Detailed Tomato Leaf Disease Detection using Transfer Learning.

Project Report submitted in the partial fulfilment

Of Bachelor of Technology

In

Information and Technology Engineering

By

Kunj Jariwala(A024) Meet Patel (A041) Kevin Mehta (A112)

Under the supervision of

Dr. Ketan Shah
(HOD, IT Department, MPSTME)

SVKM's NMIMS University

(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING Vile Parle (W), Mumbai-56

2023-24

CERTIFICATE

This is to certify that the project entitled **"Detailed Tomato Leaf Disease Detection using Transfer Learning"**, has been done by **A024 Kunj Jariwala, A041 Meet Patel and A112 Kevin Mehta** under my guidance and supervision & has been submitted in partial fulfilment of degree of Bachelor of Technology in Information Technology of MPSTME, SVKM's NMIMS (Deemed-to-be University), Mumbai, India.

Mentor: Dr. Ketan Shah

(HOD, IT Department)

Examiner

Date:31-10-23

Place: Mumbai

(HOD)name and sign

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the Project Work undertaken during our B.Tech. Final Year. We owe a special debt of gratitude to our Project Mentor **Dr. Ketan Shah**, communications for his constant support and guidance throughout the course of our major project. It is only his cognizant efforts that our endeavors have seen the light of the day. We thank him for believing in this project and motivating us to our utmost potential through constant advice and

remarks.

We would not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our peers for their guidance and support in the completion of the project.

NAME	ROLL NO	SAP ID
Kunj Jariwala	A024	70012000200
Meet Patel	A041	70012000241
Kevin Metha	A112	70042000135

B.Tech. (IT)

Semester VII

Mumbai Campus

ABSTRACT

The "Tomato Leaf Diseases Detection" represents a robust solution for identifying and diagnosing diseases affecting tomato plants. Early detection and effective treatment are crucial to mitigate these effects. This innovative web application harnesses the power of deep learning and transfer learning techniques, with recently released MobileNetV2 model architecture, to offer users an efficient tool for disease detection in tomato leaves. The application is capable of distinguishing between healthy and nine different diseases that afflict tomato plants.

In this project, the MobileNetV2 model serves as a pre-trained neural network that has been fine-tuned on a dataset of over 30,000 tomato leaf images. The transfer learning approach leverages the model's ability to extract intricate patterns and features from images, resulting in accurate and reliable disease detection. Users can simply input an image of a tomato leaf via the web application and provide a diagnosis using the transfer learning model .

By combining the strengths of deep learning and user-friendly interface design, this project addresses the critical challenge of disease management in tomato plants while promoting sustainable agricultural practices.

Table of Content

Chapter 1 Introduction

Tomatoes are a staple in diets worldwide and play a crucial role in global food security. However, the tomato industry in India has faced challenges due to the surge in tomato prices, primarily attributed to the susceptibility of tomato plants to various diseases.

These diseases can substantially reduce tomato leaf yields and the quality of tomatoes. Addressing this issue requires early detection and effective management of these diseases.

Traditionally, disease detection has relied heavily on the visual inspection of tomato plants by experienced agronomists or farmers. While this method is valuable, it has limitations. It can be subjective, and not all diseases are easily detectable at their initial stages, especially on large-scale farms where manual inspections can be time-consuming and impractical. In recent years, technological advancements have offered innovative solutions to these challenges. Computer vision and machine learning have made it possible to automate the process of disease detection in tomato plants. This automation is achieved by leveraging deep learning techniques and a specific model called

MobileNetV2, which is known for its efficiency and exceptional performance in image classification tasks.

MobileNetV2 is used as a transfer learning model in this project. Transfer learning involves taking a pre-trained deep learning model and fine-tuning it on a specialized dataset—in this case, a vast collection of tomato leaf images. This process enables the model to learn intricate patterns and features associated with various tomato leaf diseases, enhancing its ability to identify them accurately.

The primary goal of the Tomato Leaf Diseases Detection Web Application is to empower users, including farmers and horticulturists, with a user-friendly tool for disease detection and management. The application offers several key features:

1. **Disease Identification:** Users can easily upload multiple images of tomato leaves to the application. The MobileNetV2 model, working behind the scenes,

swiftly analyzes these images and accurately identifies the presence of one or more of the nine common diseases known to affect tomato plants.

