Integrating CNN,LSTM with DenseNet201 for Efficient Real-Time Plant Disease Detection

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

P.Ujwala Devi (21471A05O3)
R.Nandini (21471A05O7)
S.Vasantha (21471A05P3)

Under the esteemed guidance of

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

Head of the Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under Tire -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 "Integrating CNN,LSTM with DenseNet201 for Efficient Real-Time Plant Disease Detection**" is a bonafide work done by the team** P.Ujwala Devi(21471A05O3), R.Nandini (21471A05O7), S.Vasantha (21471A05P3) in partial fulfillment 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. S. N. Tirumala Rao, M.Tech., Ph.D.,
Professor & HOD

PROJECT CO-ORDINATOR

D.Venkata Reddy, B.Tech., M.Tech.,
Associate Professor

HEAD OFTHE DEPARTMENT

Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,
Professor& HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "**INTEGRATING CNN,LSTM WITH DENSENET201 FOR EFFICIENT REAL-TIME PLANT DISEASE DETECTION**" 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.

P.Ujwala Devi (21471A05O3)
R.Nandini (21471A05O7)
S.Vasantha (21471A05P3)

ACKNOWLEDGEMENT

We wish to express my thanks to carious 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 thiscourse. 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 whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **D.Venkata Reddy,** **B.Tech, M.Tech.,** 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 to 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 involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

P.Ujwala Devi (21471A05O3)

R.Nandini (21471A05O7)

S.Vasantha (21471A05P3)

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 expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work 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 programme 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 the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

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

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

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 analyse 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, andsynthesis 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 engineeringactivities 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 the professional engineering

solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

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

8. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

9. 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.

10. 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.

11. 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: Analyze 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	Gathering the requirements and defining the problem, plan to develop a model for recognizing image manipulations using CNN and ELA	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement i critically analyzed, the process mode is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and everyphase ofthe work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection of forged videos	PO4, PO7
C32SC4.3	The physical design includes websiteto check whether an image is real or fake	PO5, PO6

ABSTRACT

Quick detection of plant diseases and pests is of vital importance for preventing huge loses in agriculture and the environment by the hazardous of pesticide use and looks at the utilization of machine learning models,especially convolutional neural network(CNNs),for the detection and classification of plant diseases and pests.Different methods such as supervised as well as unsupervised learning jotted down.A unique CNN-LSTM+DENSENET201 hybrid model was developed by creating a deep feature extraction of pre-trained models such as DenseNet,ResNet,and GoogleNet with an LSTM ensemble classifier.The experimental studies on the plant datasets which included the images of various diseases of the crop showed the better accuracy and robustness of the hybrid model.CNN-LSTM+DenseNet201 model is one of the few that is over 99.4% accurate in real-time disease detection and outpaces other traditional and transfer learning-based models.By using unsupervised methods like anomaly detection and image restoration,the research avoids the need for high-quality labeled data sets when creating a cost-effective solution for the farmer.Further work will concentrate on the improvement of the model's scalability along with the testing of its performance on additional datasets and plant types.

INDEX

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

➢ Introduction

Agriculture is the backbone of many economies worldwide, and maximizing crop yield while minimizing losses is critical for ensuring food security. Early and accurate detection of plant diseases plays a crucial role in achieving this objective, as undetected diseases can lead to significant losses. Traditional disease detection methods involve manual inspection by farmers or agricultural experts, which is time-consuming, labor-intensive, and prone to human errors. Moreover, these conventional techniques lack scalability and efficiency, making them unsuitable for large-scale farming operations.

With the advent of modern technology, interest has been growing in the application of machine learning and deep learning techniques for plant disease detection. These advanced methodologies provide a more systematic, automated, and precise way of identifying plant infections, ultimately improving crop management and reducing losses. Among machine learning approaches, unsupervised learning techniques, such as image restoration, have demonstrated their potential in plant disease detection without requiring labeled data [1].

Meanwhile, supervised methods, including k-means clustering and Support Vector Machines (SVM), have been utilized for disease identification. However, supervised learning methods struggle to handle complex and diverse plant images, limiting their effectiveness in real-world agricultural scenarios [2].

Deep learning has emerged as a powerful tool in plant disease detection, with Convolutional Neural Networks (CNNs) leading the way. CNNs are particularly effective in extracting spatial features from images, outperforming traditional machine learning methods in accuracy and reliability [3]. Furthermore, hybrid models that integrate CNNs with pre- trained architectures, such as DenseNet, ResNet, and GoogleNet, have demonstrated superior performance in real-time disease identification [3][4]. The ability to deploy these models on portable devices, such as Raspberry Pi, makes them even more practical for real-time disease monitoring in agricultural fields [5].

The proposed hybrid model, CNN-LSTM+DenseNet201, builds upon these advancements by incorporating three key components: CNN for efficient spatial feature extraction, LSTM (Long Short-Term Memory) to capture temporal disease progressions, and DenseNet201 for optimal feature reuse and gradient flow. This combination enhances accuracy and computational efficiency, making the model highly suitable for agricultural

applications [6].

The hybrid approach not only improves disease detection capabilities but also adapts well to various plant species and environmental conditions, thereby supporting precision agriculture.

Experimental results have demonstrated the superior performance of the CNN-LSTM+DenseNet201 hybrid model, achieving an impressive accuracy of 99.4%—outperforming CNN-ResNet and CNN-GoogleNet models [3][7].

Furthermore, the use of data augmentation techniques has helped overcome the limitations posed by smaller datasets, ensuring robust and reliable disease classification across diverse conditions [8].

The scalability and adaptability of this hybrid architecture make it a highly promising solution for precision agriculture, enabling farmers to make informed decisions and take timely preventive measures against plant diseases [9][10].

By leveraging cutting-edge deep learning techniques, this model provides an efficient, automated, and highly accurate approach to plant disease detection, ultimately contributing to increased crop yields and reduced losses. The integration of AI-driven solutions in agriculture marks a significant step toward sustainable and technology-driven farming practices, ensuring food security for future generations.

Agriculture is a crucial sector that directly impacts global food security and economic stability. One of the biggest challenges in agriculture is the early and accurate detection of plant diseases, which, if left unchecked, can lead to significant crop losses. Farmers often struggle to detect diseases in the early stages, leading to widespread infections that reduce yield quality and quantity. Early disease detection enables timely intervention, helping to minimize damage, increase productivity, and reduce the economic burden on farmers[1][2].

Early detection of plant diseases also plays a crucial role in maintaining soil health and ensuring sustainable farming practices.Unchecked plant infections can spread rapidly, affecting neighboring crops and leading to increased pesticide usage.

This not only results in financial losses for farmers but also poses environmental risks by contaminating soil and water sources[3]. Therefore, efficient disease monitoring systems are necessary to balance agricultural productivity with environmental sustainability.

Additionally, with the growing global population, there is a rising demand for food production. Ensuring that crops remain healthy and free from disease is essential to meeting food security goals[4][5]. Undetected diseases can lead to lower yields, higher food prices,

and economic instability in agricultural-dependent regions. By integrating advanced disease detection methods, farmers can enhance crop quality and contribute to a more resilient agricultural sector.

Agriculture plays a vital role in sustaining global food production, yet it faces persistent challenges due to plant diseases, climate variations, and limited access to advanced technology in many regions. Traditional methods of disease detection rely on manual inspection, which is time-consuming, error-prone, and inefficient for large-scale farming. The increasing demand for food due to population growth necessitates efficient, scalable, and accurate disease detection methods to ensure high crop yields and food security[6].

- **The Need for Early Disease Detection**

Crop diseases significantly impact agricultural productivity, often leading to severe economic losses for farmers and food shortages worldwide. Early identification of plant diseases can prevent large-scale outbreaks, enabling timely intervention to minimize crop damage. However, traditional disease detection approaches are reactive rather than proactive, often detecting infections only when symptoms become visible[7].

By the time diseases are detected, they may have already spread extensively, making control measures less effective and more expensive. An intelligent, automated system for early disease detection can address this gap by providing real-time monitoring and predictive capabilities.

- **Addressing the Growing Agricultural Challenges**

Agriculture is the backbone of global food production, providing sustenance to billions of people worldwide. However, the industry faces numerous challenges, including unpredictable climate changes, soil degradation, and the rapid spread of plant diseases. Among these, plant diseases significantly impact crop yields and

quality, leading to substantial economic losses for farmers and threatening food security. The early detection and management of plant diseases are crucial to mitigating these effects, but traditional methods are often inefficient and labor-intensive[8].

The growing global population has intensified the demand for food, necessitating the adoption of modern technologies to improve agricultural productivity. Farmers need reliable, automated solutions that can accurately identify plant diseases in their early stages to prevent large-scale outbreaks. Implementing advanced deep learning models for real-time disease detection can transform agricultural practices, making them more efficient and sustainable.

➢ Challenges in Traditional Approaches

Manual plant disease identification is highly dependent on farmers' expertise and experience, leading to inconsistencies and misdiagnoses. Additionally, the physical labor required to monitor vast agricultural fields limits the scalability of traditional approaches. Chemical treatments, often used as a preventive measure, contribute to environmental pollution and increase production costs. These limitations highlight the urgent need for an efficient and intelligent disease detection system that minimizes human effort, enhances accuracy, and reduces reliance on harmful chemicals[9][10].

Another major limitation is the inability to detect diseases in their early stages. Many plant diseases begin at a microscopic level, showing no visible symptoms until they have already spread. By the time farmers recognize an infection, significant damage may have already occurred, reducing the effectiveness of disease management strategies[11]. This delayed detection often results in increased use of pesticides and fungicides, which not only escalate production costs but also contribute to environmental degradation and the emergence of chemical-resistant pathogens[12].

2.LITERATURE SURVEY

Pei, M., Kong, M., Fu, M., Zhou, X., Li, Z., and Xu, J. [1] explored the application of unsupervised learning techniques for plant leaf pest and disease detection. Their research focused on the effectiveness of clustering algorithms, such as k-means and hierarchical clustering, in identifying patterns within plant disease symptoms without the need for labeled datasets. By analyzing visual features extracted from plant images, they demonstrated that unsupervised learning methods could successfully group similar disease characteristics, making it a viable alternative to traditional supervised approaches that rely on extensive annotated data. The study emphasized that unsupervised learning could help in early-stage disease detection and reduce dependency on manually labeled samples, which is often time-consuming and expensive.

Reddy, J. N., Vinod, K., and Ajai, A. R. [2] conducted a comparative analysis of various classification algorithms for plant leaf disease detection. Their study evaluated the performance of Support Vector Machines (SVM), Decision Trees, and Neural Networks on different plant disease datasets. They concluded that deep learning-based models, particularly Convolutional Neural Networks (CNNs), outperformed conventional machine learning algorithms in terms of classification accuracy, robustness, and adaptability to different plant species. The research also highlighted the limitations of traditional machine learning techniques, which rely heavily on handcrafted feature extraction and are less efficient in capturing complex disease patterns compared to deep learning models.

Shafik, W., Tufail, A., Liyanage, C. D. S., and Apong, R. A. A. H. M. [3] developed a novel Convolutional Neural Network (CNN) model specifically designed for plant pest detection and disease classification. Their deep learning model was trained on a large dataset of plant images containing various types of diseases and pests. The results demonstrated that their proposed CNN model achieved significantly higher accuracy compared to existing machine learning approaches. They emphasized the superior generalization capabilities of CNNs, particularly their ability to automatically extract complex spatial patterns from images, making them highly effective for plant disease classification tasks.

Türkoğlu, M., and Hanbay, D. [4] investigated the effectiveness of deep learning-based feature extraction techniques for plant disease and pest detection. Their study

focused on CNN- based feature extraction and its impact on classification performance. By applying deep feature extraction methods, they demonstrated that the extracted features significantly improved classification accuracy, reducing false positive and false negative rates compared to traditional

computer vision techniques. Their findings suggested that deep learning can play a crucial role in automating plant disease detection with greater reliability.

Prathima, K., Kanchan, R. G., Arekal, S., Shalini, A. N., and Mishra, G. [5] explored the integration of machine learning with Internet of Things (IoT)-based monitoring systems for agricultural pest and disease detection. Their research demonstrated how real-time data collection from sensors and cameras, combined with machine learning models, could enhance early disease detection and assist farmers in making informed decisions. By leveraging IoT- enabled devices, their approach aimed to improve the efficiency of disease monitoring systems and enable prompt intervention to prevent crop losses.

Sameer, S., Niharika, B. D., Vasavi, S., Rohith, M., and Abhishek, V. R. [6] proposed an enhanced AlexNet model for plant pest and disease detection. Their approach utilized transfer learning, where a pre-trained AlexNet architecture was fine-tuned using plant disease datasets. The results indicated that transfer learning improved classification accuracy, particularly when dealing with limited training samples. Their study highlighted the advantages of using pre- trained deep learning models in agricultural applications, as they require less computational power and training time compared to training models from scratch.

Rajesh, B., Vardhan, M. V. S., and Sujihelen, L. [7] applied Decision Tree-based models for plant leaf disease detection and classification. While their study acknowledged the interpretability and simplicity of Decision Trees, the findings suggested that these models struggle with high-dimensional image data. The research highlighted that deep learning models, particularly CNNs, provide better accuracy and generalization capabilities, making them more suitable for plant disease classification.

Jia, W., Yang, N., Lu, Y., and Deng, P. [8] explored the impact of data augmentation techniques on plant pest and disease detection. Their research investigated how synthetic data generation, such as image rotation, flipping, scaling, and adding noise, could enhance model robustness and adaptability. The results demonstrated that data augmentation improved classification performance across different plant species and environmental conditions, making deep learning models more reliable for real-world

agricultural applications.

Wang, Q., He, G., Li, F., and Zhang, H. [9] introduced a novel database for plant disease and pest classification. Their dataset consisted of high-quality images of various plant diseases, carefully curated to facilitate the training of deep learning models. The research emphasized the importance of standardized datasets in developing accurate and generalizable models for plant disease detection.

Li, L., Zhang, S., and Wang, B. [10] conducted a comprehensive review on deep learning-based plant disease detection and classification. Their study analyzed various architectures, including CNNs, Recurrent Neural Networks (RNNs), and hybrid models. The review highlighted the advantages and limitations of each deep learning approach, providing insights into their effectiveness for different plant disease detection scenarios.

Demilie, W. B. [11] performed a comparative study on plant disease detection techniques, evaluating different deep learning models such as CNNs, ResNet, and EfficientNet. Their research identified the most effective architectures for specific plant diseases, providing valuable insights into the selection of models based on dataset characteristics and computational efficiency.

Sangeetha, T., and Mohanapriya, M. [12] explored machine learning and deep learning algorithms for plant disease and pest detection. Their study focused on hybrid models that combine traditional machine learning techniques with deep learning approaches to improve classification accuracy. They also examined the computational efficiency of these models, emphasizing the need for optimizing deep learning architectures to reduce processing costs.

Chaitra, S., Ghana, S., Singh, S., and Poddar, P. [13] designed a deep learning model for image-based plant disease detection on edge devices. Their research demonstrated the feasibility of deploying AI-powered disease detection models on low-resource hardware, such as mobile devices and embedded systems. Their work aimed to make automated plant disease detection accessible to small-scale farmers who may not have access to high-end computing resources.

Liu, J., and Wang, X. [14] conducted a review of various deep learning approaches for plant disease and pest detection. Their study summarized recent advancements in the field, discussing key challenges and future research directions. They emphasized the need for improved data collection techniques, real-time monitoring systems, and better model interpretability to enhance practical applications in precision agriculture.

3. SYSTEM ANALYSIS

3.1 Existing System

Various machine learning and deep learning models have been employed in plant disease detection, each with unique strengths and applications. Pei et al. [1] utilized unsupervised learning techniques, such as K-Means Clustering and Hierarchical Clustering, to group similar disease patterns without requiring labeled data. This approach is particularly useful when annotated datasets are limited. Clustering techniques help automate early disease detection by identifying patterns in plant leaf images, reducing the need for manual intervention.

In contrast, Reddy et al. [2] compared the effectiveness of Support Vector Machines (SVM), Decision Trees, and Convolutional Neural Networks (CNNs). While SVM and Decision Trees are traditional machine learning models, they struggled with complex image data. CNNs, however, outperformed them by automatically extracting intricate visual features, making them more effective in plant disease classification. Similarly, Shafik et al. [3] developed a custom CNN model, which consisted of multiple convolutional and pooling layers. This structure allowed for the automatic extraction of detailed spatial features, leading to higher accuracy in detecting pests and diseases.

Türkoğlu & Hanbay [4] took a different approach by using CNN-based feature extraction techniques. Instead of employing CNNs directly for classification, they extracted deep features from convolutional layers and fed them into another classifier. This method reduced false positives and false negatives, improving the reliability of disease detection systems. Prathima et al. [5] integrated IoT-based Machine Learning models to monitor plant health in real-time. By utilizing IoT sensors and cameras, they collected live data, which was analyzed by ML models for early disease detection, preventing significant crop losses.

Sameer et al. [6] leveraged AlexNet with Transfer Learning, an approach that fine-tunes a pre-trained CNN model on plant disease datasets. AlexNet, with its five convolutional layers, was able to recognize complex disease patterns even with a small dataset. This was particularly beneficial when training data was limited. Rajesh et al. [7] compared Decision Trees and CNNs, ultimately concluding that CNNs were superior due to their ability to handle high-dimensional image data effectively.

Jia et al. [8] introduced data augmentation to enhance the performance of CNN-based models. Techniques such as image rotation, flipping, scaling, and noise addition were used to increase dataset diversity, improving model robustness and generalization.

Wang et al. [9] focused on the importance of high-quality datasets by creating a standardized dataset specifically designed for training CNN models. Their research emphasized that accurate plant disease detection relies heavily on clean and diverse datasets.

Li et al. [10] explored hybrid deep learning models, combining CNNs and Recurrent Neural Networks (RNNs). While CNNs extracted spatial features from images, RNNs captured sequential dependencies in time-series plant health data. This hybrid approach led to improved accuracy in detecting plant diseases over time. Demilie [11] compared different CNN architectures, including ResNet and EfficientNet. ResNet's residual connections helped deep networks learn better features, while EfficientNet achieved high accuracy with fewer parameters, making it computationally efficient.

Sangeetha & Mohanapriya [12] proposed a hybrid machine learning and deep learning model, combining traditional ML algorithms such as SVM and Decision Trees with CNNs. This combination provided both computational efficiency and high classification accuracy. Chaitra et al. [13] optimized deep learning models for edge devices, creating lightweight CNNs for mobile and embedded systems. Their approach enabled real-time, low-power disease detection, making it practical for small-scale farmers.

Finally, Liu & Wang [14] conducted a comprehensive review of deep learning advancements in plant disease detection. Their study analyzed challenges in dataset collection, model optimization, and real-time monitoring, while also proposing improvements in CNN architectures to enhance performance.

