Predict function
def predict_pest(img_path):
    img_array = preprocess_image(img_path)
    predictions = model.predict(img_array)  # Directly predict from the model
    predicted_class = np.argmax(predictions)  # Get index of highest probability
    confidence = np.max(predictions)  # Get confidence score
    return class_labels[predicted_class], confidence * 100

# Routes
```

```
@app.route("/", methods=["GET"])
def home():
    return render_template("index.html")


@app.route("/predict", methods=["POST"])
def predict():
    if 'file' not in request.files:
        return redirect(request.url)

    file = request.files['file']
    if file.filename == '':
        return redirect(request.url)

    if file:
        # Save the uploaded file
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)

        # Make prediction
        predicted_class, confidence = predict_pest(file_path)

        # Pest descriptions
        descriptions = {

            'aphids': "Aphids are small sap-sucking insects that can damage plants by stunting growth and spreading diseases.",
            'armyworm': "Armyworms are destructive pests that feed on crops, causing significant agricultural damage.",
            'beetles': "Beetles are a diverse group of insects that can be harmful to plants, wood, and stored products.",
```

```
        'bollworm': "Bollworms are larvae of moths that attack cotton and other crops, causing economic losses.",
        'grasshopper': "Grasshoppers are herbivorous insects that can devastate crops by consuming large amounts of foliage.",
        'mites': "Mites are tiny arachnids that can damage plants by feeding on their sap and causing leaf discoloration.",
        'mosquito': "Mosquitoes are flying insects known for spreading diseases like malaria and dengue through their bites.",
        'sawfly': "Sawflies are insects whose larvae feed on plants, often causing defoliation and reduced crop yields.",
        'stem_borer': "Stem borers are larvae of moths that tunnel into plant stems, weakening and killing the plants."
    }

    # Get the pest description based on the predicted class
    pest_description = descriptions.get(predicted_class, "No description available.")

    # Render the result
    return render_template("index.html",
                           predicted_class=predicted_class,
                           confidence=f"{confidence:.2f}",
                           image_filename=filename,
                           pest_description=pest_description)