2.Disease Description: Beyond disease identification, the application provides users with detailed descriptions of the detected diseases. This educational component is essential for users to understand the nature of the problem, enabling them to make informed decisions regarding disease management.

3.Preventive Measures: Perhaps the most critical aspect of the application is the guidance it offers on disease prevention and management. Users receive practical recommendations on how to prevent and mitigate the identified diseases. This proactive approach promotes sustainable and healthy tomato cultivation practices, ultimately benefiting both farmers and consumers.

1.1 Background of the Project Topic:

Tomatoes are a fundamental tomato leaf in global agriculture, playing a crucial role in human nutrition and food security. India, a major contributor to global tomato production, faces a persistent challenge in the form of fluctuating tomato prices. These price fluctuations are closely tied to the susceptibility of tomato plants to various diseases, which can significantly reduce tomato leaf yields and tomato quality. Addressing this issue hinges on the early detection and effective management of these diseases.

Traditional methods of disease detection in tomato plants heavily rely on visual inspections conducted by experienced agronomists and farmers. While these manual inspections provide valuable insights, they suffer from inherent limitations. They are subject to human subjectivity and may not identify diseases in their early stages, particularly on large-scale farms where manual inspections can be impractical.

In recent years, technological advancements have opened up new possibilities to overcome these challenges. Automation of disease detection in tomato plants has become a reality, driven by advances in deep learning techniques. A notable example of this technological leap is the utilization of MobileNetV2, a potent deep learning

model known for its efficiency in image classification. Our project aspires to empower farmers and horticulturists with a user-friendly tool for disease detection and management. The application encompasses features such as disease identification, detailed disease descriptions, and practical recommendations for disease prevention and mitigation. By merging advanced technology with agriculture, this project aims to enhance food security, economic sustainability, and environmental stewardship within the realm of tomato cultivation. Subsequent sections will delve into the technical intricacies and outcomes of this pioneering initiative.

1.2 Motivation and Scope of the Report:

The driving force behind our Tomato Leaf Diseases Detection Web Application project stems from a compelling need to address critical challenges faced by the tomato industry in India. Tomatoes serve as a dietary staple globally and are pivotal for ensuring food security. However, the industry grapples with volatile tomato prices, primarily attributed to the vulnerability of tomato plants to a spectrum of diseases. These diseases pose a significant threat by substantially reducing tomato leaf yields and the quality of tomatoes. Traditional disease detection methods, often reliant on manual inspections, suffer from subjectivity and inefficiency. Notably, not all diseases manifest visibly during their initial stages, making early detection crucial yet challenging, especially on large-scale farms.

In recent years, technology has offered innovative solutions to these challenges. With the advent of computer vision and machine learning, we can now revolutionize disease detection in tomato plants. The project leverages the power of deep learning and specifically employs the MobileNetV2 model, renowned for its efficiency in image classification tasks. MobileNetV2 is integrated as a transfer learning model, fine-tuned using an extensive dataset of tomato leaf images. This empowers the model to discern intricate patterns and features associated with various tomato leaf diseases, significantly enhancing its capacity for accurate identification.

Our primary goal is to empower users, including farmers and horticulturists, with a user friendly, technology-driven web application tool for effective disease detection and management. The project achieves this through an amalgamation of cutting-edge technology, education, and sustainability. Users can effortlessly upload multiple tomato leaf images to the web application, which operates seamlessly with the MobileNetV2 model in the background. Beyond simple identification, the application educates users about the detected diseases, offering comprehensive descriptions of their causes, symptoms, and potential impacts. Most importantly, it provides users with actionable recommendations for disease prevention and management, fostering sustainable agricultural practices.

The scope of the Tomato Leaf Diseases Detection project is far-reaching and forward looking, with a core emphasis on transforming disease management in the tomato industry. The project's focal point is the development of a robust disease identification system. This system, underpinned by deep learning and MobileNetV2, facilitates rapid and precise disease detection in tomato plants. However, it doesn't stop there. The project extends its scope into the realm of education, ensuring that users, including farmers and horticulturists, gain a profound understanding of the detected diseases. This understanding includes insights into disease causes, symptoms, and potential impacts on tomato leaves.