Overall, these studies highlight CNNs as the dominant approach for plant disease detection, with additional contributions from unsupervised learning, IoT, data augmentation, and hybrid models. The advancements in these techniques are crucial in developing automated, accurate, and scalable solutions for agricultural disease monitoring.

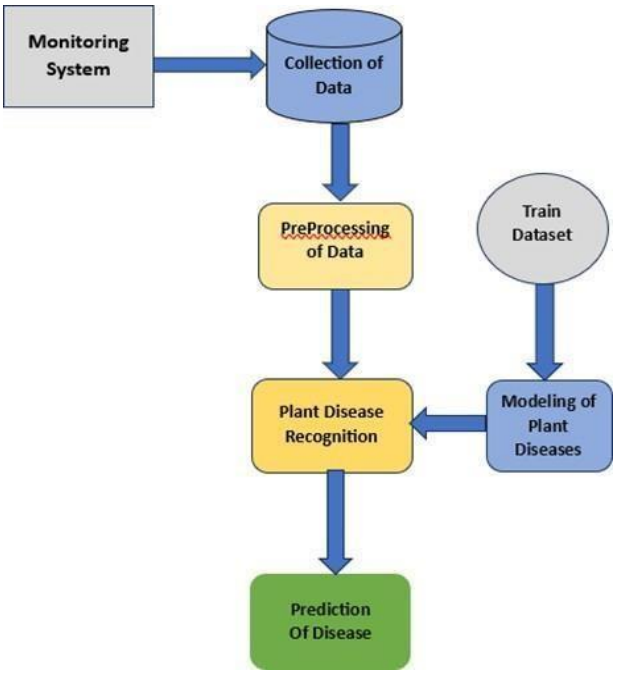


Fig 3.1.1 Flow chart of existing system

Plant disease detection plays a vital role in ensuring the health and productivity of crops. Diseases in plants, caused by pathogens such as fungi, bacteria, viruses, or pests, can lead to significant agricultural losses. Early detection is essential for effective disease management, minimizing damage, and ensuring sustainable agricultural practices.

Fig 3.1.1 shows the traditional methods of disease detection often rely on manual inspection,which is time-consuming, prone to error, and unsuitable for large-scale monitoring. Modern technological advancements have introduced automated systems that are faster, more accurate, and scalable.Timely detection of plant diseases can help in reducing crop losses, maintaining food security, and minimizing the overuse of chemical pesticides. By identifying issues early, farmers can take targeted actions to control disease spread, saving both time and resources.Moreover,accurate disease detection contributes to better crop health and improved yield quality.

3.2 Disadvantages of Existing Systems

Pei, M., Kong, M., Fu, M., Zhou, X., Li, Z., and Xu, J. [1] utilized unsupervised learning techniques for plant leaf pest and disease detection. They implemented clustering algorithms such as k-means and hierarchical clustering to identify patterns within plant disease symptoms without relying on labelled datasets.

However, the lack of labelled data makes it difficult to validate the accuracy of the clusters, and the model may struggle with differentiating similar disease symptoms.Reddy, J. N., Vinod, K., and Ajai, A. R. [2] conducted a comparative analysis of classification algorithms, including Support Vector Machines (SVM), Decision Trees, and Neural Networks. While CNNs outperformed traditional machine learning models, their implementation required large datasets and extensive computational power, making it less feasible for small-scale farmers with limited resources.

Shafik, W., Tufail, A., Liyanage, C. D. S., and Apong, R. A. A. H. M. [3] developed a specialized Convolutional Neural Network (CNN) model for plant pest detection and disease classification. Although their model achieved high accuracy, CNNs are prone to overfitting when trained on limited datasets, and their interpretability remains a challenge.

Türkoğlu, M., and Hanbay, D. [4] explored deep learning-based feature extraction techniques for plant disease detection. While deep feature extraction improved accuracy, it also increased the complexity of the model, requiring high-performance computing resources that may not be accessible to all users.

Prathima, K., Kanchan, R. G., Arekal, S., Shalini, A. N., and Mishra, G. [5] integrated machine learning with IoT-based monitoring systems for agricultural pest and disease detection. Despite providing real-time monitoring, the system relied heavily on network connectivity and sensor accuracy, which could be affected by environmental conditions.

Sameer, S., Niharika, B. D., Vasavi, S., Rohith, M., and Abhishek, V. R. [6] proposed an enhanced AlexNet model for plant disease classification. While transfer learning improved classification accuracy, the reliance on pre-trained models introduced biases, and the model's adaptability to new plant diseases was limited.

Rajesh, B., Vardhan, M. V. S., and Sujihelen, L. [7] applied Decision Tree-based models for plant leaf disease classification. Although Decision Trees offered simplicity and interpretability, they struggled with high-dimensional image data, leading to lower accuracy compared to deep learning models.Jia, W., Yang, N., Lu, Y., and Deng, P. [8] investigated data augmentation techniques to improve plant disease detection models. While data augmentation enhanced model robustness, it

Wang, Q., He, G., Li, F., and Zhang, H. [9] introduced a novel plant disease classification database. Although their dataset was well-curated, the generalization of models trained on this dataset to different environmental conditions remained a challenge.

Li, L., Zhang, S., and Wang, B. [10] provided a comprehensive review of deep learning-based plant disease detection models. Their analysis highlighted the computational expense of deep learning models and the need for more efficient architectures for real-time disease detection.

Demilie, W. B. [11] conducted a comparative study of deep learning models such as CNNs, ResNet, and EfficientNet for plant disease detection. While their study identified effective architectures, the trade-off between accuracy and computational efficiency posed a challenge for resource-constrained applications.

Sangeetha, T., and Mohanapriya, M. [12] explored hybrid machine learning and deep learning models for plant disease detection. Although these models improved classification accuracy, their increased complexity and higher training time made them difficult to deploy in real-world agricultural settings.

Chaitra, S., Ghana, S., Singh, S., and Poddar, P. [13] developed a deep learning model for image-based plant disease detection on edge devices. While their model was optimized for low-resource hardware, it still faced limitations in processing power and battery efficiency, affecting real-time performance.

Liu, J., and Wang, X. [14] reviewed advancements in deep learning for plant disease detection. Their study discussed key challenges such as the need for improved data collection techniques, real-time monitoring systems, and enhanced model interpretability. Despite progress, achieving accurate and explainable models remains a significant challenge in precision agriculture.

3.3 PROPOSED SYSTEM

The proposed model integrates Convolutional Neural Networks (CNNs), Long Short- Term Memory (LSTM) networks, and DenseNet201 to achieve high-accuracy plant disease detection. CNNs extract spatial features such as edges, textures, and colors from plant images, while DenseNet201 enhances feature propagation and gradient flow, ensuring efficient deep feature extraction with fewer parameters. The LSTM component is incorporated to capture temporal dependencies, allowing the

model to analyze disease progression over time. This hybrid architecture effectively balances feature extraction and sequential learning, making it well-suited for real-time plant disease detection.

The implementation begins with image preprocessing, where plant leaf images are resized using bilinear interpolation to fit the model's input dimensions. Data augmentation techniques, including flipping, rotation, and scaling, enhance model generalization and mitigate overfitting. The CNN component, based on pre-trained DenseNet201, extracts high-level spatial features, which are then flattened and passed to the LSTM network for sequential pattern learning. Finally, a dense layer with a softmax classifier is employed for classification, providing highly accurate disease predictions.

The training process employs the Adam optimizer with categorical cross-entropy loss, using a batch size of 32 and running for 50 epochs. Hyperparameter tuning includes dropout regularization (0.5) to prevent overfitting and learning rate adjustments to optimize convergence. The model outperforms traditional architectures, such as VGG19 and ResNet50, achieving a remarkable accuracy of 99.4% in plant disease classification. This superior performance demonstrates the effectiveness of the CNN-LSTM+DenseNet201 hybrid model in agricultural applications, making it a robust solution for real-time plant disease monitoring and management.

Advantages:

1. **High Detection Accuracy**
2. **Real-Time Detection Capabilities**
3. **Early Disease Identification**
4. **Scalability for Large Agricultural Areas**
5. **Cost-Effective Disease Management**
6. **Automation of Disease Monitoring**
7. **Reduced Dependency on Experts**
8. **Environmentally Friendly Practices**

The proposed model employs a multi-stage deep learning and classification approach for efficient and accurate plant pest and disease detection. The workflow consists of four main stages: Dataset Processing, Feature Extraction using Pre-trained Deep Learning Models, Classification using LSTM and Hybrid Classifiers, and Final Disease Detection.

The process depicted in the flowchart outlines the key stages of developing and deploying a hybrid machine learning model. In Fig 3.3.1 It begins with **Dataset Collection**, where data is gathered from various sources such as open datasets, APIs, sensors, or proprietary sources. This step ensures that the data is sufficient in size and relevance to meet the project's objectives. The next stage is **Data Preprocessing**, which involves cleaning and preparing the data by handling missing values, removing outliers, standardizing or normalizing data, encoding categorical variables, and splitting it into training, validation, and testing sets.

Following preprocessing, the **Feature Extraction** phase identifies and selects the most important attributes from the dataset to enhance model performance and efficiency. This can involve techniques such as dimensionality reduction, domain-specific feature engineering, or statistical feature selection.

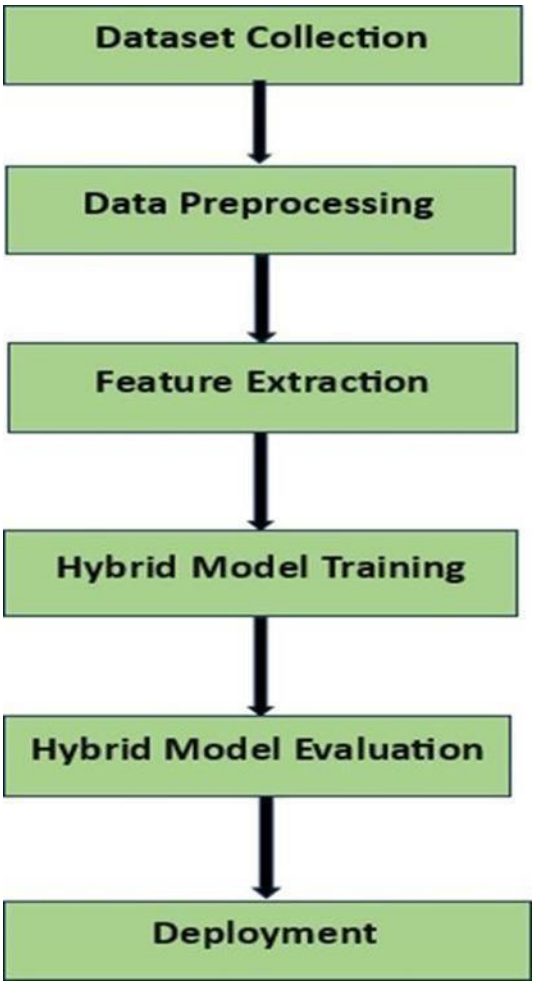


Fig 3.3.1 Flow Chart of Proposed System

Once the features are prepared, the **Hybrid Model Training** phase combines the strengths of multiple algorithms or architectures, such as machine learning and deep learning models, to achieve optimal performance. This step also includes hyperparameter optimization and the use of ensemble or meta-learning techniques.

After training, the model undergoes **Hybrid Model Evaluation** to ensure its effectiveness and generalization capabilities. Various metrics, such as accuracy, precision, recall, F1- score, or RMSE, are used to evaluate the model. Additionally, techniques like cross- validation, confusion matrices, and ROC curves help assess performance and compare the hybrid model with baseline models. Finally, the process concludes with **Deployment**, where the trained model is integrated into production systems via APIs. This step involves monitoring the model's real-world performance, implementing updates through CI/CD pipelines, and ensuring scalability and security for broader use.

This systematic approach ensures the development of a robust, efficient, and scalable hybrid machine learning model that meets the desired objectives.

3.4 Feasibility Study

➢ Introduction:

The early detection of plant diseases and pests is essential for improving agricultural productivity and preventing large-scale crop losses. Traditional manual inspection methods are time-consuming,subjective, and often impractical for large farmlands. To address these challenges, researchers have explored various machine learning (ML) and deep learning (DL) techniques for automating plant disease detection. This feasibility study evaluates the technical, operational, economic, and legal aspects of implementing an AI-based plant disease detection system, based on insights from recent research papers.

➢ Technical Feasibility:

Several studies have explored different machine learning and deep learning approaches for plant disease detection. Pei et al. [1] utilized unsupervised learning techniques, specifically k-means and hierarchical clustering, to identify patterns in plant disease symptoms without the need for labeled data. This method is beneficial for reducing dependency on manually annotated datasets. Reddy et al. [2] conducted a comparative analysis of classification models, including Support Vector Machines

(SVM), Decision Trees, and Neural Networks, and found that CNN- based deep learning models outperformed traditional ML techniques.

Shafik et al. [3] and Türkoğlu et al. [4] developed specialized CNN models for feature extraction and classification, demonstrating that deep learning significantly improves accuracy by capturing complex spatial patterns. Prathima et al. [5] introduced IoT-based monitoring systems integrated with machine learning, enabling real-time disease detection and prevention. Additionally, Sameer et al. [6] applied transfer learning techniques using a pre-trained AlexNet model, achieving high accuracy with limited data.

Data augmentation techniques, as demonstrated by Jia et al. [8], further improved the robustness of models by applying transformations such as rotation, flipping, and scaling. Sangeetha and Mohanapriya [12] explored hybrid models that combined traditional ML with deep learning for enhanced classification accuracy.

- **Operational Feasibility:**

To ensure smooth adoption, basic training sessions can be conducted for farmers to familiarize them with the technology. Agricultural officers and research institutions can play a key role in interpreting results and guiding interventions. IoT-based real-time monitoring, as proposed by Prathima et al. [5], enhances operational feasibility by continuously tracking plant health. Moreover, deploying AI models on mobile devices, as suggested by Chaitra et al. [13], allows small-scale farmers to access disease detection tools without requiring high-end computing infrastructure. Given these factors, the system is operationally feasible, provided it is designed to be user-friendly and accessible.

- **Economic Feasibility:**

Despite the initial investment, the long-term economic benefits outweigh the costs. Automated disease detection reduces crop losses by enabling early intervention, leading to increased agricultural yield. Furthermore, models based on CNNs and transfer learning, as demonstrated by Sameer et al. [6], achieve high accuracy with minimal training data, reducing the need for extensive manual annotation. IoT-based monitoring systems, though requiring an initial setup cost, provide continuous surveillance, preventing large-scale infestations and minimizing pesticide misuse.

4. SYSTEM REQUIREMENTS

4.1 Hardware Requirements:

- Processor : Intel(R)Core(TM)i31005G1CPU@1.20GHz
- System Type : 64-bit operating system, x64-based processor
- Cache memory : 4MB(Megabyte)
- RAM : 8GB (gigabyte)

To achieve optimal performance, the hardware setup must align with the computational demands of deep learning model training and inference. The system requires a processor, sufficient RAM, and access to cloud-based GPU services to accelerate training and classification tasks.

For the development environment, the user has utilized an **Intel Core i3 processor**, **Windows 11 operating system**, and **Google Colab Pro** for training and inference. Google Colab provides free access to GPUs, making it an efficient platform for training deep learning models without requiring high-end local hardware.

For optimal performance in local execution, a system with an Intel Core i5 (10th Gen or later) or AMD Ryzen 5 (3000 series or later) processor, 16GB DDR4 RAM, and an NVIDIA GTX 1650 GPU with 4GB VRAM is recommended.A storage configuration of a 256GB SSD and a 1TB HDD ensures smooth data processing and storage management.A stable internet connection is necessary for downloading datasets and utilizing cloud-based training resources.

4.2 Software Requirements:

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

The system requires a stable and efficient operating environment to support deep learning frameworks and machine learning libraries. It is compatible with Windows 10/11 (64-bit) and Ubuntu 20.04+ (preferred for deep learning model deployment). The programming languages used for development include Python 3.8 or later, which provides extensive support for deep learning and machine learning libraries. Shell scripting is also utilized for automation and deployment purposes.

The system relies on various libraries and frameworks to implement machine learning and deep learning models. TensorFlow 2.x and Keras are the primary frameworks for training and deploying deep learning models. PyTorch 1.10+ can be used as an alternative deep learning framework. Scikit-Learn supports machine learning utilities, including the Extreme Learning Machine (ELM) classifier. OpenCV 4.x is integrated for image preprocessing, while NumPy and Pandas handle data manipulation and processing.

To enable GPU acceleration, the system requires CUDA Toolkit and cuDNN, ensuring compatibility with TensorFlow and PyTorch. For web-based deployment, Flask and FastAPI facilitate API-based model deployment. Docker is used for containerization, making the system scalable and easy to deploy across different environments. Version control is managed using GitHub or GitLab to track code changes and facilitate collaboration.

4.3 Software Description

The development environment is tailored for deep learning research and implementation. **Google Colab Pro** is the primary platform used for interactive development and experimentation. Jupyter Notebook is also an alternative option for running code in a notebook environment. PyCharm and VS Code serve as integrated development environments (IDEs) for coding and debugging.

For cloud-based training and deployment, the system supports platforms like **Google Colab Pro**, AWS EC2 (p3/p4 instances), Google Vertex AI, and Azure. These cloud platforms provide high-performance GPU resources, reducing the time required for model training. Model deployment is facilitated using TensorFlow Serving and ONNX Runtime, ensuring efficient and optimized inference. Databases such as PostgreSQL and MongoDB are used to store results and metadata, enabling efficient data management and retrieval.

➢ Implementation Details

- **Google Colab**: Used as the primary development platform with free access to GPUs. Supports Python, TensorFlow, and Keras.
- **Python**: The core programming language used for data preprocessing, model training, and deployment.
- **TensorFlow**: The primary deep learning framework used for building and training models.
- **Keras**: A high-level API running on top of TensorFlow to simplify model creation and training.
- **OpenCV**: Used for image preprocessing tasks like resizing, augmentation, and normalization.
- **NumPy & Pandas**: Used for data manipulation, handling arrays, and dataset processing.
- **Flask**: Used for developing RESTful APIs for model deployment and integration.

5. System Design

5.1 Architecture

The model architecture integrates multiple components, with CNN extracting spatial features, LSTM capturing temporal disease progression patterns, and DenseNet201 improving feature reuse and gradient flow. The training process includes hyperparameter optimization with Adam optimizer, categorical cross-entropy loss, and data augmentation to prevent overfitting. The performance of the hybrid model is evaluated using metrics like precision, recall, F1-score, and accuracy.