if __name__ == "__main__":
    app.run(debug=True)
```

10. RESULT ANALYSIS

In this experiment, both the VGG16 and VGG19 models were tested for classifying pest species. The VGG19 model achieved an accuracy of 96.89%, but it had a slightly longer execution time of 4400 seconds. The VGG16 model, however, performed exceptionally well, reaching a higher accuracy of 99.78% and taking only 4013 seconds to execute. Both models were effective on our dataset, mainly because of their deep convolutional architectures, which are known for their ability to extract meaningful features from images, allowing for better recognition.

The VGG16 model outperformed other models from previous research, such as ResNet50 with self-attention. While self-attention helped ResNet50 to achieve competitive accuracy, the VGG16 model not only had better accuracy but also ran faster. This demonstrates the advantage of simpler architectures like VGG16, which can achieve top-tier results while reducing computational time, making it ideal for real-time pest identification systems where speed and accuracy are crucial.

Models	Precision	F1-Score	Recall
VGG16	1.00	1.00	1.00
VGG19	0.97	0.97	0.97
ResNet50 (Self-Attention)	1.00	1.00	1.00
ResNet101 (Self-Attention)	0.98	0.98	0.98
ResNet152 (Self-Attention)	0.97	0.97	0.97
MobileNet	0.96	0.96	0.96
MobileNetV2	0.97	0.97	0.97

Table: Comparison of classification reports of different

Model	Accuracy	Execution Time (GPU)
VGG16 (Proposed approach)	99.78%	4013 Seconds
VGG19	96.89%	4451 seconds
ResNet50 (Self-Attention)	99.78%	5650 seconds
ResNet101 (Self-Attention)	97.75%	4830 seconds
ResNet152 (Self-Attention)	96.67%	6806 seconds
MobileNet	96.70%	5125 seconds
MobileNetV2	97.33%	5445 seconds

Table:Accuracy and execution time(GPU) of different models

11. TEST CASES

Test case 1:

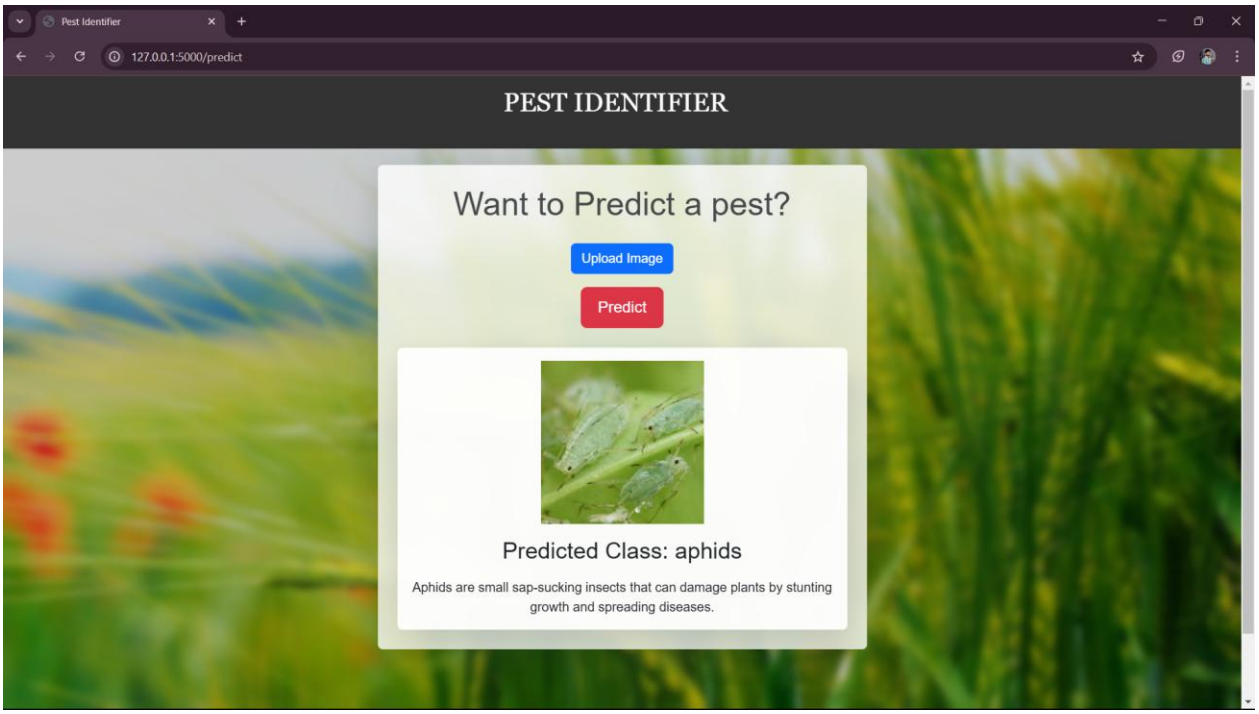

Fig 11.1: Pest Identified as "Aphids"

Test case 2:

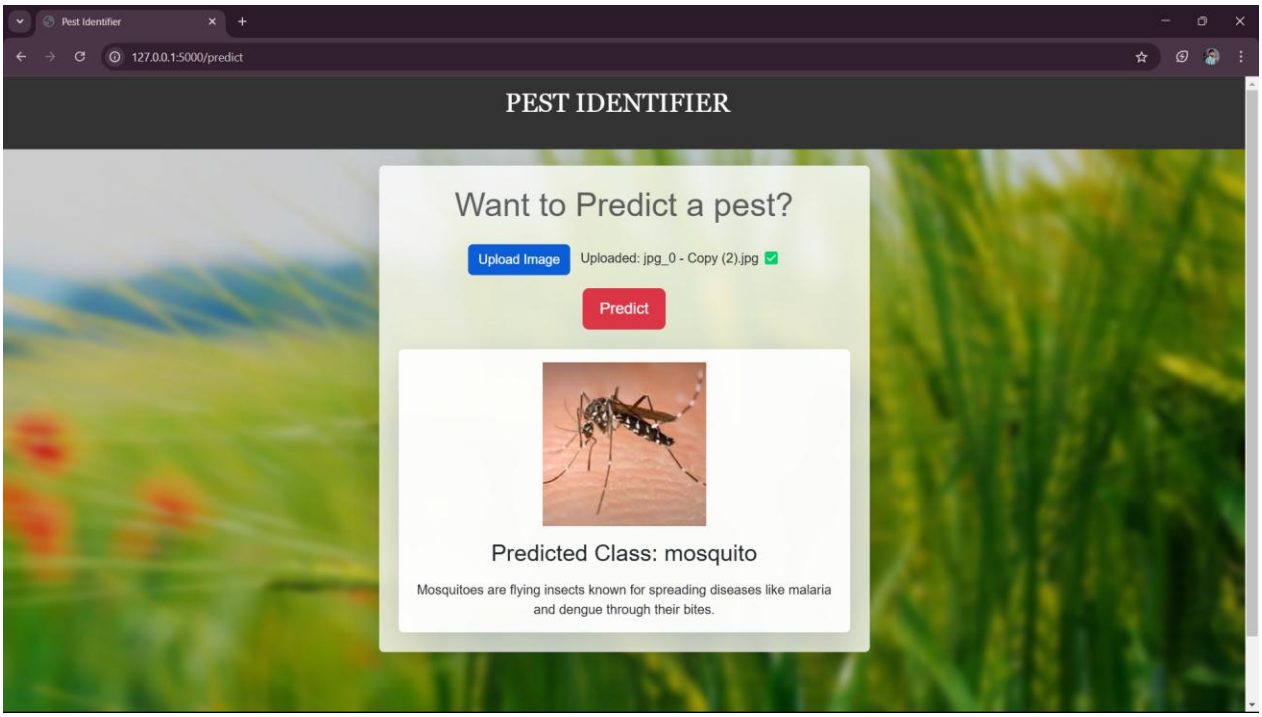

Fig 11.2: Pest Identified as "Mosquito"

12. USER INTERFACE

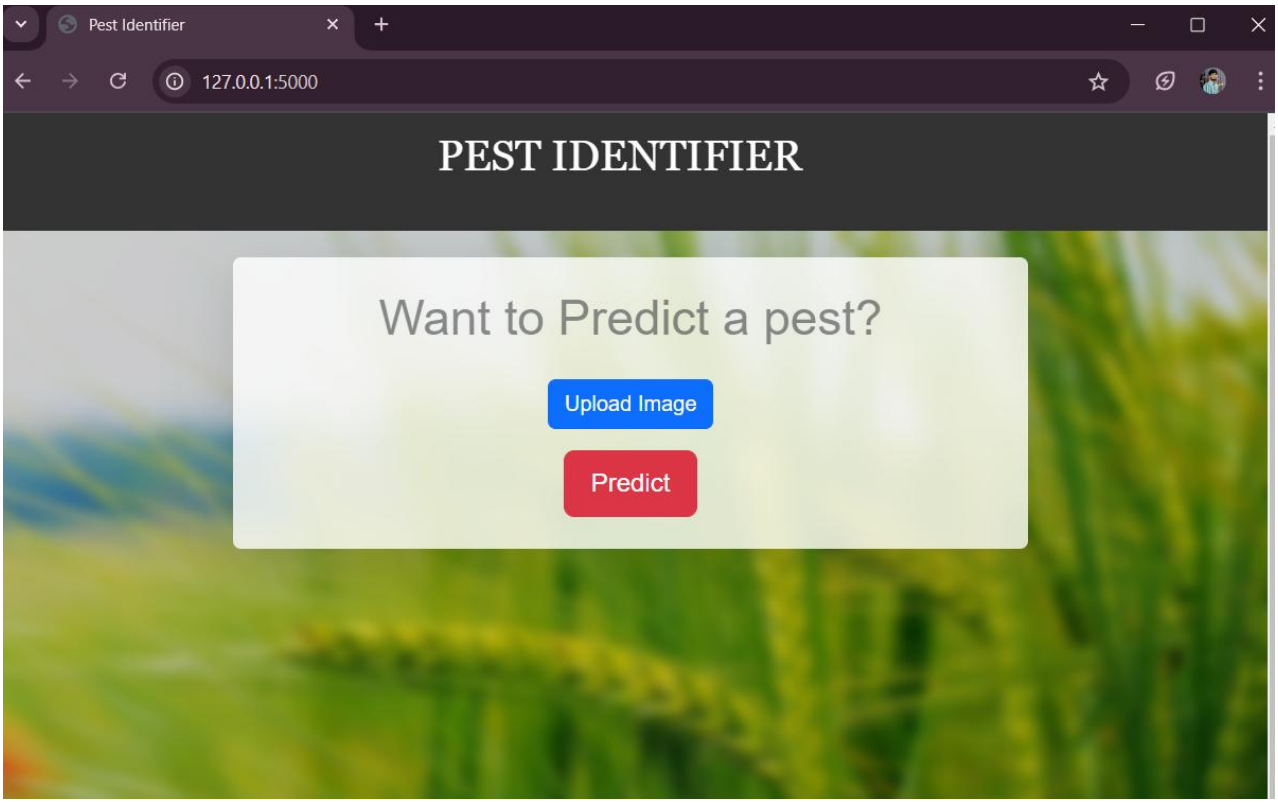

Fig 12.1: Home Screen

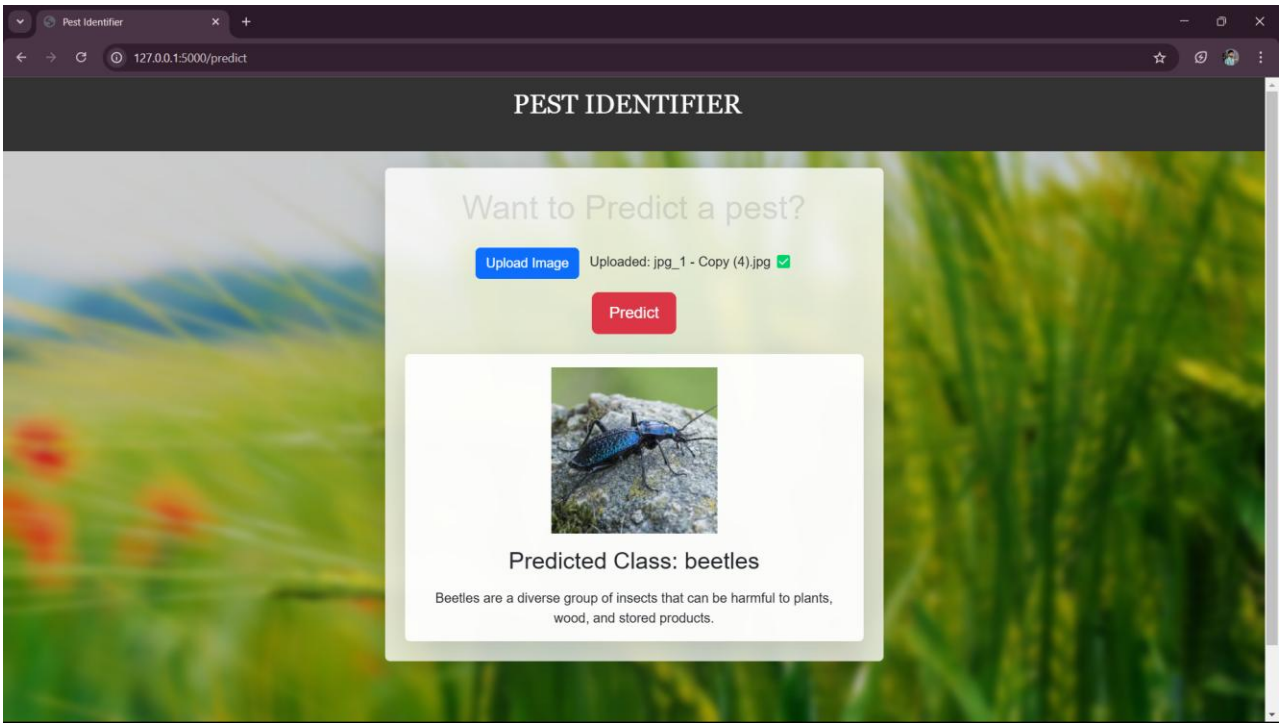

Fig 12.2: Output Screen