1.3 Problem Statement:

The central challenge addressed by this research project is the critical need to revolutionize disease management in the tomato industry, particularly in India, with a primary focus on the welfare of farmers. Farmers in the tomato industry grapple with volatile tomato prices and the susceptibility of tomato plants to various diseases, posing significant threats to their livelihoods. Traditional methods of disease detection, reliant on manual labor and visual inspection, are laden with limitations. They are often labor-intensive, time-consuming, and prone to subjectivity, making them impractical for large-scale farming operations. Moreover, access to expert agronomists for visual inspection is constrained in remote or resource-constrained regions, leaving many farmers without vital guidance.

Furthermore, these traditional methods are not always effective at identifying diseases in their early stages, leading to substantial tomato leaf losses and financial setbacks for farmers. In response to these pressing challenges, there is an urgent need to harness technological innovations, such as deep learning and MobileNetV2, to automate disease detection in tomato plants. This automation can enhance accuracy, timeliness, and accessibility, directly benefiting farmers.The proposed solution aims to develop a user-friendly Tomato Leaf Diseases Detection Web Application, empowering farmers and horticulturalists with a reliable tool for disease detection and management. Beyond identification, this application serves as an educational resource, offering comprehensive disease descriptions and proactive preventive measures. It is designed for accessibility, offering a cost-effective and efficient alternative to traditional methods, ultimately bolstering tomato leaf production efficiency and contributing to food security and economic prosperity for farmers.

In essence, the problem statement underscores the urgency of overcoming the limitations of traditional disease detection methods, directly benefiting farmers by protecting their tomato crops and livelihoods, and highlights the immense potential for technology-driven solutions to fortify the tomato industry, ensure tomato leaf protection, and promote sustainability.

1.4 Salient Contribution of the Project:

Our project stands as a beacon of progress in addressing the formidable challenges that loom over the tomato industry, with a particular emphasis on the Indian landscape. Leveraging a substantial dataset comprising 30,000 images and adopting cutting-edge technologies, the system has redefined the landscape of disease detection in tomato plants. Its heightened precision is a direct boon to farmers, whose livelihoods are intimately connected to the health of their tomato crops. The user-friendly Tomato Leaf Diseases Detection Web Application is a testament to technological innovation, extending a robust tool not limited to any specific category of users.

This application empowers not only farmers but a broad spectrum of individuals, ranging from those tending to backyard gardens to agricultural enthusiasts. It enables swift and accurate disease identification, but it doesn't end there. This application doubles as a comprehensive educational resource, endowing users with profound insights into the causes, symptoms, and potential impacts of tomato leaf diseases. Beyond knowledge, it delivers actionable recommendations for disease prevention and management, underscoring sustainable agricultural practices that benefit not just farmers but the entire ecosystem. The technology-driven solution removes the shackles of geographical constraints and limited resources, rendering disease detection scalable and accessible. These contributions, underpinned by technology, education, and sustainability, converge to fortify the tomato industry, ensuring the security of tomato crops and economic prosperity for farmers and consumers. It is a profound stride towards a healthier, more secure, and more sustainable food supply chain, accessible to all who seek to partake in its benefits.

Chapter 2 Literature survey

Traditional methods for detecting tomato leaf diseases relied on human scouting and laboratory tests, which are time-consuming and expensive. Computer vision and machine learning techniques have emerged as powerful automated approaches for plant disease detection and classification in recent years. Earlier studies explored hand-crafted feature extraction methods like color histograms, texture analysis, morphological operations etc combined with classifiers like SVM and KNN. However, these had limitations in accuracy and generalization capability. Deep learning models, especially convolutional neural networks (CNNs), have shown tremendous success in complex visual recognition tasks across various domains. Researchers have started leveraging deep CNNs for plant disease identification as well, given their ability to learn robust features directly from pixel data. Initial works experimented with shallow CNN architectures on relatively small datasets of leaf images. Recent papers have adopted transfer learning strategies by fine-tuning advanced pre-trained models like VGG, ResNet, Inception etc. on larger plant disease image datasets. This has significantly improved the performance.

Data augmentation techniques like flipping, rotation and PCA-color augmentation are commonly used to expand datasets. Class activation maps are utilized for interpreting predictions. Most studies have focused on classifying a few common tomato leaf diseases. There is scope for identifying more disease classes, detecting multiple diseases per leaf, and conducting multi-crop comparative studies.