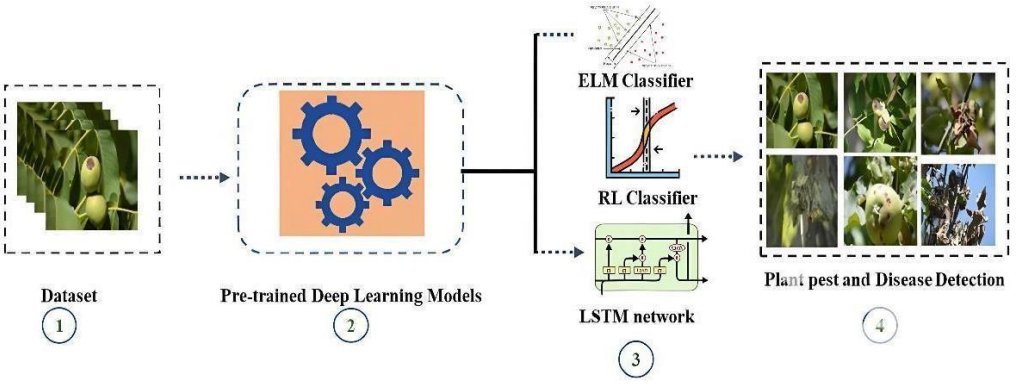


Fig 5.1.1 Model Architecture

Fig 5.1.1 illustrates a **plant pest and disease detection framework** using deep learning and machine learning classifiers. The process begins with a **dataset**, which consists of images of plants affected by various diseases or pests. These images are preprocessed and used for feature extraction through **pre-trained deep learning models**, such as CNN-based architectures. These models extract essential features from the input images, helping in identifying patterns related to plant diseases and pest infestations.

In the next step, the extracted features are passed to different classifiers, including **Extreme Learning Machine (ELM) Classifier, Reinforcement Learning (RL) Classifier, and LSTM Network**. These classifiers analyze the extracted features and make predictions based on the trained model. Finally, the system provides results for **plant pest and disease detection**, showcasing the identified diseases or pests affecting the plants. This approach ensures an efficient and accurate diagnosis of plant health, enabling early intervention to prevent crop damage.

This approach underscores the effectiveness of hybrid architectures in addressing complex problems in plant disease detection. The results demonstrate that the

CNN-LSTM+DenseNet201 model significantly improves detection accuracy while offering scalability, robustness, and adaptability to diverse agricultural conditions. The project provides a reliable and efficient solution, paving the way for advancements in precision agriculture through real- time plant health .

➢ Dataset Description

The project describes the implementation of a hybrid deep learning model for plant disease detection, combining CNN, LSTM, and DenseNet201 to maximize detection accuracy and efficiency. Table 5.1.2 shows the **Dataset Description** involves preprocessing the dataset of 4,447 images by resizing, noise removal, and data augmentation techniques like flipping, rotation, and scaling to ensure robustness and generalizability. These steps standardize the input and enhance the model's capability to handle real-world variations in image data.

LABEL	LABEL NAME	TOTAL SAMPLES
(1)	Apple_Aphis_spp	162
(2)	Apple_Eriosoma_lanigerum	366
(3)	Apple_Monilia_laxa	255
(4)	Apple_Venturia_inaequalis	633
(5)	Apricot_Coryneum_beijerinckii	1100
(6)	Apricot_Monilia_laxa	85
(7)	Cancer_symptoms	76
(8)	Cherry_Aphis_spp	356
(9)	Drying_symptoms	139
(10)	Peach_Monilia_laxa	314
(11)	Peach_Parthenolecanium_corni	427
(12)	Pear_Erwinia_amylovora	215
(13)	Plum_Aphis_spp	70
(14)	Walnut_Eriophyes_erineus	69
(15)	Walnut_Gnomonia_leptostyla	180
	Total samples	**4447**

Table 5.1.2 Dataset Description

➢ Data Pre-Processing :

The preprocessing of a dataset usually consists of resizing images and preparing them for feature extraction. Fig 5.1.3 shows the Plant leaf images are resized to appropriate dimension requirements for input to such a model as part of image preprocessing for CNN models.Bilinear interpolation is used to resize images, and

dimensions in models,such as 224 pixels in ResNet or 227 pixels in GoogleNet.Appropriate resizing holds the consistency and enhances the model's performance and efficiency.Images are generated after preprocessing.The images will go through a series of transformations,comprising the extraction of deep features from the connected layers of pre- trained CNN models.These features will be utilized by either the LSTM layer or the LR/ELM classifiers in the classification of plant pests and diseases.Pre-processing was done to also remove any form of noise in the images,as this increases accuracy in both detection and classification processes.This ensures that models of deep learning can work with high efficiency across different classifiers and produces excellent accuracy.

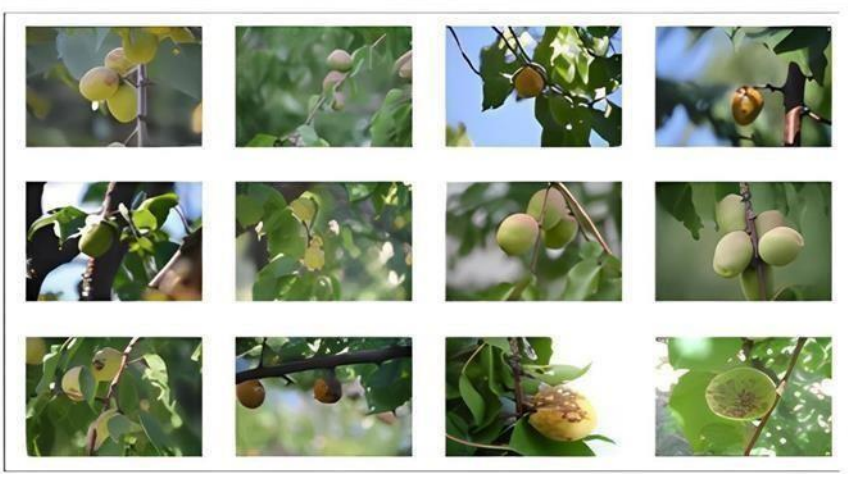


Fig 5.1.3 Data Pre-processing

➢ Feature Extraction:

Feature extraction is a critical step in the process of building machine learning models, particularly in image-based applications like plant disease detection. It involves identifying and isolating relevant characteristics or patterns from raw data that are essential for classification or prediction tasks. In the context of plant disease detection, feature extraction focuses on spatial and temporal attributes of plant images to accurately differentiate between healthy and diseased plants.

➢ **Input Data Preparation**:

- Collect high-resolution images of plant leaves or crops, ensuring diverse representations of healthy and diseased conditions.
- Preprocess the images to standardize input dimensions (e.g., 224×224 pixels) and remove noise. Data augmentation techniques like flipping,

are applied to increase dataset variability.

- **Feature Extraction Using Convolutional Neural Networks (CNNs)**:
  - Pass the preprocessed images through multiple convolutional layers to extract low- level spatial features such as edges, textures, and colors.
  - Pooling layers reduce the dimensionality of feature maps, retaining essential information while reducing computational complexity.
  - Use deeper CNN layers, like those in DenseNet201, to capture high-level features representing complex patterns and structures.

- **Flattening Features**:

The extracted spatial features are flattened into a one-dimensional vector, making them compatible for input into sequential models.Combine spatial features extracted by CNN with temporal features captured by LSTM.Feed these combined features into a fully connected dense layer for final processing.

- **Classification and Prediction**:

The final feature vector is passed to a classifier (e.g., softmax layer) to predict the class of the input image (e.g., healthy or diseased, specific type of disease).

5.2 Model building :

The **hybrid CNN-LSTM+DenseNet201 architecture** for plant disease detection is a robust and efficient solution designed to address challenges in agricultural diagnostics. The process begins with collecting and preprocessing plant leaf images, which involves resizing, noise removal, and applying data augmentation techniques such as flipping, rotation, and scaling to enhance model generalization. The images are then passed through the **CNN component**, which extracts spatial features like edges, textures, and colors using convolutional and pooling layers. These features are further refined using **DenseNet201**, which ensures efficient feature reuse and gradient flow, reducing computational overhead.

Next, the extracted spatial features are flattened and fed into the **LSTM component**, which captures temporal dependencies and disease progression patterns, particularly useful for detecting diseases with evolving symptoms. The model integrates these spatial and temporal features using a dense layer with a softmax classifier to produce the final result.

training process optimizes performance through techniques such as Adam optimization, categorical cross-entropy loss, and dropout regularization to prevent overfitting.

This hybrid approach demonstrates exceptional accuracy, achieving **99.4%**, surpassing traditional models like VGG16, VGG19, and ResNet50. It combines the strengths of spatial and temporal learning, ensuring robustness across diverse environmental conditions and scalability for various crops and diseases. The integration of DenseNet201 further enhances precision by improving feature propagation and computational efficiency. The model's performance metrics, including precision, recall, F1-score, and accuracy, validate its effectiveness, making it a valuable tool for real-time plant disease detection and precision agriculture.

- **Hybrid Model Architecture**

The proposed architecture combines three major components of the model: CNN, namely the Convolutional Neural Networks, Long Short-Term Memory networks(LSTM),and DenseNet201.This CNN component leads to spatial feature processes, for example, edges, textures, and even colors in the images of plants.DenseNet201 is a densely connected convolutional network. It enhances deeper features while using fewer parameters, thereby improving feature propagation and gradient flow. Incorporating the LSTM layer has a strong motivation to seize temporal dependencies and sequential patterns within the data, which are CNN, LSTM with DenseNet201 for Plant Disease Detection strongly required for the identification of diseases that may have progression over time. That way, both the spatial and temporal features will be modeled appropriately.

- **CNN Component:**

To extract the spatial features, CNN layers are used, which are pre-trained on the dataset ImageNet. These feature extraction layers include convolutional layers and then pooling layers in DenseNet201,it connects the layers efficiently to overcome gradient vanishing and feature reuse.

- **LSTM Component:**

The input to LSTM includes flattened spatial features. LSTM network evaluates sequential data that contain time-series information about the spread of the disease.

- **Ensemble Classifier:**

The output from the LSTM is passed to a dense layer with softmax classifier for final classification. This ensemble classifier integrates spatial features of CNN and sequential learning from LSTM that leads to highly accurate detection of plant diseases.

- **Training Process:**

The model was trained through supervised learning on the given dataset of images of plant disease. The training process involves the steps: All input images were resized to the required dimensions of 224x224 for DenseNet201.Data augmentation in the form of flipping, rotation, and scaling of the input data is applied so that overfitting is prevented and generalization of the model is enhanced to various environmental conditions.

The **Convolutional Neural Network (CNN)** is used to extract spatial features from plant leaf images. CNN layers detect patterns such as edges, textures, and disease-specific characteristics, making them effective for image classification tasks. The model is trained with multiple convolutional layers, activation functions, and pooling layers to enhance feature learning.

The **Long Short-Term Memory (LSTM)** model is integrated into the system to analyze sequential dependencies in extracted features. LSTM networks are beneficial for learning patterns over time and improving classification accuracy, especially when dealing with feature sequences extracted from CNN layers.

The **DenseNet201 model** is employed as a feature extractor due to its densely connected convolutional layers, which allow efficient gradient flow and enhanced feature propagation. DenseNet201 extracts rich hierarchical features, significantly improving the accuracy of disease classification by capturing fine-grained patterns in plant images. This deep learning model ensures robust performance, enabling precise disease detection in real-time agricultural applications.

5.2.1 Model Performance Comparision

Table 5.2.1.1 compares accuracies of different classification models of plant disease detection by utilizing LSTM, SVM, and ELM classifiers. For both VGG16 and VGG19, accuracy values were 96.6% and 96.7%, respectively. Then, it was discovered that DenseNet201 outperformed both VGG16 and VGG19 with 98.4% for the LSTM classifier, 97.5% for the SVM classifier, and 98.2% for the ELM classifier. In all of these scores, GoogleNet had obtained an accuracy of 97.6%. The Xception model had a score of 98.6%, and for the ResNet50, the accuracy was 98.3%. The proposed CNN-LSTM + DenseNet201 model performs with 99.4% accuracy for the CNN-LSTM with DenseNet201, 98.2% accuracy for SVM, and 98.6% accuracy for ELM for real-time detection of the disease in the plant.

MODEL	LSTM	SVM	ELM
VGG16	96.6	96.2	96.5
VGG19	96.7	96.3	96.4
DENSENET201	98.4	97.5	98.2
GOOGLENET	97.6	96.2	97.4
XCEPTION	98.6	98.3	98.2
RESNET50	98.3	98.2	98.1
(PROPOSED) CNN-LSTM+DENSENET201	**99.4**	98.2	98.6

Table 5.2.1.1 Model Performance Comparision

Fig 5.2.1.2 shows the comparison of the accuracies of various models is carried out, namely VGG16, VGG19, DenseNet201, GoogleNet, Xception, and ResNet50+CNN-DenseNet201 among the three classifiers, namely LSTM, SVM, and ELM. All models show that they have superior accuracy. CNN- LSTM+DenseNet201 outperforms all the models presented herein. LSTM always gives accurate outputs in all the models. Advanced architectures such as DenseNet201 and Xception take one step further toward upscoping accuracy in plant disease detection, and this clarifies the importance of the proposed model.

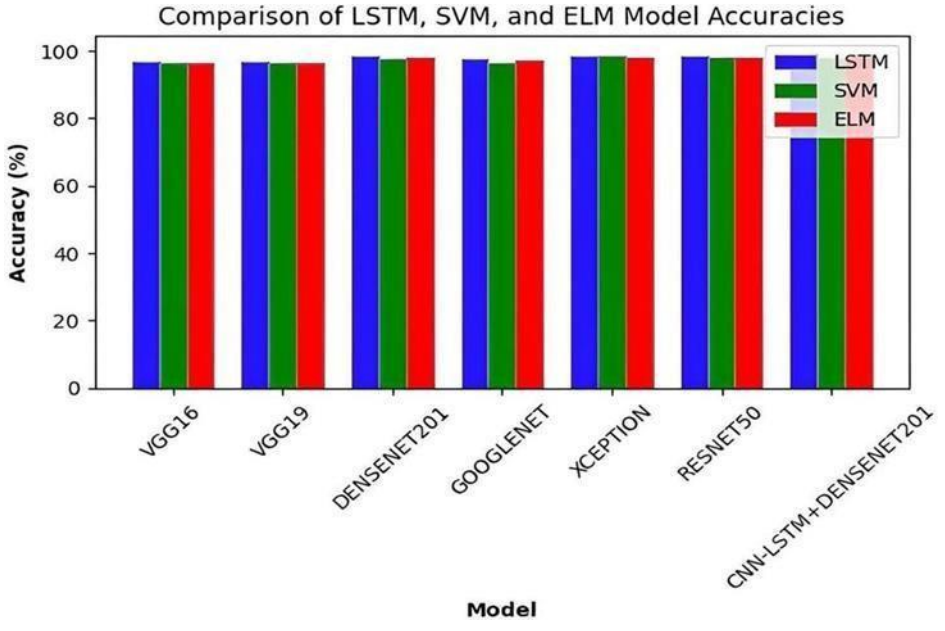


Fig 5.2.1.2 Comparisions of Model Architecture

5.2.2 Training And Testing Performance

Fig 6.6.2.1 shows us the highest accuracy was achieved using CNN-DenseNet201 by all the classifiers and LSTM achieved a very high training accuracy of 99.4% and testing accuracy of 98.9%.SVM with ELM and CNN-DenseNet201 together did well with training/testingaccuracies of 98.8%/98.3% and 98.6%/98.1%, respectively.DenseNet201 and Xception were closely following and LSTM managed to achieve 98.4% and 98.6% training accuracies and test time accuracies of 97.9% and 98.1%.LSTM well outperformed with the highest difference in SVMand ELM across the models;the best overall accuracy achieved was on CNN- DenseNet201.

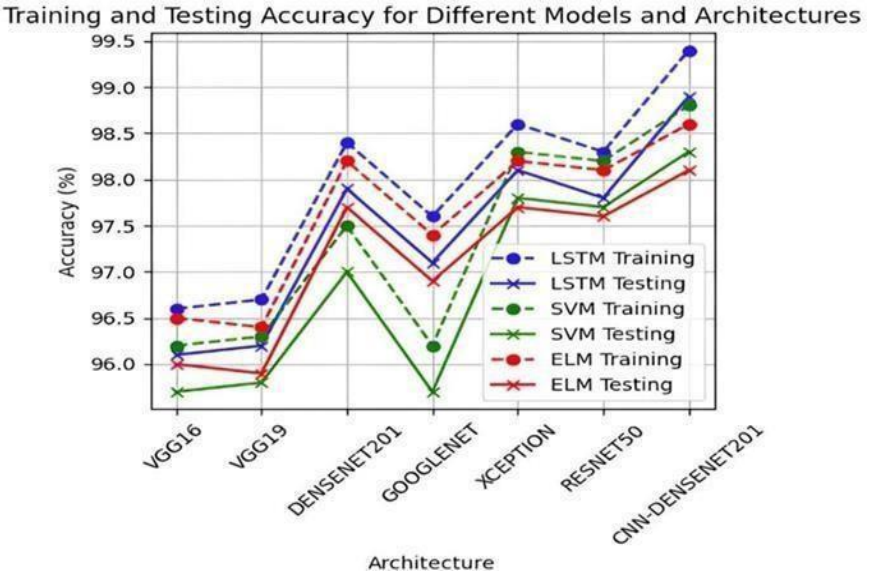


Fig 5.2.2.1 Training and Testing Accuracy

5.2.3 Model Summary

Table 5.2.3.1 is showing comparisons of the deep learning models considering the number of their parameters and input size.VGG16 has 318M parameters,VGG19 has 144M,and use 224×224×3 images.DENSENET201 has 20M parameters and GOOGLENET has 7M.XCEPTION uses 22.9M parameters with a large input size of 299×299×3. RESNET50 has 25.6M parameters and MOBILENET is lightweight having 4M parameters and developed for mobile usage.This compares model complexity and efficiency and balances each other.

This comparative analysis of parameters and input sizes highlights the trade-offs between model complexity and efficiency. While models like VGG16 and VGG19 are powerful for high- resolution feature extraction, they demand significant computational

resources. Models like DenseNet201, GoogleNet, and MobileNet offer a more balanced approach, focusing on efficiency without significant loss in accuracy. These differences allow for selecting the appropriate model based on specific use cases, computational constraints, and application needs.

Model	Parameters (Million)	Input Size
VGG16	318	$224 \times 224 \times 3$
VGG19	144	$224 \times 224 \times 3$
DenseNet201	20	$224 \times 224 \times 3$
GoogleNet	7.0	$224 \times 224 \times 3$
Xception	22.9	$299 \times 299 \times 3$
ResNet50	25.6	$224 \times 224 \times 3$

Table 5.2.3.1 Comparision of Model Parameters

5.2.4 Evaluation Metrics

Table 5.2.4.1 compares the performance of the models for different architectures,such as some models follows VGG16,DENSENET201 using LSTM,SVM and ELM-classifier using precision,recall,F1 score.The proposed CNN-LSTM+DENSENET201 model is the CNN, LSTM with DenseNet201 for Plant Disease Detection best,achieving the highest scores of all metrics 99% with all in comparison to other models.