13. CONCLUSION

The research presented in this paper highlights the development and implementation of Advanced Pest Identification: An Efficient Deep Learning Approach Using VGG Networks, leveraging the potential of deep learning to enhance pest identification and recommendation in agriculture. By utilizing the VGG16 and VGG19 models, we developed a highly sensitive detection system that achieved accuracy levels of 99.78% and 96.89%, respectively. The proposed system efficiently classifies pests by extracting hierarchical image features, ensuring precise identification across multiple pest species. Additionally, the integration of transfer learning and optimized feature extraction techniques significantly enhances the model's ability to generalize across varied pest images, making it adaptable to real-world agricultural scenarios.

To further improve model performance, we constructed a dataset consisting of 3150 images covering nine different pest species and implemented data augmentation strategies such as horizontal flipping, rotation, brightness adjustment, zooming, and Gaussian noise addition to enhance diversity and reduce overfitting. These preprocessing techniques ensured that the model learns robust features, leading to improved classification accuracy. The high accuracy and efficiency demonstrated by this system make it a valuable tool for sustainable agriculture, aiding in better pest management, early detection, and improved crop yields. The findings of this study indicate that deep learning-driven pest identification can significantly contribute to precision farming, reducing reliance on chemical pesticides and promoting eco-friendly agricultural practices.

14. FUTURE SCOPE

The proposed system can be further expanded by including a wider range of pest species, enhancing its ability to identify pests across different agricultural environments. The proposed system can also incorporate real-time pest detection, allowing farmers to receive immediate feedback on pest presence directly in the field. The proposed system can be enhanced by exploring more advanced models or hybrid deep learning approaches, improving