In summary, deep CNNs have emerged as a promising technique for automated diagnosis of tomato plant diseases from images. With sufficient labeled training data, they can enable timely and accurate detection. Future work should focus on improving generalization capability across different crop varieties, growth stages and capture conditions. There are opportunities to develop end-to-end systems from disease detection to personalized recommendation of treatment measures.**[1]**

This paper addresses a critical challenge in deep learning-based image classification, which is the requirement for a large number of training samples. The study focuses on the practical issue of obtaining sufficient samples for several disease categories, which has historically hindered accurate classification.

The primary objective of this paper is to enhance the accuracy and robustness of image classification, particularly in the context of imbalanced datasets. The authors employ a combination of weighted random sampling algorithms and data augmentation techniques during the image pre-processing stage to balance different category samples. This step is crucial in ensuring a consistent and representative dataset for training.

Additionally, the study introduces a novel CNN architecture based on MobileNetV2, coupled with a targeted dropout algorithm. This approach is designed to enhance the robustness of the conventional model. The results demonstrate that the proposed model achieves performance levels very close to those of the conventional CNN architecture. Furthermore, the detrimental effects of Gaussian and Salt&Pepper noise are mitigated, and the technique exhibits significant performance improvements, especially on complex background samples, compared to traditional deep learning methods.

In summary, this paper presents a comprehensive approach to address the challenges associated with imbalanced datasets in deep learning-based image classification. The integration of weighted random sampling, data augmentation, and a targeted dropout algorithm, along with the use of MobileNetV2, yields a model with enhanced robustness and performance. This methodology holds potential for applications where sample availability is limited, and can lead to more accurate and reliable image classification results.**[2]**

This paper addresses the critical issue of tomato diseases and their impact on food production safety. Recognizing the significance of automated pre-crop disease identification for farmers, the study proposes a method for identifying tomato leaf diseases using an optimized MobileNetV2 model.

To facilitate this research, a dataset comprising 20,400 images of tomato diseases was meticulously compiled from images captured in greenhouses and sourced from the PlantVillage database. The optimized MobileNetV2 model was subsequently trained on this dataset, resulting in the development of a robust classification model for tomato leaf diseases.

Following experimental validation, the model demonstrated an impressive average recognition accuracy of 98.3%, with a recall rate of 94.9%. Notably, this represents a notable improvement of 1.2% and 3.9% over the original model, respectively. Furthermore, the model exhibited an average prediction speed of approximately 76 milliseconds per image, outperforming the original model by 2.94%.

To further assess the performance of the optimized MobileNetV2 model, comparative analysis was conducted against feature extraction network models such as Xception, Inception, and VGG16 using transfer learning. The results underscored the superiority of the proposed model, showcasing recognition accuracy improvements ranging from 0.4 to 2.4 percentage points compared to the aforementioned models.

The findings of this study hold significant implications for both the agricultural and precision farming sectors. By providing a reliable method for identifying tomato diseases, this research offers invaluable technical support to farmers, enabling them to enhance production and income. Moreover, it contributes to the broader endeavour of plant growth monitoring in the context of precision agriculture.**[3]**

This study presents an innovative approach for the early detection of tomato leaf spot, leveraging the MobileNetv2-YOLOv3 model to strike a balance between accuracy and real-time detection. The proposed method introduces the GIoU bounding box regression loss function to enhance the precision of tomato gray leaf spot recognition.

A lightweight network model, MobileNetv2-YOLOv3, is devised, with MobileNetv2 serving as the backbone network. This design choice facilitates seamless integration with mobile terminals. Additionally, a pre-training strategy combining mixup training and transfer learning is employed to enhance the model's generalization capabilities.

The study conducts a thorough statistical analysis of images captured under four distinct conditions. Model performance is evaluated using metrics like the F1 score and Average Precision (AP), and compared against Faster-RCNN and SSD models. The experimental results demonstrate a substantial improvement in the recognition effectiveness of the proposed model.

In the test dataset comprising images taken under well-lit conditions without leaf shelter, the F1 score and AP value achieve an impressive 94.13% and 92.53%, respectively, with an average Intersection over Union (IOU) value of 89.92%. Across all test sets, the F1 score and AP value stand at 93.24% and 91.32%, with an average IOU value of 86.98%.

Furthermore, the model showcases exceptional computational efficiency. On a GPU, it achieves a remarkable object detection speed of 246 frames per second. For extrapolation of a single 416 × 416 image, the speed is 16.9 milliseconds. On a CPU, the detection speed reaches 22 frames per second, with an extrapolation speed of 80.9 milliseconds. Additionally, the model occupies a modest 28 megabytes of memory.