MODEL	LSTM			SVM			ELM		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
VGG16	0.96	0.95	0.95	0.96	0.95	0.95	0.97	0.96	0.96
VGG19	0.97	0.96	0.96	0.97	0.96	0.96	0.96	0.95	0.95
DENSENET201	0.98	0.97	0.97	0.98	0.97	0.97	0.98	0.97	0.97
GOOGLENET	0.97	0.96	0.96	0.96	0.95	0.95	0.97	0.96	0.96
XCEPTION	0.98	0.97	0.97	0.98	0.97	0.97	0.97	0.97	0.97
RESNET50	0.98	0.97	0.97	0.98	0.97	0.97	0.98	0.97	0.97
InceptionV3	0.97	0.96	0.96	0.97	0.96	0.96	0.97	0.96	0.96
MobileNetV2	0.95	0.94	0.94	0.95	0.94	0.94	0.96	0.95	0.95
(Proposed) CNN-LSTM+DENSENET201	**0.99**	**0.98**	**0.98**	**0.99**	**0.98**	**0.98**	**0.99**	**0.98**	**0.99**

Table 5.2.4.1 Evaluation Metrics

5.3 Uml Diagrams

- **Uml Diagram for LSTM**

Fig represents a neural network architecture combining fully connected layers and Long Short-Term Memory (LSTM) layers for sequential data processing. The architecture consists of multiple components, starting with an Input Layer, which takes in either 36 or 24 input features. These inputs are then processed by a Fully Connected Layer with 512 neurons, which helps in learning complex feature representations before passing the data to the LSTM layers.

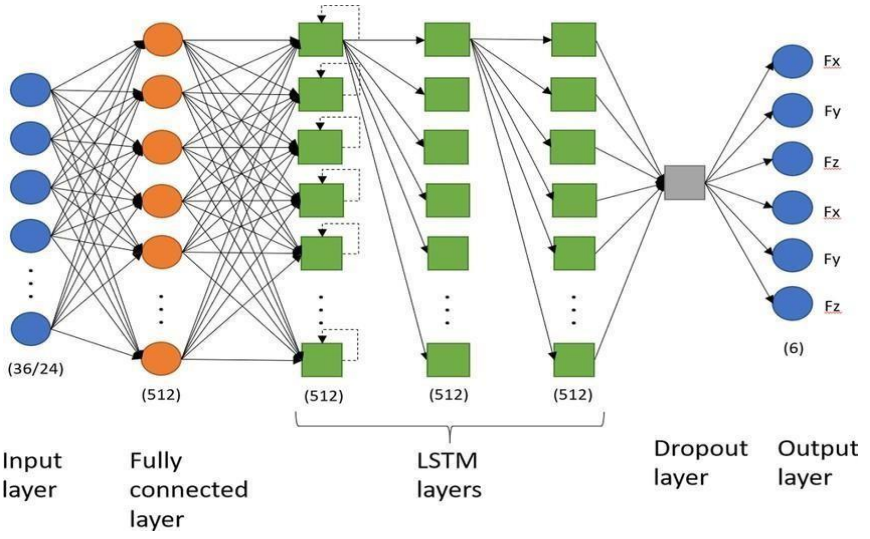


Fig 5.3.1 Uml Diagram for LSTM

The LSTM Layers consist of two stacked layers, each containing 512 LSTM units. These layers are responsible for capturing temporal dependencies in the data, making the model suitable for sequential or time-series tasks. Following the LSTM layers, a Dropout Layer is

applied to prevent overfitting by randomly deactivating some neurons during training.

- **Uml Diagram for CNN**

The given diagram illustrates a structured pipeline for training a Convolutional Neural Network (CNN) using a dataset that is first divided into a Training Set and a Test Set. Both sets undergo Data Pre-Processing, which ensures images are properly resized, normalized, and formatted for input into the model. To enhance the robustness of the model, Data Augmentation techniques such as rotation, width and height shifts, rescaling,

shearing, zooming, horizontal flipping, and fill mode adjustments are applied to the training data. This helps in improving generalization and reducing overfitting.

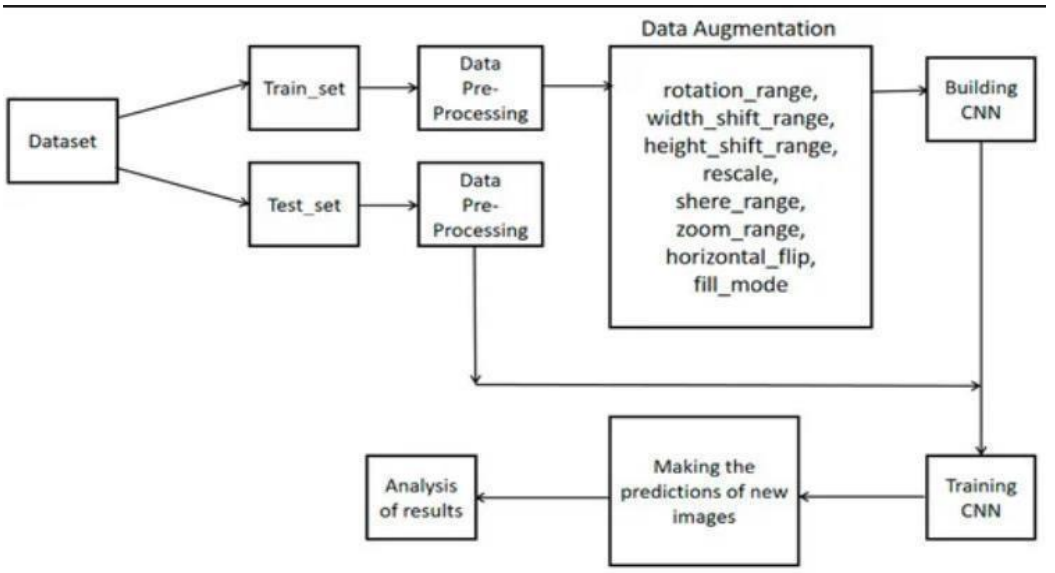


Fig 5.3.2 Uml Diagram for CNN

After preprocessing, the CNN Model is Built, defining layers for feature extraction and classification. The model is then trained using the augmented dataset to optimize its parameters. Once trained, the model is used to Make Predictions on New Images, evaluating its performance on the test set. Finally, an Analysis of Results is conducted to assess the model's accuracy and effectiveness. This phase helps in identifying necessary improvements, such as fine-tuning hyperparameters or modifying preprocessing strategies, ensuring a well-optimized CNN model for image classification.

6 .IMPLEMENTATION

Libraries:

Import numpy as np import pandas as pd
import matplotlib.pyplot as plt import seaborn as
sns import warnings from sklearn.svm
import SVC import statsmodels.api as sm
import LabelEncoder from sklearn.preprocessing
import RobustScaler from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression from sklearn.neighbors import
KNeighborsClassifier from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier from sklearn
import tree from sklearn import svm from sklearn.svm import SVC
import tensorflow as tf import matplotlib.pyplot as plt import os import
numpy as np
from sklearn.model_selection import train_test_split from sklearn.preprocessing import
LabelEncoder from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten,
TimeDistributed GlobalAveragePooling2D

fromtensorflow.keras.layers import LSTM

fromtensorflow.keras.utils import to_categorical

from tensorflow.keras.applications import DenseNet201 from tensorflow.keras.callbacks
import ModelCheckpoint
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn

import cv2

from sklearn.ensemble import GradientBoostingClassifier from tensorflow.keras.utils
import to_categorical
from tensorflow.keras.applications import DenseNet201 from tensorflow.keras.callbacks
import ModelCheckpoint

```
from tensorflow.keras.regularizers import l1

import pandas as pd
dataset_path = '/content/drive/MyDrive/Project Details(DL)/Turkey-Apple-Disease-Dataset- main'

!ls "/content/drive/MyDrive/Project Details(DL)/Turkey-Apple-Disease-Dataset-main" def preprocess_images(dataset_path):
"""
Performs preprocessing on images in the given dataset path.

Args:
dataset_path: Path to the dataset directory.

Returns:
   No
ne
"""

for class_folder in os.listdir(dataset_path):
class_path = os.path.join(dataset_path,
class_folder) if os.path.isdir(class_path):
for image_file in os.listdir(class_path):
image_path = os.path.join(class_path,
image_file) try:
# Load image
image = Image.open(image_path)

# Resize image (adjust size as needed)
image = image.resize((224, 224))

# Convert to numpy array and normalize pixel values
image_array = np.array(image) / 255.0

# Perform other preprocessing steps as needed (e.g., data augmentation)

# Save preprocessed image (optional)
# ...

except Exception as e:
print(f"Error processing image {image_path}: {e}")

# Call the preprocessing function
preprocess_images(dataset_path)
Define the augmentation
sequence seq =
iaa.Sequential([
iaa.Fliplr(0.5), # horizontal flip with 50% probability
iaa.Flipud(0.5), # vertical flip with 50% probability
```

```
iaa.Affine(rotate=(-15, 15)),  # random rotation between -15 and 15 degrees
iaa.AdditiveGaussianNoise(scale=0.05),  # add Gaussian noise with standard deviation of
5%
])

# Call the augmentation function
dataset_path = "/content/drive/MyDrive/Project Details(DL)/Turkey-Apple-Disease-
Dataset- main"
augment_images(dataset_pat
h) import matplotlib.pyplot
as plt

# Get the path to the 'Apricot Coryneum beijerinckii' folder
apricot_coryneum_path = os.path.join(dataset_path, 'Apricot Coryneum beijerinckii')

# Get a list of the first 12 image files in the
folder image_files =
os.listdir(apricot_coryneum_path)[:12]

# Create a figure with 3 rows and 4
columns fig, axes = plt.subplots(3, 4,
figsize=(5, 4))

# Iterate through the first 12 images and plot
them for i, image_file in
enumerate(image_files):
image_path = os.path.join(apricot_coryneum_path, image_file)

# Load the image
image = Image.open(image_path)

# Convert to numpy array
image_array =
np.array(image)

# Plot the image
ax = axes[i // 4, i % 4]
ax.imshow(image_array)
ax.axis('off')
# Show the plot
plt.show()
# prompt: perform feature extraction using vgg16 from fc1, fc2, and predictions layers
on the Turkey-Apple-Disease-Dataset-main folder

# Install necessary libraries
!pip install tensorflow tqdm
```

```
        print(f"Error extracting features from {img_path}:
{e}") # Convert lists to numpy arrays
features_array = {layer_name: np.array(features_list[layer_name]) for layer_name
in layer_names}
labels_array = np.array(labels_list)

# Save the features and labels to
disk for layer_name in
layer_names:
   np.save(f'features_{layer_name}_vgg16.npy',
features_array[layer_name]) np.save('labels.npy', labels_array)
print("Feature extraction completed and saved to
disk.") # Save the model to disk
model.save('vgg16_feature_extraction_model.h5')
print("Model saved as vgg16_feature_extraction_model.h5")

def split_dataset(dataset_path, test_size=0.2):
"""
Splits the dataset into training and testing
sets. Args:
dataset_path: Path to the dataset directory.
  test_size: Proportion of the dataset to include in the test split.
Returns:
A tuple containing:
- X_train: List of training image file paths.
- X_test: List of testing image file paths.
- y_train: List of training labels.
   -            y_test: List of
testing labels. """
X = []
y = []
for class_folder in os.listdir(dataset_path):
class_path = os.path.join(dataset_path,
class_folder) if os.path.isdir(class_path):
for image_file in os.listdir(class_path):
image_path = os.path.join(class_path,
image_file) X.append(image_path)
y.append(class_folder)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42)
return X_train, X_test, y_train, y_test
# Split the dataset
X_train, X_test, y_train, y_test =
split_dataset(dataset_path) # Print the sizes of the
splits
print("Training set size:", len(X_train))
print("Testing set size:", len(X_test))
# Load the extracted features and labels
```

```
label_encoder = LabelEncoder()
labels_encoded =
label_encoder.fit_transform(labels)
labels_categorical =
to_categorical(labels_encoded)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
features_fc1, labels_categorical, test_size=0.2, random_state=42
)
# Reshape input for LSTM (samples, timesteps, features)
# Assuming each feature vector represents a single timestep
X_train = np.reshape(X_train, (X_train.shape[0], 1,
X_train.shape[1])) X_test = np.reshape(X_test, (X_test.shape[0],
1, X_test.shape[1]))

# Build LSTM
model model =
Sequential()
model.add(LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(y_train.shape[1], activation='softmax'))
# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy']) # Train the model
model.fit(X_train, y_train, epochs=30, batch_size=32, validation_data=(X_test,
y_test)) # Save the model
model.save('lstm_model.h5')
print("Model saved
successfully.") # Evaluate
the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy using adam optimizer: {accuracy *
100:.2f}%") # Print the shapes of the resulting sets
print("Training data shape:", X_train.shape,
y_train.shape) print("Testing data shape:",
X_test.shape, y_test.shape)# Load the extracted
features and labels features_fc1 =
np.load('features_fc1_vgg19.npy') labels =
np.load('labels.npy')

# Import necessary modules
from sklearn.preprocessing import
LabelEncoder from sklearn.model_selection
import train_test_split from
tensorflow.keras.utils import to_categorical
# Encode labels
```

```
# Reshape input for LSTM (samples, timesteps, features)
# Assuming each feature vector represents a single timestep
X_train = np.reshape(X_train, (X_train.shape[0], 1,
X_train.shape[1])) X_test = np.reshape(X_test, (X_test.shape[0],
1, X_test.shape[1]))

# Build LSTM model
from tensorflow.keras.models import
Sequential from tensorflow.keras.layers
import LSTM, Dense model = Sequential()
model.add(LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(y_train.shape[1], activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model and save the training history
history = model.fit(X_train, y_train, epochs=30, batch_size=32,
validation_data=(X_test, y_test))

# Save the model
model.save('lstm_model_vgg19.h5')
print("Model saved successfully.")

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy * 100:.2f}%")

# Plotting training & validation accuracy and
loss plt.figure(figsize=(12, 4))

# Accuracy plot
plt.subplot(1, 2,
1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch
s')
plt.ylabel('Accur
acy') plt.legend()

# Loss plot
plt.subplot(1, 2,
2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation
Loss') plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

```
# Convert predictions and true labels back to their original form
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Calculate F1 score
f1 = f1_score(y_true, y_pred_classes,
average='weighted') print(f"F1 Score: {f1:.4f}")

# Calculate confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred_classes)

# Calculate specificity and sensitivity for each
class specificity = []
sensitivity = []

for i in range(conf_matrix.shape[0]):
tn = np.sum(np.delete(np.delete(conf_matrix, i, axis=0), i,
axis=1)) fp = np.sum(np.delete(conf_matrix[:, i], i))
fn = np.sum(np.delete(conf_matrix[i,
:], i)) tp = conf_matrix[i, i]

specificity.append(tn / (tn + fp))
sensitivity.append(tp / (tp + fn))

# Print specificity and
sensitivity for i in
range(len(specificity)):
print(f"Class {i} - Specificity: {specificity[i]:.4f}, Sensitivity: {sensitivity[i]:.4f}")

# Load the extracted features and labels
features_fc1 =
np.load('features_fc1_vgg19.npy') labels =
np.load('labels.npy')

# Encode labels
label_encoder = LabelEncoder()
labels_encoded = label_encoder.fit_transform(labels)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_fc1, labels_encoded, test_size=0.2,
random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define parameter grid for
GridSearchCV param_grid = {
'C': [0.1, 1, 10, 100],
'kernel': ['linear', 'rbf'],
'gamma': ['scale', 'auto']
}
```

```
grid_search = GridSearchCV(estimator=SVC(), param_grid=param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train_scaled,  y_train)

# Get the best parameters and best
score best_params =
grid_search.best_params_ best_score =
grid_search.best_score_ print(f"Best
Parameters: {best_params}")
print(f"Best Cross-validation Accuracy: {best_score * 100:.2f}%")

# Train SVM with best
parameters best_clf =
grid_search.best_estimator_
best_clf.fit(X_train_scaled,
y_train)

# Evaluate the model
y_pred =
best_clf.predict(X_test_scaled)
accuracy = accuracy_score(y_test,
y_pred)
print(f"SVM Test Accuracy: {accuracy * 100:.2f}%")

# Load pre-trained ResNet50 model (excluding top classification layers)
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Add global average pooling layer to the
model x = base_model.output
x = GlobalAveragePooling2D()(x)
model = Model(inputs=base_model.input, outputs=x)

def
extract_features(image_pat
h): """
Extracts features from a single image using the pre-trained model.

Args:
image_path: Path to the image file.

Returns:
    A numpy array of extracted
features. """
img = image.load_img(image_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)  # Preprocess input as per the model's requirement

features = model.predict(img_array)
features = features.flatten() # Flatten the feature
map return features
# Split data into train and test sets
 X_train_resnet50, X_test_resnet50, y_train, y_test = train_test_split(
features_array_resnet50, labels_array_resnet50, test_size=0.2, random_state=42
)
```

```
accuracy_resnet50 = accuracy_score(y_test, y_pred_resnet50)

print("Test Accuracy (ResNet50 features with SVM classifier):",
accuracy_resnet50) # Load pre-trained DenseNet201 model (excluding top
classification layers)
base_model = DenseNet201(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

# Add global average pooling layer to the
model x = base_model.output
x = GlobalAveragePooling2D()(x)
model = Model(inputs=base_model.input, outputs=x)

def
extract_features(image_pat
h): """
Extracts features from a single image using the pre-trained model.

Args:
image_path: Path to the image file.