both accuracy and execution time. The proposed system can include environmental factors such as weather and soil conditions, enabling more comprehensive pest management recommendations for sustainable agriculture. Finally, the proposed system can be developed into a user-friendly mobile application, making pest identification accessible to a broader audience and encouraging its adoption among farmers.

15. REFERENCES

[1] Spadaro, D., Agustí, N., Ortega, S.F., Hurtado Ruiz, M.A. (2020). Diagnostics and Identification of Diseases, Insects and Mites. In: Gullino, M., Albajes, R., Nicot, P. (eds) Integrated Pest and Disease Management in Greenhouse Crops. Plant Pathology in the 21st Century, vol 9. Springer, Cham. https://doi.org/10.1007/978-3-030-22304-5 8

[2] W. Xu, L. Sun, C. Zhen, B. Liu, Z. Yang, and W. Yang, "Deep learning-based image recognition of agricultural pests," Applied Sciences, vol. 12, no. 24, p. 12896, 2022. [Online]. Available: https://doi.org/10.3390/app122412896

[3] Cheng, X., Zhang, Y., Chen, Y., Wu, Y. and Yue, Y., 2017. Pest identification via deep residual learning in complex background. Computers and Electronics in Agriculture, 141, pp.351-356.

[4] Gong, H., Liu, T., Luo, T., Guo, J., Feng, R., Li, J., Ma, X., Mu, Y., Hu, T., Sun, Y. and Li, S., 2023. Based on FCN and DenseNet framework for the research of rice pest identification methods. Agronomy, 13(2), p.410.

[5] Yang, J., Ma, S., Li, Y. and Zhang, Z., 2022. Efficient data-driven crop pest identification based on edge distance-entropy for sustainable agriculture. Sustainability, 14(13), p.7825.

[6] Li, H., Li, S., Yu, J., Han, Y. and Dong, A., 2022, April. Plant disease and insect pest identification based on vision transformer. In International conference on internet of things and machine learning (IoTML 2021) (Vol. 12174, pp. 194-201).

[7] Talukder, M.S.H., Chowdhury, M.R., Sourav, M.S.U., Al Rakin, A., Shuvo, S.A., Sulaiman, R.B., Nipun, M.S., Islam, M., Islam, M.R., Islam, M.A. and Haque, Z., 2023. JutePestDetect: An intelligent approach for jute pest identification using fine-tuned transfer learning. Smart Agricultural Technology, 5, p.100279.