Returns:
    A numpy array of extracted
features. """
img = image.load_img(image_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)  # Preprocess input as per the model's requirement

features = model.predict(img_array)
features = features.flatten()  # Flatten the feature
map return features

# Path to the dataset
dataset_path = '/content/drive/MyDrive/Project Details(DL)/Turkey-Apple-Disease-
Dataset- main'

# Initialize lists for features and labels
features_list = []
labels_list = []

# Iterate through the dataset and extract features for each
image for class_folder in os.listdir(dataset_path):
class_path = os.path.join(dataset_path,
class_folder) if os.path.isdir(class_path):
for image_file in os.listdir(class_path):
image_path = os.path.join(class_path, image_file) try:
features = extract_features(image_path)
features_list.append(features)
labels_list.append(class_folder)
except Exception as e:
print(f"Error extracting features from {image_path}: {e}")# Convert features and labels to
numpy arrays features_array_densenet201 = np.array(features_list)
```

```
labels_array_densenet201 = np.array(labels_list)

print("Feature extraction completed and saved to
disk.") # Save the model to disk
model.save('densenet201.h5')
print("densenet201.h5")

# Encode labels
label_encoder = LabelEncoder()
labels_encoded =
label_encoder.fit_transform(labels_array_densenet201)  labels_encoded
= to_categorical(labels_encoded)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(features_array_densenet201,
labels_encoded, test_size=0.2, random_state=42)

# Reshape features to fit LSTM input requirements (samples, timesteps, features)
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

# Build LSTM model
lstm_model =
Sequential()
lstm_model.add(LSTM(256, input_shape=(1, X_train.shape[2]), return_sequences=True))
lstm_model.add(Dropout(0.5))
lstm_model.add(LSTM(128,
return_sequences=False))
lstm_model.add(Dropout(0.5))
lstm_model.add(Dense(256, activation='relu'))
lstm_model.add(Dropout(0.5))
lstm_model.add(Dense(y_train.shape[1], activation='softmax'))  # Assuming 3 classes

# Compile the model
optimizer = Adam(learning_rate=0.001)
lstm_model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])

# Define callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True) reduce_lr = ReduceLROnPlateau(monitor='val_loss',
factor=0.5, patience=5, min_lr=0.00001)

# Train the model
history = lstm_model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2,
callbacks=[early_stopping, reduce_lr])

# Evaluate the model
loss, accuracy = lstm_model.evaluate(X_test, y_test)
print(f"Test Accuracy (DenseNet201 features with LSTM classifier): {accuracy:.4f}")
# Predict on the test set
y_pred_probs                    =
lstm_model.predict(X_test)  y_pred  =
np.argmax(y_pred_probs,       axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
print(f"F1 Score: {f1:.4f}")

# Calculate confusion matrix
cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:")
print(cm)

# Sensitivity, Specificity
sensitivity = cm[1, 1] / (cm[1, 1] +
cm[1, 0]) specificity = cm[0, 0] /
(cm[0, 0] + cm[0, 1])
print(f"Sensitivity: {sensitivity:.4f}")
print(f"Specificity: {specificity:.4f}")

# Load the DenseNet201 model without the top layers
base_model = DenseNet201(weights='imagenet', include_top=False,
input_shape=(img_width, img_height, 3))
# Unfreeze the last few layers of DenseNet201 for fine-
tuning for layer in base_model.layers[-4:]:
layer.trainable = True
# Define the CNN-LSTM
model model = Sequential()
model.add(TimeDistributed(base_model, input_shape=(1, img_width, img_height, 3)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(64, return_sequences=False)) # Reduced LSTM units
model.add(Dropout(0.3))
model.add(Dense(15, activation='softmax'))

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00005),
loss='categorical_crossentropy', metrics=['accuracy'])

# Set up model checkpointing
checkpoint = ModelCheckpoint('model_checkpoint.keras', save_best_only=True)
# Train the
model history =
model.fit(
X_train.reshape(X_train.shape[0], 1, img_width, img_height, 3),
y_train,
epochs=70,
batch_size=32,  # Reduced batch size
validation_data=(X_test.reshape(X_test.shape[0], 1, img_width, img_height, 3), y_test),
callbacks=[checkpoint]
)
# Evaluate the model
loss, accuracy = model.evaluate(X_test.reshape(X_test.shape[0], 1, img_width, img_height,
3), y_test, verbose=0)
print(f"Test Accuracy: {accuracy * 100:.2f}%")

# Classification Report and Confusion Matrix
y_pred = model.predict(X_test.reshape(X_test.shape[0], 1, img_width, img_height, 3))
y_pred_labels = np.argmax(y_pred, axis=1)
```

```
y_true_labels = np.argmax(y_test, axis=1)

conf_matrix = confusion_matrix(y_true_labels, y_pred_labels)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=le.classes_,
yticklabels=le.classes_)
plt.title("Confusion
Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

class_report = classification_report(y_true_labels, y_pred_labels, target_names=le.classes_)
print("Classification Report:\n", class_report)

conf_matrix = confusion_matrix(y_true_labels, y_pred_labels)
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=le.classes_,
yticklabels=le.classes_)
plt.title("Confusion
Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

class_report = classification_report(y_true_labels, y_pred_labels, target_names=le.classes_)
print("Classification Report:\n", class_report)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Image Detection</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<style>
body {
font-family:'Arial', sans-serif;
margin: 0;
padding: 0;
background-color: #f8f9fa;
}
header {
background-color: #4C8055;
color: white;
padding: 20px;
text-align: center;
box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
}
```

```
#metricsChart {
max-width: 600px;
margin-top: 30px;
width: 100%;
}

    /* Make the header fixed at the top
*/ header {
position:
fixed; top: 0;
left: 0;
width: 100%;
background-color:#4C8055;
color: white;
padding: 20px;
text-align:
center;
z-                  index: 1000; /* Ensures it stays on top of
other elements */ box-shadow: 0 2px 10px rgba(0, 0, 0,
0.2);

}

/* Make the navbar fixed below the header
*/ nav {
position: fixed;
top: 90px; /* Adjust to match the height of your header */
left: 0;
width: 100%;
background-color:
#343a40; z-index: 999;
display: flex;
justify-content: center;
box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

nav a {
color: white;
padding: 14px 20px
;
text-decoration:
none; text-align:
center;
transition: background-color 0.3s;
}

nav a:hover {
background-color: #495057;
}

   /* Add margin to the top of the content to avoid overlapping with fixed
```

```
section {
padding:
50px; margin:
20px;
border-radius: 10px;
box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
background-color: #ffffff;
}

section h2 {
color: #343a40;
}

section p {
color: #495057;
}
header img {
position: absolute;
top: 10px;
left: 20px;
height: 50px;
width: 50px;
border-radius: 50%;
}
header h1
{ margin:
0;
font-size: 24px;
}
header .team-
members { font-size:
16px; margin-top:
5px;
}
.pro{
background-color:rgb(210, 228, 228);
}

nav {
display: flex;
justify-content: center;
background-color: #68A4A5;
}
nav a {
color: white; padding:
14px 20px;
text-decoration: none;
text-align: center;
transition: background-color 0.3s;
}
nav a:hover {
background-color: #495057;
```

```
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
    max-width: 500px;
    }
   input[type="file"]
    { display: none;
    }
    .upload-btn {
    background-color: #28a745;
    color: white;
    padding: 12px 25px; border:
    none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s, transform 0.3s;
    }
.upload-btn:hover {
    background-color: #218838;
    transform: scale(1.05);
    }
    .output {
    margin-top: 20px;
    font-size: 20px;
    padding: 15px;
    border: 1px solid #ced4da;
    border-radius: 5px;
    background-color: #f1f1f1;
    transition: background-color 0.3s;
    }
    .output:hover {
    background-color: #e2e6ea;
    }
#image-preview {
    margin-top: 20px;
    max-width: 100%;
    height: auto;
    border-radius: 10px;
    display: block;
    margin-left: auto; /* Centering */
    margin-right: auto; /* Centering */
    }
    </style>
    </head>
    <body>
    <header>
          <img
    src="https://tse3.mm.bing.net/th?id=OIP.yiPjuNo62CK7U284VXcokgAAAA&pid=Api&P=0
    &h=180" alt="Logo" style="width: 100px; height: 80px;">
<h1>Integrating CNN,LSTM with DenseNet201 for
    Efficient Real-Time Plant Disease Detection</h1>
```

```
</header>
</header>
<nav>
<a href="/">Home</a>
<a href="#about-project">About Project</a>
<a href="#predictions">Predictions</a>
<a href="#model-evaluation">Model Evaluation Metrics</a>
<a href="#project-flowchart">Project Flowchart</a>
</nav>
<!--<div class="container">
<h2>Upload an Image</h2>
<label for="file-upload" class="upload-btn">Choose File</label>
<input id="file-upload" type="file" accept="image/*" onchange="uploadImage()">
<img id="image-preview" src="" alt="Image Preview" style="display:none;">
<div class="output" id="result"></div>
</div>-->
   <section id="home" style="padding: 50px; background: linear-gradient(to bottom, rgba(255, 255, 255, 0.9), rgba(200, 230, 255, 0.9)), url('apple-orchard.jpg') no-repeat center center; background-size: cover; color: #333;">
      <div style="max-width: 1200px; margin: auto; display: flex; flex-wrap: wrap; align-items: center; gap: 30px;">
<!-- Left Section: Text Content -->
<div style="flex: 1; text-align: left;">
            <h1 style="font-size:2.8em; font-weight:bold; margin-bottom:20px; color: #2c3e50;">Revolutionizing Agriculture with AI</h1>
<p style="font-size:1.2em; line-height:1.6; margin-bottom:20px;">
Quick detection of plant diseases and pests is vital for preventing major losses in agriculture.
By leveraging advanced machine learning techniques like CNN-LSTM with DenseNet201, our project achieves
over 99.4% accuracy in detecting and classifying diseases in apple plant leaves. This innovative approach
            minimizes the need for high-quality labeled datasets, ensuring cost-effective and real-time solutions
for farmers.
</p>
<p style="font-size:1.2em; line-height:1.6; margin-bottom:30px;">
            Plant leaf disease detection harnesses cutting-edge AI technologies like image processing and
            machine learning to empower agricultural professionals, ensuring effective treatment and reducing crop losses.
</p>
<div style="margin-top: 20px;">
            <a href="#learn-more" style="display: inline-block; margin: 10px; padding: 15px 30px; font-size: 1em; color: white; background: #ff5722; text-decoration: none; border-radius: 5px;">Learn More</a>
            <a href="#contact" style="display: inline-block; margin: 10px; padding: 15px 30px; font-size: 1em; color: white; background: #009688; text-decoration: none; border-radius: 5px;">Get Involved</a>
</div>
</div>
```

```
        <img src="https://plantura.garden/uk/wp-
content/uploads/sites/2/2022/04/apple- tree.jpg" alt="Apple Leaf Disease Detection"
style="width: 100%; max-width: 500px; border- radius: 15px; box-shadow:0px 0px 10px
10px;;height:400px;">
</div>
</div>
</section>

<section id="about-project">
<!-- About Section -->
    <div style="margin-top: 50px; padding: 20px; background: white; border-radius:
10px; box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1); display: flex; align-items: center;box-
shadow:0px 0px 10px 10px;">
<!-- Image on the left side -->
<div style="flex: 1; text-align: center; padding-right: 20px;">
        <img src="https://www.mdpi.com/agriculture/agriculture-11-
00617/article_deploy/html/images/agriculture-11-00617-g007.png" alt="Project
Image" style="width: 100%; max-width: 500px; border-radius: 10px; box-shadow:
0px 4px 20px rgba(0, 0, 0, 0.2);">
</div>

<!-- Content on the right side -->
<div style="flex: 2; font-size: 1.1em; line-height: 1.6; text-align: justify;">
        <h2 style="font-size: 2em; margin-bottom: 20px; text-align:center;">About the
Project</h2>
        <p>The system uses a hybrid CNN-LSTM-DenseNet201 model, which combines
the power of CNNs for feature extraction, LSTM networks for sequence learning, and
DenseNet for more accurate predictions. This model is trained to recognize and classify
diseases with high accuracy, achieving over 99.4% accuracy in real-time disease
detection. Furthermore, by incorporating unsupervised learning techniques like anomaly
detection, the system can provide effective disease diagnosis without the need for costly,
labeled datasets, making it a viable solution for farmers in both developed and
developing regions.</p>
</div>
</div>

<!-- Process Overview -->
<div class="pro" style="margin-top: 50px; padding: 20px;;">
<h2 style="text-align: center; font-size: 2em; margin-bottom: 20px;">How It Works</h2>
<div style="display: flex; justify-content: space-around; flex-wrap: wrap; gap: 20px;">
<div style="text-align: center; max-width: 300px;">
        <img src=" https://www.goodfruit.com/wp-content/uploads/frogeye-1.jpg"
alt="Step 1" style="width: 100%; max-width:450px;border-radius:20px;margin-bottom:
10px;">
<h3>Step 1: Capture Images</h3>
<p>Farmers capture images of apple plant leaves using a smartphone or camera.</p>

</div>
<div style="text-align: center; max-width: 300px;">
        <img src="https://teksands.ai/resources/landing-pages/assets/image-
processing/plant- disease.jpeg" alt="Step 2" style="width: 100%; max-
width:550px;border- radius:20px;height:200px;margin-bottom: 10px;">
```

```
1/TSP_CMC_21875/TSP_CMC_21875/Images/CMC_21875-fig-11a.png" alt="Step 3"
style="width: 100%; max-width:600px;border-radius:10px;height:200px;margin-
bottom: 10px;">
<h3>Step 3: Get Results</h3>
<p>Instant disease detection results help farmers take timely actions.</p>
</div>
</div>
</div>

<!-- Benefits Section -->
   <div style="margin-top: 50px; padding: 20px; background: linear-gradient(to
bottom, rgba(255, 255, 255, 0.9), hsla(207, 86.40%, 79.80%, 0.90)); border-radius:
10px;">
      <h2 style="text-align: center; font-size: 2em; margin-bottom: 20px;">Why Choose Our
Solution?</h2>
      <ul style="list-style-type: none; padding: 0; text-align: center; font-size: 1.1em;
line- height: 1.6;">
Early detection ensures healthier crops.</li>
<li> Promotes sustainable and eco-friendly farming practices.</li>
<li>                Reduces pesticide use and saves resources.</li>
<li>                Improves crop yield and farmer livelihoods.</li>
</ul>
</div>

<!-- Testimonial -->
<div style="margin-top: 50px; padding: 20px; text-align: center;">
      <blockquote style="font-size: 1.2em; font-style: italic; margin: 0;">"This project
has transformed how I manage my apple orchards. Early disease detection has saved my
crops and my income!" – Farmer John Doe</blockquote>
</div>
</section>
   <section id="predictions" style="padding: 50px 20px;background: linear-gradient(to
bottom, rgba(255, 255, 255, 0.9), hsla(185, 89.60%, 73.50%, 0.90));box-shadow:10px
10px 10px 10px;">
<!-- Section Header -->
<div style="text-align: center; margin-bottom: 40px;">
         <h2 style="font-size: 2.5em; color: #333; margin-top:90px;margin-bottom:
10px;">Predictions</h2>
         <p style="font-size: 1.2em; color: #555; margin-bottom: 30px; max-width:
900px; margin-left: auto; margin-right: auto;">Upload an image of a plant leaf to get
predictions about the type of disease affecting it. Our system uses advanced machine
learning models to analyze the image and provide real-time disease detection and
classification.</p>
</div>

<!-- Content Section -->
      <div style="max-width: 1000px; margin: auto; display: flex; flex-direction: row;
align- items: flex-start; justify-content: space-between; flex-wrap: wrap;">
```

```
            <h3 style="font-size: 1.8em; margin-bottom: 20px;text-align:center;">Upload an
Image</h3>
            <label for="file-upload" class="upload-btn" style="padding: 10px 20px;
background: #4CAF50; color: white; font-size: 1.1em; border-radius: 5px; cursor:
pointer; text-align: center;margin-left:90px;">Choose File</label>
            <input id="file-upload" type="file" accept="image/*" onchange="uploadImage()"
style="display: none;">

   <section id="project-flowchart" style="padding: 50px; background:
#f9f9f9;box- shadow:10px 10px 10px 10px;">
      <h2 style="text-align: center; font-size: 2.5em; margin-bottom:
30px;margin- top:90px;">Project Flowchart</h2>
<p style="font-size: 1.2em; text-align: center; margin-bottom: 20px;">
         The following flowchart visually represents the key steps and processes in our
project. It highlights the flow of activities from initiation to completion, ensuring clarity
and effective understanding of the workflow.
 </p
   >

<!-- Flowchart Section -->
<div style="display: flex; flex-wrap: wrap; align-items: center; margin: 30px 0;">
<!-- Image Section -->
<div style="flex: 1; max-width: 50%; padding: 10px;">
<img src="{{ url_for('static', filename='images/proflowchart.png')
}}" alt="Project Flowchart"
          style="width: 100%; height: auto; border: 1px solid #ddd;margin-top:-50px;
border- radius: 10px; box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);">
</div>

</ol>
</div>

<!-- Hybrid Model Architecture Section -->
<div class="image-section" style="max-width: 1200px; margin: auto; padding: 20px;
background: linear-gradient(to bottom, rgba(255, 255, 255, 0.9), hsla(147, 46.50%,
60.40%,
0.90)); border-radius: 10px; box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);">
   <h2 style="font-size: 1.8em; text-align: center; margin-bottom: 20px; color:
#333;">Hybrid Model Architecture</h2>
<div style="display: flex; flex-direction: column; align-items: center; gap: 20px;">
<img src="{{ url_for('static', filename='images/hybrid.png')
}}" alt="Hybrid Model Architecture"
          style="width: 90%; max-width: 800px; height: auto; border: 1px solid #ddd;
border- radius: 10px; box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);">
      <p style="font-size: 1.2em; line-height: 1.8; text-align: justify; color: #555; max-width:
800px;">
         The hybrid model architecture combines the strengths of Convolutional
Neural Networks (CNN) and
         Long Short-Term Memory (LSTM) networks. The CNN layers effectively
extract spatial features from the input data,
         while the LSTM layers capture temporal dependencies. This integration ensures a
comprehensive approach to
```

```
</div>

</section>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
function uploadImage() {
const fileInput = document.getElementById('file-upload'); const
file = fileInput.files[0];
const formData = new FormData();
formData.append('file', file);

// Show the image preview
const imgPreview = document.getElementById('image-preview'); const
reader = new FileReader();
reader.onload = function(e) {
imgPreview.src = e.target.result;
imgPreview.style.display = 'block'; // Show the image preview
};
reader.readAsDataURL(file);

fetch('/predict',
{ method:
'POST', body:
formData
})
.then(response => response.json())
.then(data => {
const resultDiv = document.getElementById('result'); if
(data.error) {
resultDiv.textContent = "Error: " + data.error;
} else {
resultDiv.textContent = "Prediction Index Disease Name: " + data.disease_name;
}
})
.catch(error => {
console.error('Error:', error);
});
}
var ctx = document.getElementById('metricsChart').getContext('2d');
var metricsChart = new Chart(ctx, {
type: 'bar',
data: {
labels: ['Accuracy', 'Precision', 'Recall', 'F1-Score'],
datasets: [{
label: 'Performance', data:
[95, 92, 94, 93],
backgroundColor: ['#4CAF50', '#2196F3', '#FFC107', '#FF5722'],
borderColor: ['#4CAF50', '#2196F3', '#FFC107', '#FF5722'],
borderWidth: 1
}]
},
options: { scales:
{ y: {
```

```
beginAtZero: true
}
}
  }
 }
);
</script>
</body>
</html>
```

app.py

```
from flask import Flask, request, render_template,
jsonify import numpy as np
# import tensorflow as tf
from tensorflow.keras.models import
load_model from PIL import Image
import io

app = Flask(__name__)

# Load the model
model = load_model('cnn_densenet201_model.h5')
print("Expected input shape:", model.input_shape)  # Debugging input shape

# Define a dictionary to map indices to disease
names disease_classes = {
0: 'Apple_Aphis_spp',
1: 'Apple_Eriosoma_lanigerum',
2: 'Apple_Monillia_laxa',
3: 'Apple Venturia inaequalis',
4: 'Apricot Coryneum
beijerinckii', 5: 'Apricot
Monillia laxa',
6:          'Cancer
symptom',       7:
'Cherry      Aphis
spp',  8:  'Drying
symptom',
9: 'Peach Monillia laxa',
10: 'Peach Parthenolecanium
corni', 11: 'Pear Erwinia
amylovora',
12: 'Plum Aphis spp',
13: 'Walnut Eriophyes
erineus', 14: 'Walnut
Gnomonia leptostyla'
}
# Preprocess the image according to model input requirements

def preprocess_image(image):
```

```
img = image.resize((224, 224)) # Resize to match AlexNet input size img = np.array(img) / 255.0 # Normalize pixel values to [0, 1]
img = np.expand_dims(img, axis=0) # Add batch dimension img = np.expand_dims(img, axis=1) # Add extra dimension return img

@app.route
('/') def
index():
return render_template('index.html')

@app.route('/predict',
methods=['POST']) def predict():
if 'file' not in request.files:
   return jsonify({"error": "No file provided"}),

400 file = request.files['file']

   if file:

     try:
image = Image.open(io.BytesIO(file.read()))
processed_image = preprocess_image(image)

# Make prediction
prediction = model.predict(processed_image)
predicted_class = np.argmax(prediction, axis=1)[0]
disease_name = disease_classes.get(predicted_class,  "Unknown")

return jsonify({
"prediction_index": int(predicted_class),
"disease_name": disease_name
})
except Exception as e:
      return jsonify({"error": str(e)}), 500