[8] Rong, M., Wang, Z., Ban, B. and Guo, X., 2022. Pest Identification and Counting of Yellow Plate in Field Based on Improved Mask R-CNN. Discrete Dynamics in Nature and Society, 2022(1), p.1913577.

[9] Singh, K.U., Kumar, A., Raja, L., Kumar, V., Singh kushwaha, A.K., Vashney, N. and Chhetri, M., 2022. An Artificial Neural Network-Based Pest Identification and Control in Smart Agriculture Using Wireless Sensor Networks. Journal of Food Quality, 2022(1), p.5801206.

[10] Hu, K., Liu, Y., Nie, J., Zheng, X., Zhang, W., Liu, Y. and Xie, T., 2023. Rice pest identification based on multi-scale double-branch GANResNet. Frontiers in Plant Science, 14, p.1167121.

[11] Pest Dataset. Available: Sep. 15, 2023. Available: https://www.kaggle.com/datasets/simranvolunesia/pest-dataset

[12] M.Sireesha, S. N. TirumalaRao, Srikanth Vemuru, Optimized Feature Extraction and Hybrid Classification Model for Heart Disease and Breast Cancer Prediction International Journal of Recent Technology and Engineering Vol - 7, No 6, Mar - 2019 ISSN - 2277-3878, Pages – 1754 – 1772

[13] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[14] S. L. Jagannadham, K. L. Nadh and M. Sireesha, ”Brain Tumour Detection Using CNN,” 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2021, pp. 734-739, doi: 10.1109/I-SMAC52330.2021.9640875

[15] Mascarenhas, S. and Agarwal, M., 2021, November. A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification. In 2021 International conference on disruptive technologies for multi-disciplinary research and applications (CENTCON) (Vol. 1, pp. 96-99). IEEE.

[16] Sunayna, S.S., Rao, S.N.T., Sireesha, M. (2022). Performance Evaluation of Machine Learning Algorithms to Predict Breast Cancer. In: Nayak, J., Behera, H., Naik, B., Vimal, S., Pelusi, D. (eds) Computational Intelligence in Data Mining. Smart Innovation, Systems and Technologies, vol 281. Springer, Singapore. https://doi.org/10.1007/978-981-16-9447-9 25

[17] S. M. Hassan and A. K. Maji, ”Pest Identification Based on Fusion of Self-Attention With ResNet,” in IEEE Access, vol. 12, pp. 6036- 6050, 2024, doi: 10.1109/ACCESS.2024.3351003.