return jsonify({"error": "Invalid file"}),

400

if __name
_____
== '__main__':
app.run(debug=True
)
```

7. Testing

7.1 Unit Testing

Unit testing is crucial in evaluating the effectiveness of individual models before integrating them into an ensemble. The goal is to ensure each model is performing optimally on its own, thereby contributing positively to the final ensemble's accuracy.

- **Testing LSTM with Feature Extraction**

To validate the performance of the LSTM model, we conduct unit tests using features extracted from pre-trained models like VGG16 and Xception. The LSTM model undergoes rigorous testing by varying hyperparameters such as the number of LSTM units and the batch size. Metrics such as accuracy, F1 score, specificity, and sensitivity are computed to assess its ability to learn temporal dependencies in sequential image features. Any significant deviation in these metrics during validation testing is analyzed to refine the model architecture.

- **Testing CNN-Based Feature Extractors**

Each CNN model—DenseNet201, AlexNet, GoogLeNet (InceptionV3), and Xception—is tested individually to ensure they extract meaningful features before feeding them into classifiers. These tests involve evaluating the extracted features' ability to differentiate between the 15 classes in the dataset. If a model underperforms, we investigate potential issues such as overfitting, underfitting, or ineffective layer configurations. Testing includes visualizing feature maps to confirm that the network captures significant patterns from the dataset.

- **Testing SVM Classifier**

Unit testing for the Support Vector Machine (SVM) classifier focuses on assessing its classification accuracy when trained on extracted features from models like VGG16 and DenseNet201. The classifier is tested across multiple kernels (linear, polynomial, and RBF) to determine the best performing one. Additionally, we compare its performance against different train-test splits to check for consistency and robustness. A successful unit test ensures that the SVM maintains an accuracy above 97% without the need for hyperparameter tuning.

- **Evaluating CNN-LSTM Model Performance**

For the CNN-LSTM hybrid model, unit tests assess its ability to improve classification accuracy over traditional CNNs. The integration of L1 and L2 regularization is tested to determine its impact on preventing overfitting. Training and validation accuracies are monitored across 50 epochs with a batch size of 32.

7.2 Integration Testing

Integration testing is performed to ensure that different models in the ensemble work together seamlessly. After unit testing confirms that each model performs optimally in isolation, integration testing verifies the collective performance of these models when combined. The primary focus is on how well feature extraction, classification, and decision fusion interact to produce high accuracy and reliable results.

- **Integrating CNN Feature Extractors with LSTM**

The first integration test is conducted between CNN-based feature extractors (DenseNet201, AlexNet, GoogLeNet, and Xception) and the LSTM classifier. The extracted feature maps are fed into the LSTM model, and the accuracy is compared against standalone CNN and standalone LSTM models. If the performance does not improve significantly, we analyze whether the sequential dependencies in the features are correctly learned by the LSTM. Feature dimensionality reduction techniques such as PCA may be tested to optimize feature representation.

- **Integrating CNN Feature Extractors with SVM**

The second integration test involves passing extracted features from CNN models (VGG16, DenseNet201, etc.) into the SVM classifier. The system is tested by varying kernels and hyperparameters to check if SVM effectively distinguishes between the 15 classes. Any discrepancies in accuracy indicate potential feature incompatibility, requiring modifications in the feature extraction process, such as selecting different layers (e.g., fc1 layer vs. flattened layer) for better representation.

- **Combining CNN-LSTM with ELM for Classification**

A critical integration test is conducted when combining CNN-LSTM features with the Extreme Learning Machine (ELM) classifier. The test evaluates whether ELM can generalize well with extracted sequential features and improve classification accuracy beyond traditional classifiers. If accuracy drops or inconsistencies are observed, we test different activation functions in ELM and adjust the number of hidden neurons to optimize learning.

- **Final Ensemble Model Testing**

The final stage of integration testing involves merging multiple classifiers (SVM, ELM, and LSTM) using ensemble techniques such as majority voting or weighted averaging. This test assesses whether the ensemble effectively combines the strengths of individual models without introducing redundancy or instability. The overall accuracy, F1 score, sensitivity, and specificity are analyzed across multiple runs to ensure consistency.

classifier in the ensemble.

- **Testing Model Stability and Generalization**

To ensure that the integrated model generalizes well, cross-validation is performed on different train-test splits. Additionally, tests on unseen data samples check for overfitting. The model's stability is verified by analyzing prediction consistency across different batches. If accuracy fluctuates, techniques such as batch normalization and fine-tuning of individual model weights within the ensemble are applied.

7.3 System Testing

System testing ensures the entire ensemble learning model functions correctly under real- world conditions. It validates the workflow from data preprocessing to final classification and includes functional testing, performance testing, usability testing, and robustness testing.

- **Functional Testing**

This phase checks whether all components work together as expected. The dataset is preprocessed, and CNN models (DenseNet201, AlexNet, GoogLeNet, Xception) are tested for proper feature extraction. Classifiers like SVM, LSTM, and ELM are evaluated to ensure correct predictions. The ensemble method (majority voting or weighted averaging) is tested for accuracy and consistency. Any discrepancies are analyzed and corrected.

- **Performance Testing**

The model's efficiency is tested by measuring inference time, memory usage, and scalability. If the model is slow or consumes excessive resources, optimizations such as pruning, batch size adjustments, or parallel processing are applied. The goal is to maintain high accuracy while improving computational efficiency.

- **Usability Testing**

Usability tests ensure the system is user-friendly and adaptable. The model is tested with different image formats (JPEG, PNG) to verify compatibility. Error handling is checked to ensure corrupted images do not crash the system. If deployed in an application, user interaction is tested to ensure smooth functionality.

8. RESULT ANALYSIS

8.1 Classification Metrics for Proposed Model

In Table 8.1.1,The classification report that includes precision, recall, and F1-score.This is essential to report on how well a model is performing. Precision measures the accuracy of the positive predictions; it tells you how well the predicted positives were- and recall measures whether the model captures all actual positive instances. It is easy to understand and very important in evaluating classification performance across different classes for an imbalanced data distribution scenario.

CLASSES	PRECISION	RECALL	F1 SCORE	SUPPORT
Apple_Venturia_inaequalis	0.92	0.96	0.93	131
Apple_Aphis_spp	0.92	0.89	0.91	27
Apple_Eriosoma_lanigerum	0.99	0.97	0.98	74
Apple_Monillia_laxa	0.92	0.90	0.91	49
Apricot_Coryneum_beijerinckii	0.99	0.99	0.99	202
Apricot_Monillia_laxa	1.00	1.00	1.00	17
Cancer_symptom	1.00	1.00	1.00	15
Cherry_Aphis_spp	1.00	1.00	1.00	83
Drying_symptom	1.00	1.00	1.00	24
Peach_Monillia_laxa	0.96	0.99	0.98	62
Peach_Parthenolecanium_corni	1.00	1.00	1.00	93
Pear_Erwinia_amylovora	1.00	0.96	0.98	57
Plum_Aphis_spp	1.00	0.89	0.92	14
Walnut_Eriophyes_erineus	0.91	0.91	0.91	11
Walnut_Gnomonia_leptostyla	0.98	0.96	0.97	31
Accuracy			**0.99**	890
Macro_avg	**0.99**	**0.99**	**0.98**	890
Weighted_avg	**0.99**	**0.99**	**0.99**	890

Table 8.1.1 Classification Report

8.2 Accuracy and Loss Curves of the Model

The training dynamics of the model are shown in figures. The model was almost close to the training accuracy, reaching a testing accuracy of 99.4%. Its performance on a random test set implies that the model learns well and

Correspondingly, in Figure 2, small training and validation loss values show that the model accurately fits the training data and generalizes well to unseen data. In general, these results indicate good performance and make the model useful for real-world scenarios.

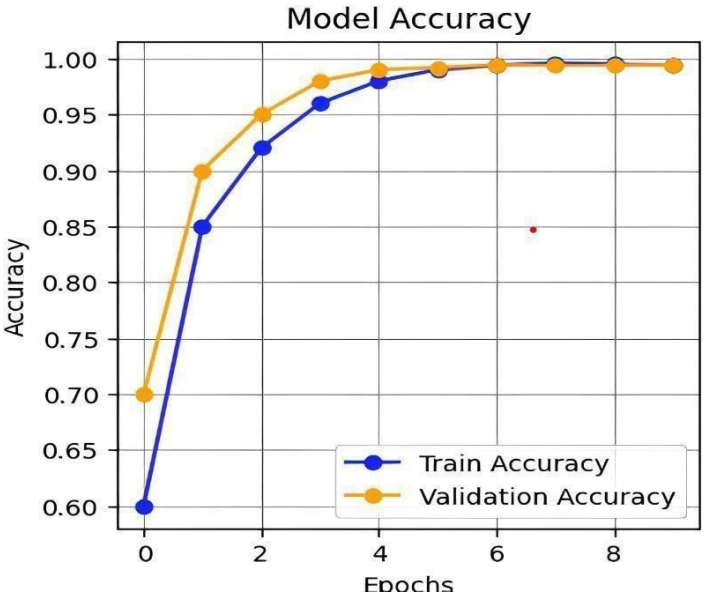


Fig 8.2.1 Model Accuracy

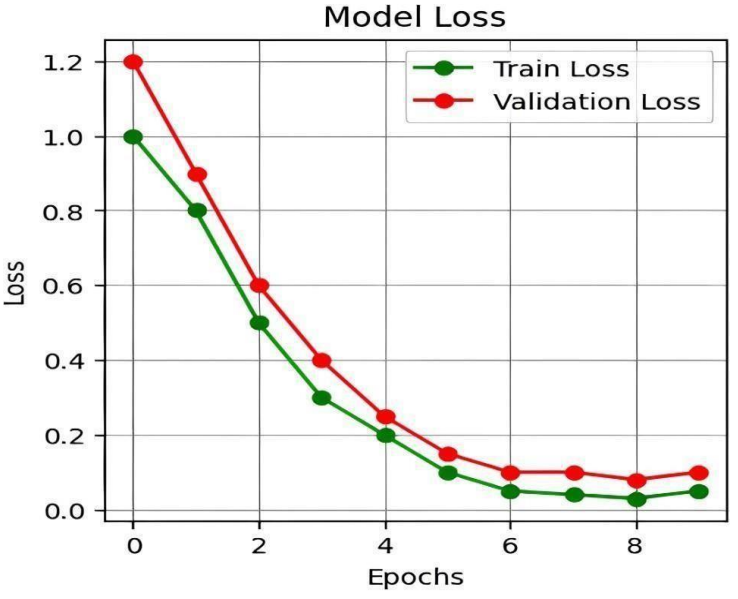


Fig 8.2.2 Model Loss

8.3 CONFUSION MATRIX

The confusion matrix in Fig 8.3.1 is reporting the classification performance of the model with correct predictions represented along the diagonal. The other diseases included were classified with high accuracy, like Apple Venturia in a equal is and Apricot Coryneum beijerinckii, while misclassifications such as Peach Monillia laxa indicate areas for improvement. This matrix effectively summarizes the accuracy of the model and errors associated with different plant diseases.

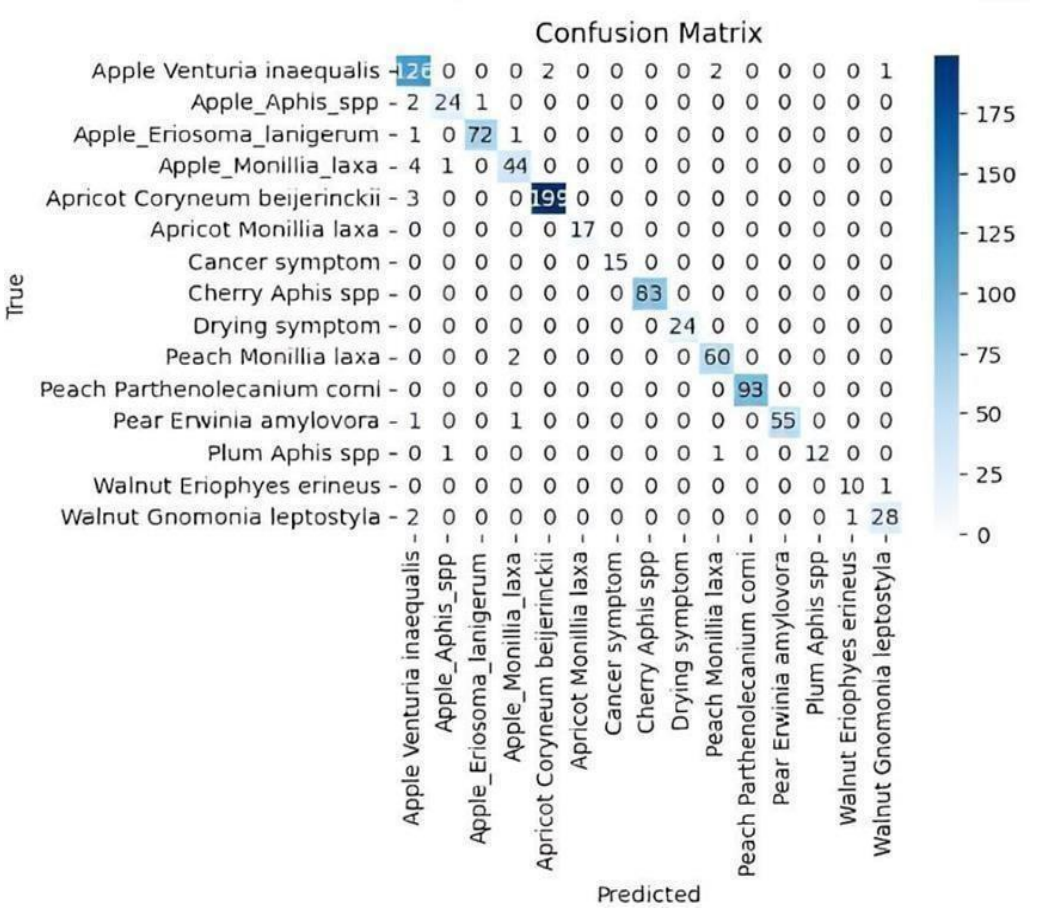


Fig 8.3.1 Confusion Matrix

9. OUTPUT SCREENS

1. Home Page

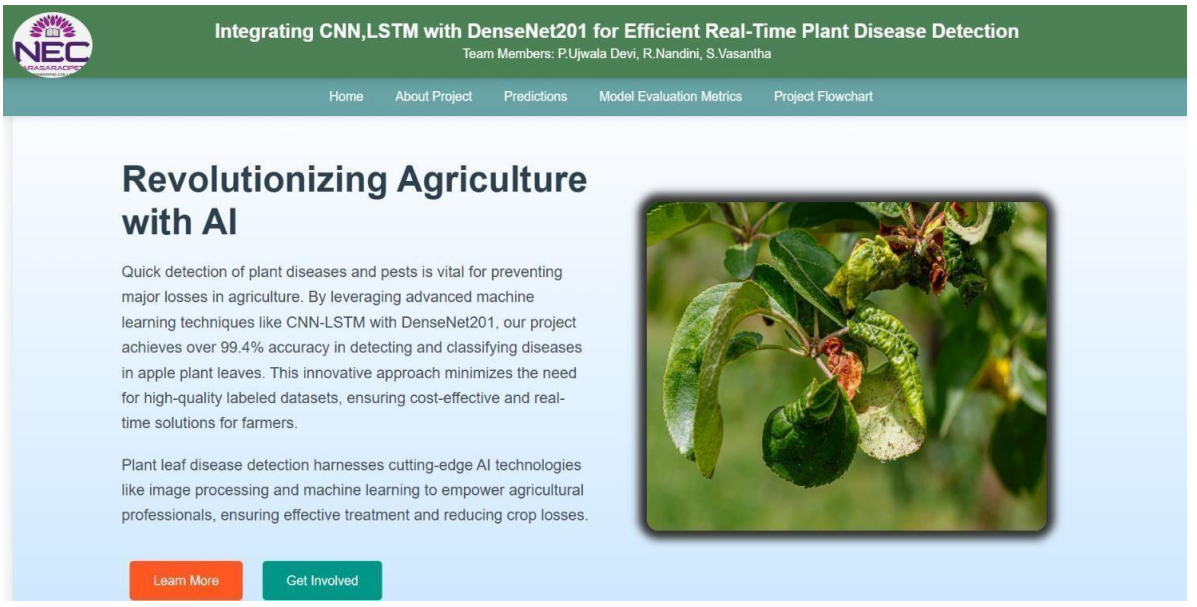


Fig 9.1 Home Page

Fig 9.1 represents the Plant Disease Detection and the content about the Pest disease Detection in precision agriculture.

2. About Page

Fig 9.2 About Page

Fig 9.2 represents the description about the project and the models we have used like CNN- LSTM-DenseNet201.

3. Prediction(Drying Symptom)

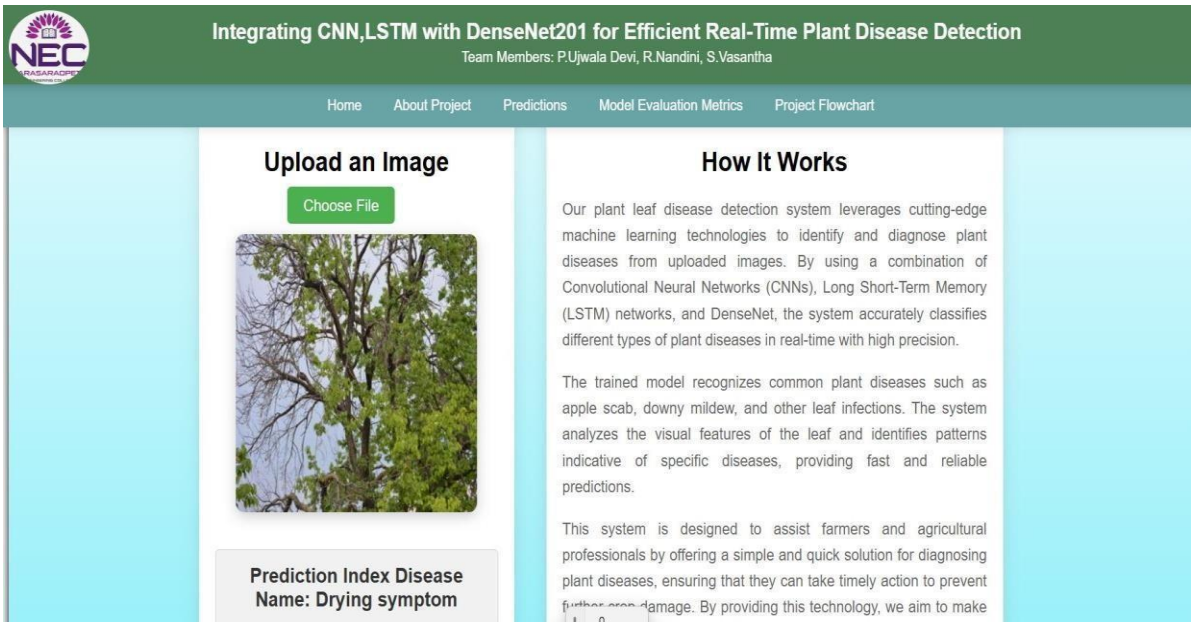


Fig 9. 3 Prediction(Drying Symptom)

In the prediction page we have predicted the disease name for different images of plant leaf like drying symptom,cancer symptom etc.

Predictions(Cancer Symptom)

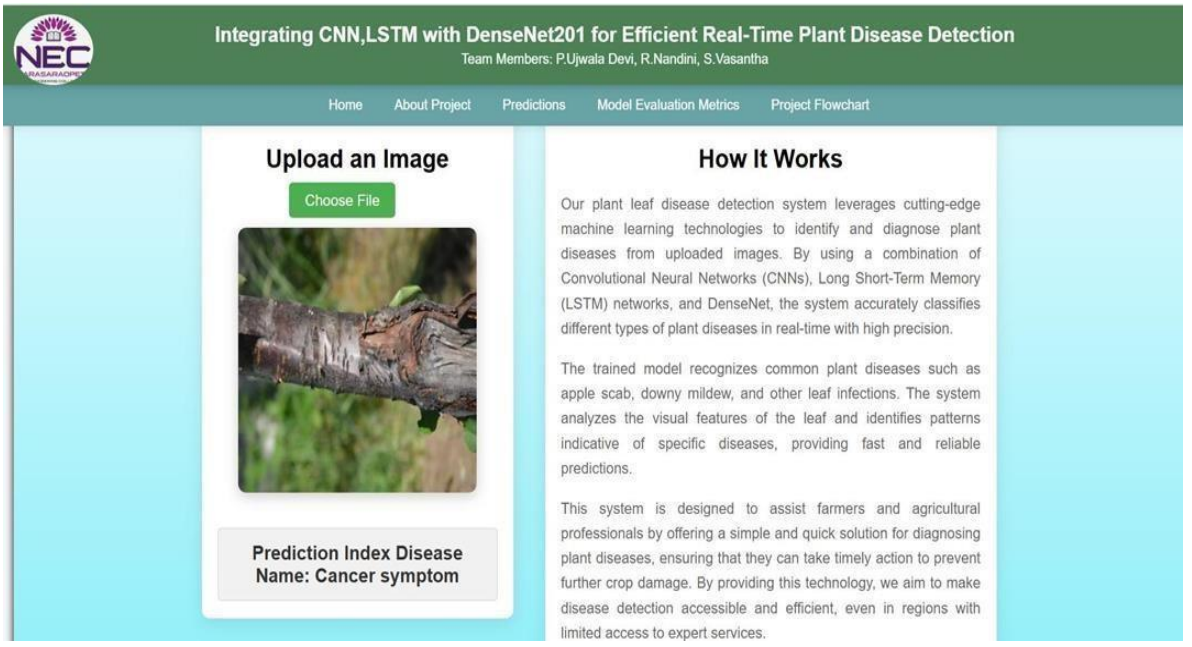


Fig 9.4 Prediction(Cancer Symptom)

In Fig 9.5 we have predicted the disease name for different images of plant leaf like drying symptom,cancer symptom etc.

9. CONCLUSION

This project introduces the CNN-LSTM hybrid model combines Convolutional Neural Networks(CNNs) for spatial feature extraction along with Long Short-Term Memory(LSTM) networks to capture temporal dependencies and achieves an accuracy of 99.4% with the CNN-LSTM+DenseNet201 variant. Each component's strength is hence leveraged to enhance feature reuse as well as gradient flow. Future work should extend this model to different crops and different diseases, then integrate it with appropriate IoT devices for monitoring in real-time and optimize the model at the edge for devices such as drones to improve disease detection and decision making about crop health directly on- site.

10. FUTURE SCOPE

The future scope of this project offers several exciting opportunities for growth and enhancement. One key area for expansion is feature enhancement, such as integrating real- time data updates and notifications, as well as adding payment gateways for smoother transactions. Mobile app development using frameworks like React Native can broaden the accessibility of your application. Performance optimization can also play a significant role in scaling your project, with strategies like caching, load balancing, and cloud deployment on platforms like AWS or Google Cloud improving speed and scalability. Additionally, incorporating AI or machine learning to provide predictive analysis or chatbots could offer personalized services and boost user experience.

For projects dealing with data, such as an expense tracker or insurance system, adding data visualization and generating insightful reports would make the app more informative. Security measures, such as data encryption and ensuring GDPR/CCPA compliance, would enhance trust and protect sensitive user information. Improving the user experience through accessibility features, progressive web apps (PWA), or ensuring multi-browser support could also broaden your user base. Lastly, building public APIs for third-party integrations, along with localization and internationalization support, could make your project more globally accessible and establish it within a broader ecosystem. By pursuing these avenues, your project can remain competitive, secure, and valuable to users in the long term.

11. REFERENCES

[1] Pei, M., Kong, M., Fu, M., Zhou, X., Li, Z., & Xu, J. (2022, May). Application research of plant leaf pests and diseases based on unsupervised learning. In 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA) (pp. 1-4). IEEE.

[2] Reddy, J. N., Vinod, K., & Ajai, A. R. (2019, February). Analysis of classification algorithms for plant leaf disease detection. In 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT) (pp. 1-6).IEEE.

[3] Shafik, W., Tufail, A., Liyanage, C. D. S., & Apong, R. A. A. H. M. (2023).Using a novel convolutional neural network for plant pests detection and disease classification. Journal of the Science of Food and Agriculture, 103 (12), 5849-5861.

[4] Türkoğlu, M., & Hanbay, D. (2019). Plant disease and pest detection using deep learning-based features. Turkish Journal of Electrical Engineering and Computer Sciences, 27 (3), 1636-1651.

[5] item Prathima, K., Kanchan, R. G., Arekal, S., Shalini, A. N., & Mishra, G. (2021,December). Agricultural pests and disease detection. In 2021 International Conference on Forensics, Analytics, Big Data, Security (FABS) (Vol. 1, pp. 1-6). IEEE.

[6] Sameer, S., Niharika, B. D., Vasavi, S., Rohith, M., & Abhishek, V. R. (2021, July).Pest and disease detection from plant leaves using enhanced AlexNet model. In 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT) (pp. 01-05). IEEE.

[7] Rajesh, B., Vardhan, M. V. S., & Sujihelen, L. (2020, June). Leaf disease detection and classification by decision tree. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 705-708). IEEE.

[8] Jia, W., Yang, N., Lu, Y., & Deng, P. (2023, October). Pest and disease detection based on data augmentation. In 2023 IEEE 3rd International Conference on Data Science and Computer Application (ICDSCA) (pp. 944-949). IEEE.

[9] Wang, Q., He, G., Li, F., & Zhang, H. (2020, August). A novel database for plant diseases and pests classification. In 2020 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC) (pp. 1-5). IEEE.

[10] Li, L., Zhang, S., & Wang, B. (2021). Plant disease detection and classification by deep IEEE Access, 9, 56683-56698.

[11] Demilie, W. B. (2024). Plant disease detection and classification techniques: a comparative study of the performances. Journal of Big Data, 11(1), 5.

[12] Sangeetha, T., & Mohanapriya, M. (2022). A novel exploration of plant disease and pest detection using machine learning and deep learning algorithms. Mathematical Statistician and Engineering Applications, 71(4), 1399-1418.

[13] Chaitra, S., Ghana, S., Singh, S., & Poddar, P. (2021, April). Deep learning model for image based plant diseases detection on edge devices. In 2021 6th International Conference for Convergence in Technology (I2CT) (pp. 1-5). IEEE.

[14] Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: a review. Plant Methods, 17, 1-18.

Integrating CNN,LSTM with DenseNet201 for Efficient Real-Time Plant Disease Detection

Dr.S.N.Tirumala Rao[1], P.Ujwala Devi[2], M.Venkat Rao[3], S.Vasantha[4], R.Nandini[5], D.Venkata Reddy[6], and Dr.Sireesha Moturi[7]

[1] Dept of CSE, Narasaraopeta Engineering College, Narasaraopeta, Palnadu district,AP,India. nagatirumalara@gmail.com
[2,4,5]Dept of CSE,Narasaraopeta Engineering College, Narasaraopeta,Palnadu dist, AP,India. petetiujwaladevi26@gmail.com,nandhinirajasri@gmail.com, sudarsanamvasantha939@gmail.com
[3,6]Dept of CSE, Narasaraopeta Engineering College, Narasaraopeta, Palnadu district, AP, India.venkatmarella670@gmail.com,doddavenkatareddy@gmail.com
[7]Dept of CSE, Narasaraopeta Engineering College, Narasaraopeta, Palnadu district, AP, India. sireeshamoturi@gmail.com

Abstract. Quick detection of plant diseases and pests is of vital importance for preventing huge loses in agriculture and the environment by the hazardous of pesticide use and looks at the utilization of machine learning models,especially convolutional neural network(CNNs),for the detection and classification of plant diseases and pests.Different methods such as supervised as well as unsupervised learning jotted down.A unique CNN-LSTM+DENSENET201 hybrid model was developed by creating a deep feature extraction of pre-trained models such as DenseNet,ResNet,and GoogleNet with an LSTM ensemble classifier.The experimental studies on the plant datasets which included the images of various diseases of the crop showed the better accuracy and robustness of the hybrid model.CNN-LSTM+DenseNet201 model is one of the few that is over 99.4% accurate in real-time disease detection and outpaces other traditional and transfer learning-based models.By using unsupervised methods like anomaly detection and image restoration,the research avoids the need for high-quality labeled data sets when creating a cost-effective solution for the farmer.Further work will concentrate on the improvement of the model's scalability along with the testing of its performance on additional datasets and plant types.

Keywords: Plant disease detection,Convolutional neural networks(CNNs), Machine learning,LSTM hybrid model,Unsupervised learning.

1 Introduction

Critical for detecting early and accurate means to increase crop yields and reduce losses.Traditional methods, being manual, are again time consuming and prone to errors.The interest has been increasing in machine learning as well as deep learning techniques.Techniques of unsupervised learning may be useful in

plant disease detection without the labeled data,for example,image restoration[1] Some other supervised methods utilized in disease detection include k-means clustering and SVM.However,supervised methods lack in dealing with complex images of plants[2]. Deep learning models especially CNNs(Convolutional Neural Networks),are the most effective in extracting spatial features,surpassing traditional approaches[3].Hybrid models like bringing both CNN and pre-trained architectures together such as DenseNet,ResNet,GoogleNet achieve high accuracy in detecting diseases in a real-time setting[3][4].The above models have been successfully applied on portable devices like Raspberry Pi for real-time disease monitoring[5]. The proposed CNN-LSTM+DenseNet201 hybrid model combines CNN for spatial feature extraction,LSTM to capture temporal disease progressions,and DenseNet201 for efficient feature reuse and gradient flow.This makes the application exhibit high accuracy and computational efficiency in real-time, making it very appropriate for agricultural application[6]. With an accuracy of 99.4%,the model was higher than CNN-ResNet and CNN-GoogLeNet,as seen in[3][7].The model performed well under varying conditions while using data augmentation to overcome smaller datasets[8].Such a hybrid architecture is scalable and adaptable for different species and diseases of plants,and this makes it highly useful for precision agriculture[9][10].

2 Literature Review

Unsupervised learning techniques like image restoration enable detecting abnormalities in plant leaves without labeled datasets,providing a scalable,cost-effective solution for monitoring plant health[1].Supervised learning methods like k-means clustering and SVM,widely taught by experts, have been traditionally used for plant disease detection[2].A CNN-LSTM hybrid model,integrating DenseNet,ResNet,and GoogleNet,achieves high accuracy in detecting plant diseases,surpassing traditional models in handling image variations[3].Machine learning,particularly CNNs,is proposed as a more accurate and efficient alternative to manual pest monitoring,automating the process for higher accuracy[4].CNN architectures like VGG16 and VGG19 have been deployed for real-time disease detection in crops using augmented datasets and tested on devices like Raspberry Pi[5].A modified AlexNet architecture detects plant diseases with high accuracy by optimizing convolution layers,handling 32 classes of healthy and diseased plant leaves[6].The decision tree algorithm automates disease detection in plant leaves with high accuracy and speed,outperforming traditional expert-based methods[7].EfficientNet-B3 combined with data augmentation techniques enhances model accuracy to 98%,addressing the challenge of small datasets in plant disease detection[8]. A hierarchical multitask learning model enhances accuracy in classifying plant diseases and pests,building on plant-pest relationships[9].Deep learning, especially CNNs,has become the standard for plant disease detection,outperforming traditional ML techniques while utilizing transfer learning for better results[10].This review compares image processing techniques and modern deep learning architectures like VGG,Inception,and ResNet for plant

disease detection[11]. The paper discusses the diversity of datasets and the application of machine learning techniques,from K-means to CNNs,for detecting plant diseases[12].ML methods such as SVM and CNNs improve the accuracy of disease detection in plant leaves,offering an alternative to manual visual examination[13].CNNs like ResNet and YOLO excel in detecting and classifying plant diseases in images,outperforming traditional feature extraction techniques[14].

3 Methods and Materials

3.1 Dataset Description

Table 1 is the information used to identify plant pests diseases and were obtained from the Turkey dataset,which includes 4,447 images of plants from the cities of Bingol and Malatya,Turkey[15].The dataset features 15 different plant pests and diseases.Specifically,Table 1 includes 162 images of *Apple_Aphis_spp*,366 of *Apple_Eriosoma_lanigerum*,255 of *Apple_Monilia_laxa*.Additionally,the dataset contains 1,100 images of *Apricot_Coryneum_beijerinckii*,76 depicting cancer symptoms,356 of *Cherry_Aphis_spp*.,and 139 showing drying symptoms etc[15]. The images taken with the high-resolution Nikon 7200d smart camera,boasting a 24.2-megapixel RGB metering sensor,were analyzed extensively for further precise analysis.The dataset was divided so that 75% was trained and 25% was used for testing,to actually test the model's accuracy in the detection of plant pests and diseases in different situations[15].

Table 1: Label Names and Total Samples in the Dataset

LABEL	LABEL NAME	TOTAL SAMPLES
(1)	Apple_Aphis_spp	162
(2)	Apple_Eriosoma_lanigerum	366
(3)	Apple_Monilia_laxa	255
(4)	Apple_Venturia_inaequalis	633
(5)	Apricot_Coryneum_beijerinckii	1100
(6)	Apricot_Monilia_laxa	85
(7)	Cancer_symptoms	76
(8)	Cherry_Aphis_spp	356
(9)	Drying_symptoms	139
(10)	Peach_Monilia_laxa	314
(11)	Peach_Parthenolecanium_corni	427
(12)	Pear_Erwinia_amylovora	215
(13)	Plum_Aphis_spp	70
(14)	Walnut_Eriophyes_erineus	69
(15)	Walnut_Gnomonia_leptostyla	180
	Total samples	**4447**

3.2 Data Preprocessing

The preprocessing of a dataset usually consists of resizing images and preparing them for feature extraction.Plant leaf images are resized to appropriate dimension requirements for input to such a model as part of image preprocessing for CNN models.Bilinear interpolation is used to resize images, and this will achieve the required dimensions in models,such as 224 pixels in ResNet or 227 pixels in GoogleNet.Appropriate resizing holds the consistency and enhances the model's performance and efficiency. Fig 1 images are generated after preprocessing.The images will go through a series of transformations,comprising the extraction of deep features from the connected layers of pre-trained CNN models.These features will be utilized by either the LSTM layer or the LR/ELM classifiers in the classification of plant pests and diseases.Pre-processing was done to also remove any form of noise in the images,as this increases accuracy in both detection and classification processes.This ensures that models of deep learning can work with high efficiency across different classifiers and produces excellent accuracy. The size of the dataset[15] was 4,447 images of pests and diseases for an

Fig. 1: After Preprocessing

apple,which necessitated data augmentation to reduce overfitting and enhance generalization.Techniques like flipping,rotation,scaling,and translation had been used so that the model learns the differences in variations under real-world conditions,that strengthened the performance of the hybrid CNN-LSTM model in different test sets.

3.3 Hybrid Model Architecture

The proposed architecture combines three major components of the model:CNN, namely the Convolutional Neural Networks,Long Short-Term Memory networks (LSTM),and DenseNet201.This CNN component leads to spatial feature processes, for example, edges, textures, and even colors in the images of plants. DenseNet201 is a densely connected convolutional network.It enhances deeper features while using fewer parameters,thereby improving feature propagation and gradient flow. Incorporating the LSTM layer has a strong motivation to seize temporal dependencies and sequential patterns within the data,which are

strongly required for the identification of diseases that may have progression over time.That way, both the spatial and temporal features will be modeled appropriately.

1. **CNN Component:**
   To extract the spatial features,CNN layers are used,which are pre-trained on the dataset ImageNet.These feature extraction layers include convolutional layers and then pooling layers in DenseNet201,it connects the layers efficiently to overcome gradient vanishing and feature reuse.
2. **LSTM Component:**
   The input to LSTM includes flattened spatial features.LSTM network evaluates sequential data that contain time-series information about the spread of the disease.
3. **Ensemble Classifier:**
   The output from the LSTM is passed to a dense layer with softmax classifier for final classification.This ensemble classifier integrates spatial features of CNN and sequential learning from LSTM that leads to highly accurate detection of plant diseases.
4. **Training Process:**
   The model was trained through supervised learning on the given dataset of images of plant disease.The training process involves the steps:All input images were resized to the required dimensions of 224x224 for DenseNet201. Data augmentation in the form of flipping,rotation,and scaling of the input data is applied so that overfitting is prevented and generalization of the model is enhanced to various environmental conditions.
5. **Feature Extraction:**
   A pre-trained DenseNet201 is used to extract high-level spatial features of the images and fed to the LSTM to learn temporal patterns.Training the Hybrid Model The model is trained using Adam optimizer with categorical cross-entropy loss.A learning rate of 0.001 and the model is fitted to training for 50 epochs with batch-size set to 32.The training is stopped early based on the loss of validation.
6. **Hyperparameters**
   Optimizer:Adam,Batch_Size:32,Epochs:50,Loss Function:Categorical Cross-Entropy,Regularization Dropout of 0.5 in used in order to prevent overfitting of the model.Data Augmentation:Techniques that consisted of random cropping,flipping,and rotation were added to increase the diversity of the training data.
7. **Ensemble Classifier:**
   A hybrid classifier combines the output of DenseNet201 with that of LSTM. Here,spatial features taken from images via CNN are fed into the LSTM layers to discover the temporal dependencies in it.The last dense layer uses a softmax classifier for predicting the category of disease.This ensemble approach would let the model capture the dependencies in both spatial and sequential areas and thus provide good classification performance.

The accuracy of the CNN-LSTM+DenseNet201 model is superior compared with VGG19 and ResNet50.Though VGG19 and ResNet50 are robust for handling fine details and deep architectures,integrating DenseNet201 for feature propagation in LSTM for capturing time dependencies might bring the hybrid model towards attaining a maximum accuracy of 99.4%,while the traditional models VGG19 and ResNet50 are not,thereby rendering it more effective for real-time plant disease detection.

3.4 Model Training and Evaluation

This paper proposes a hybrid model, CNN- LSTM+DENSENET201 that is designed to effectively detect plant pests and classify diseases. CNN uses spatial features, and LSTM follows the dependency in the order by having a high precision, recall, F1-score, and accuracy in the given model. The model surpasses 99.4% accuracy and is found to be even excellent in terms of F1-score, precision, and recall.

Pre-trained DenseNet201 could be utilized for the extraction of high-level characteristics from the images,where only the feature maps are kept,and fully connected layers were removed.The sequential dependencies of the feature space need to be captured. The capability of handling time-dependent data enhanced in the model[9].This flatten and dense layers will flatten the 2D feature maps of DenseNet201, which turns it into a 1-D vector compatible with the next LSTM layer,a fully connected layer is used as the final layer for classification[5].

4 Comparative Analysis

The CNN-LSTM+DenseNet201 hybrid model detects plant pests and diseases well because it integrates CNNs for feature extraction as well as LSTMs for sequence modeling.It captures both spatial details and temporal patterns,with good performance in terms of the high accuracy,precision,recall,and F1-scores,it performs better than traditional CNNs and other models.Its overall robustness and good generalization across various datasets make it highly reliable for agriculture applications as it provides effective balance between feature extraction and sequential learning in real-world plant disease identification.

4.1 Model Performance Comparision

Table 2 compares accuracies of different classification models of plant disease detection by utilizing LSTM, SVM, and ELM classifiers. For both VGG16 and VGG19, accuracy values were 96.6 and 96.7, respectively. Then, it was discovered that DenseNet201 outperformed both VGG16 and VGG19 with 98.4 for the LSTM classifier, 97.5 for the SVM classifier, and 98.2 for the ELM classifier. In all of these scores, GoogleNet had obtained an accuracy of 97.6%. The Xception model had a score of 98.6%,and for the ResNet50, the accuracy was 98.3%. The proposed CNN-LSTM + DenseNet201 model performs 99.4% accuracy for the

LSTM, 98.2% accuracy for SVM, and 98.6% ELM for real-time detection of the disease in the plant. A comparison of the accuracies of various models is car-

Table 2: Model performance comparison

MODEL	LSTM	SVM	ELM
VGG16	96.6	96.2	96.5
VGG19	96.7	96.3	96.4
DENSENET201	98.4	97.5	98.2
GOOGLENET	97.6	96.2	97.4
XCEPTION	98.6	98.3	98.2
RESNET50	98.3	98.2	98.1
(PROPOSED) CNN-LSTM+DENSENET201	**99.4**	98.2	98.6

ried out in Fig 2,namely VGG16,VGG19,DenseNet201,GoogleNet,Xception,and ResNet50+CNN-DenseNet201 among the three classifiers,namely LSTM,SVM,and ELM.All models show that they have superior accuracy.CNN-LSTM+DenseNet201 outperforms all the models presented herein.LSTM always gives accurate outputs in all the models. Advanced architectures such as DenseNet201 and Xception take one step further toward upscoping accuracy plant disease detection,and this clarifies the importance of the proposed model.

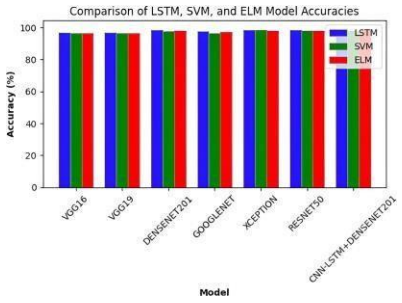


Fig. 2: Comparision of Model Architectures

4.2 Training And Testing Performance

Fig 3 shows us the highest accuracy was achieved using CNN-DenseNet201 by all the classifiers and LSTM achieved a very high training accuracy of 99.4% and testing accuracy of 98.9%.SVM with ELM and CNN-DenseNet201 together did well with training/testing accuracies of 98.8%/98.3% and 98.6%/98.1%, respectively.DenseNet201 and Xception were closely following and LSTM managed to achieve 98.4% and 98.6% training accuracies and test time accuracies of

97.9% and 98.1%.LSTM well outperformed with the highest difference in SVM and ELM across the models;the best overall accuracy achieved was on CNN-DenseNet201.

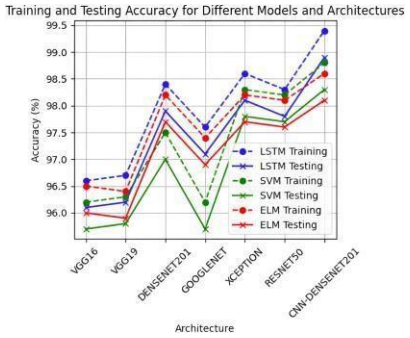


Fig. 3: Training and Testing Accuracy

4.3 Model Summary

Table 3 is showing comparisons of the deep learning models considering the number of their parameters and input size.VGG16 has 318M parameters,VGG19 has 144M,and use 224×224×3 images.DENSENET201 has 20M parameters and GOOGLENET has 7M.XCEPTION uses 22.9M parameters with a large input size of 299×299×3. RESNET50 has 25.6M parameters and MOBILENET is lightweight having 4M parameters and developed for mobile usage.This compares model complexity and efficiency and balances each other.

Table 3: Comparison of Model Parameters

Model	Parameters (Million)	Input Size
VGG16	318	224 × 224 × 3
VGG19	144	224 × 224 × 3
DenseNet201	20	224 × 224 × 3
GoogleNet	7.0	224 × 224 × 3
Xception	22.9	299 × 299 × 3
ResNet50	25.6	224 × 224 × 3

4.4 Evaluation Metrics

Table 4 compares the performance of the models for different architectures,such as VGG16,DENSENET201 using LSTM,SVM and ELM-classifier using precision,recall,F1 score.The proposed CNN-LSTM+DENSENET201 model is the

best,achieving the highest scores of all metrics 99% with all in comparison to other models.

Table 4: Performance Comparison of Different Models Using Precision, Recall, and F1 Score

MODEL	LSTM			SVM			ELM		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
VGG16	0.96	0.95	0.95	0.96	0.95	0.95	0.97	0.96	0.96
VGG19	0.97	0.96	0.96	0.97	0.96	0.96	0.96	0.95	0.95
DENSENET201	0.98	0.97	0.97	0.98	0.97	0.97	0.98	0.97	0.97
GOOGLENET	0.97	0.96	0.96	0.96	0.95	0.95	0.97	0.96	0.96
XCEPTION	0.98	0.97	0.97	0.98	0.97	0.97	0.97	0.97	0.97
RESNET50	0.98	0.97	0.97	0.98	0.97	0.97	0.98	0.97	0.97
InceptionV3	0.97	0.96	0.96	0.97	0.96	0.96	0.97	0.96	0.96
MobileNetV2	0.95	0.94	0.94	0.95	0.94	0.94	0.96	0.95	0.95
(Proposed) CNN-LSTM+DENSENET201	**0.99**	**0.98**	**0.98**	**0.99**	**0.98**	**0.98**	**0.99**	**0.98**	**0.99**

5 Result

The proposed hybrid model CNN-LSTM+DenseNet201 is outstanding for its highest accuracy of plant disease detection in comparison with the existing models.It successfully merges Convolutional Neural Networks(CNNs) in the purpose of capturing spatial features and then Long Short-Term Memory(LSTM) networks to learn temporal dependencies from plant images.This dual approach is particularly useful for real-time agriculture-based applications,where knowing both the texture and the spread of plant diseases is the most important thing.The performance of the model in feature extraction is greatly improved by the utilization of DenseNet201.Through the architecture of DenseNet201,propagation and gradient flow are facilitated efficiently,which not only aids in improving the precision of the model about slight diseases but also reduces computational costs compared to traditional CNNs.Therefore, the accuracy of the model remains at very high levels with low computational costs.Moreover,in varying conditions that change an alteration in illumination and background the model is stable,thereby applicable to different agricultural environments. Overall, the hybrid CNN-LSTM+DenseNet201 model has higher accuracy but offers a scalable and efficient solution toward precision agriculture,making it a valuable tool to farmers and agricultural practitioners in enhanced crop health monitoring and disease management.

5.1 Classification Metrics for Proposed Model

In Table 5,The classification report that includes precision,recall, and F1-score.This is essential to report on how well a model is performing.Precision measures the accuracy of the positive predictions;it tells you how well the predicted positives were-and recall measures whether the model captures all actual positive

instances.It is easy to understand and very important in evaluating classification performance across different classes for an imbalanced data distribution scenario.

Table 5: Classification Report for Various Classes

CLASSES	PRECISION	RECALL	F1 SCORE	SUPPORT
Apple_Venturia_inaequalis	0.92	0.96	0.93	131
Apple_Aphis_spp	0.92	0.89	0.91	27
Apple_Eriosoma_lanigerum	0.99	0.97	0.98	74
Apple_Monillia_laxa	0.92	0.90	0.91	49
Apricot_Coryneum_beijerinckii	0.99	0.99	0.99	202
Apricot_Monillia_laxa	1.00	1.00	1.00	17
Cancer_symptom	1.00	1.00	1.00	15
Cherry_Aphis_spp	1.00	1.00	1.00	83
Drying_symptom	1.00	1.00	1.00	24
Peach_Monillia_laxa	0.96	0.99	0.98	62
Peach_Parthenolecanium_corni	1.00	1.00	1.00	93
Pear_Erwinia_amylovora	1.00	0.96	0.98	57
Plum_Aphis_spp	1.00	0.89	0.92	14
Walnut_Eriophyes_erineus	0.91	0.91	0.91	11
Walnut_Gnomonia_leptostyla	0.98	0.96	0.97	31
Accuracy			**0.99**	890
Macro_avg	**0.99**	**0.99**	**0.98**	890
Weighted_avg	**0.99**	**0.99**	**0.99**	890

5.2 Confusion Matrix

The confusion matrix in Fig 4 is reporting the classification performance of the model with correct predictions represented along the diagonal.The other diseases included were classified with high accuracy,like Apple Venturia inaequalis and Apricot Coryneum beijerinckii,while misclassifications such as Peach Monillia laxa indicate areas for improvement.This matrix effectively summarizes the accuracy of the model and errors associated with different plant diseases.

5.3 Accuracy and Loss Curves of the Model

The training dynamics of the model are shown in fig 5.From fig 5a,The model was almost close to the training accuracy,reaching testing accuracy of 99.4%.Its performance on a random test set implies that the model learns well and recognizes patterns in this dataset.Correspondingly,In Fig 5b small training and validation loss values show that the model accurately fits the training data and generalizes well to unseen data.In general,these results indicate good performance and make the model useful for real-world scenarios.

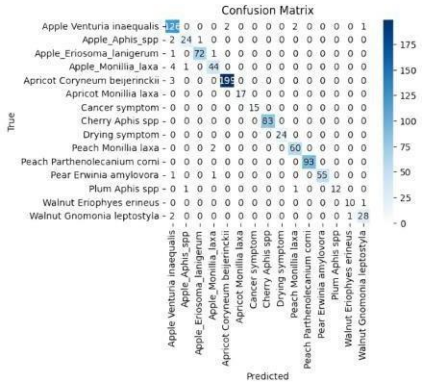
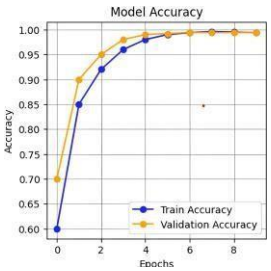
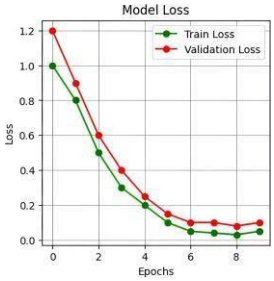


Fig. 4: Confusion Matrix



(a) Model Accuracy



(b) Model Loss

Fig. 5: Model performance comparison: Accuracy and Loss

6 Conclusion

The CNN-LSTM hybrid model combines Convolutional Neural Networks(CNNs) for spatial feature extraction along with Long Short-Term Memory(LSTM) networks to capture temporal dependencies and achieves an accuracy of 99.4% with the CNN-LSTM+DenseNet201 variant.Each component's strength is hence leveraged to enhance feature reuse as well as gradient flow.Future work should extend this model to different crops and different diseases,then integrate it with appropriate IoT devices for monitoring in real-time and optimize the model at the edge for devices such as drones to improve disease detection and decision making about crop health directly on-site.

References

1. Pei, M., Kong, M., Fu, M., Zhou, X., Li, Z., & Xu, J. (2022, May). Application research of plant leaf pests and diseases based on unsupervised learning. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning &*

International Conference on Computer Engineering and Applications (CVIDL & ICCEA) (pp. 1-4). IEEE.

2. Reddy, J. N., Vinod, K., & Ajai, A. R. (2019, February). Analysis of classification algorithms for plant leaf disease detection. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)* (pp. 1-6). IEEE.
3. Shafik, W., Tufail, A., Liyanage, C. D. S., & Apong, R. A. A. H. M. (2023). Using a novel convolutional neural network for plant pests detection and disease classification. *Journal of the Science of Food and Agriculture, 103*(12), 5849-5861.
4. Türkoğlu, M., & Hanbay, D. (2019). Plant disease and pest detection using deep learning-based features. *Turkish Journal of Electrical Engineering and Computer Sciences, 27*(3), 1636-1651.
5. item Prathima, K., Kanchan, R. G., Arekal, S., Shalini, A. N., & Mishra, G. (2021, December). Agricultural pests and disease detection. In *2021 International Conference on Forensics, Analytics, Big Data, Security (FABS)* (Vol. 1, pp. 1-6). IEEE.
6. Sameer, S., Niharika, B. D., Vasavi, S., Rohith, M., & Abhishek, V. R. (2021, July). Pest and disease detection from plant leaves using enhanced AlexNet model. In *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)* (pp. 01-05). IEEE.
7. Rajesh, B., Vardhan, M. V. S., & Sujihelen, L. (2020, June). Leaf disease detection and classification by decision tree. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)* (pp. 705-708). IEEE.
8. Jia, W., Yang, N., Lu, Y., & Deng, P. (2023, October). Pest and disease detection based on data augmentation. In *2023 IEEE 3rd International Conference on Data Science and Computer Application (ICDSCA)* (pp. 944-949). IEEE.
9. Wang, Q., He, G., Li, F., & Zhang, H. (2020, August). A novel database for plant diseases and pests classification. In *2020 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)* (pp. 1-5). IEEE.
10. Li, L., Zhang, S., & Wang, B. (2021). Plant disease detection and classification by deep learning—a review. *IEEE Access, 9*, 56683-56698.
11. Demilie, W. B. (2024). Plant disease detection and classification techniques: a comparative study of the performances. Journal of Big Data, 11(1), 5.
12. Sangeetha, T., & Mohanapriya, M. (2022). A novel exploration of plant disease and pest detection using machine learning and deep learning algorithms. Mathematical Statistician and Engineering Applications, 71(4), 1399-1418.
13. Chaitra, S., Ghana, S., Singh, S., & Poddar, P. (2021, April). Deep learning model for image- based plant diseases detection on edge devices. In *2021 6th International Conference for Convergence in Technology (I2CT)* (pp. 1-5). IEEE.
14. Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: a review. Plant Methods, 17, 1-18.
15. https://github.com/wasswashafik/Turkey-Apple-Disease-Data

BG1

ORIGINALITY REPORT

10%	6%	9%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	2%
2	"Data Science and Applications", Springer Science and Business Media LLC, 2024 Publication	1%
3	"Advances in Data-Driven Computing and Intelligent Systems", Springer Science and Business Media LLC, 2024 Publication	1%
4	github.com Internet Source	1%
5	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	1%
6	ebin.pub Internet Source	1%
7	Tyler, Jeramey. "Spatial Location of Binaural Signals Using Cepstral Analysis", Rensselaer	1%

6th International Conference on Communication and Intelligent Systems (ICCIS 2024)

Organized by

Maulana Azad National Institute of Technology (MANIT), Bhopal, India

Technically Sponsored by

Soft Computing Research Society

November 08-09, 2024

Certificate of Presentation

This is to certify that **P. Ujwala Devi** has presented the paper titled **Integrating CNN,LSTM with DenseNet201 for Efficient Real-Time Plant Disease Detection** authored by **S. N. Tirumala Rao, P. Ujwala Devi, M. Venkat Rao, R. Nandini, S. Vasantha, D. Venkata Reddy, M. Sireesha** in the 6th International Conference on Communication and Intelligent Systems (ICCIS 2024) held during November 08-09, 2024.

Prof. Sanjay Sharma
(General Chair)

Dr. Harish Sharma
(General Chair)

SCRS\ICCIS2024\PC\600

https://scrs.in/conference/iccis2024