This study not only introduces a highly effective approach for tomato leaf spot detection but also highlights the model's efficiency in real-time applications, making it a promising tool for agricultural practices.**[4]**

This article addresses a critical concern in agriculture, focusing on the early detection of leaf diseases in tomatoes, a widely consumed fruit known for its high nutritional value. These diseases can lead to significant damages, making early detection crucial. Conventional methods involving human experts for disease identification are not only costly and time-consuming but also subjective in nature. Here, computer vision emerges as a promising solution for early detection.

The study introduces a computer vision-based system capable of classifying seven distinct categories of tomato diseases: bacterial spot, early blight, late blight, leaf mold, septoria leaf spot, spider mites, and target spots. The choice of the optimized MobileNetV2 architecture demonstrates an intention to balance computational efficiency with detection performance.

To further enhance the model's performance, a modified gray wolf optimization approach is applied to fine-tune the MobileNetV2 hyperparameters. The validation process employs both standard internal and external validation methods, ultimately yielding an impressive classification accuracy of 98%.

These results underscore the significant potential of the proposed framework for early detection of tomato leaf diseases. The implications are far-reaching, as this technology holds the promise of mitigating substantial agricultural losses and contributing to a more sustainable and productive agricultural industry.**[5]**

This study addresses a critical concern in tomato production, namely, the impact of diseases like bacterial spot, early blight, and leaf mold on overall yield. Swift identification and timely treatment of these diseases are pivotal in minimizing production losses. Despite a wealth of research in this area, the study identifies a notable gap in the literature—a lack of comparative analysis between traditional Machine Learning (ML) and Deep Learning (DL) algorithms for tomato disease classification.

The researchers systematically evaluate various ML and DL algorithms using the PlantVillage tomato dataset, aiming to pinpoint the most effective models for this specific classification problem. For ML implementations, the study employs a range of manual disease feature extraction techniques. In total, 52 texture features are extracted using Local Binary Pattern (LBP) and Gray Level Co-occurrence Matrix (GLCM) methods, along with 105 color features obtained through color moment and color histogram methods.

Among these extraction methods, the COLOR+GLCM approach stands out as the most effective. When comparing the performance metrics (accuracy, precision, recall, F1 score), it becomes evident that the DL networks (AlexNet, VGG16, ResNet34, EfficientNet-b0, and MobileNetV2) consistently outperform the tested ML algorithms (Support Vector Machine (SVM), k-Nearest Neighbor (kNN), and Random Forest (RF)).

These findings not only shed light on the comparative performance of ML and DL algorithms for tomato disease classification but also establish MobileNetV2 as an exceptionally effective tool for this particular task. This study represents a significant step forward in advancing the precision and efficiency of disease identification in tomato production, with potential far-reaching benefits for agricultural practices.**[6]**

The paper begins with an introduction to the tomato, highlighting its origins and economic significance. It emphasizes the susceptibility of tomato plants to various diseases, including leaf mold, late blight, and mosaic virus, which pose significant challenges in plant management. The difficulty in completely eradicating viral diseases underscores the importance of early and accurate disease detection.

To address this challenge, the paper proposes a computer vision-based approach for disease identification, focusing on capturing leaf images and analyzing them to assess disease likelihood. The study employs deep learning techniques, specifically utilizing the MobileNet V2 architecture, known for its compactness and efficiency. The model is fine-tuned to target three specific types of tomato diseases.

The evaluation of the proposed algorithm involves testing on a substantial dataset comprising 4,671 images sourced from the PlantVillage dataset. The findings demonstrate that MobileNet V2 achieves a remarkable disease detection accuracy exceeding 90%.

This literature survey highlights the critical role of accurate and timely disease detection in tomato plants, emphasizing the potential of computer vision techniques, particularly deep learning with MobileNet V2, in addressing this challenge. The reported high accuracy rates suggest promising implications for practical implementation in agricultural settings.**[7]**

The paper introduces a novel deep learning-based methodology for distinguishing between different tomato seed cultivars. The research employs a two-scenario approach to achieve this objective.

In the first scenario, images of seeds from four distinct tomato cultivars (Sacher F1, Green Zebra, Pineapple, and Ozarowski) are captured. Each seed image is then cropped from the original image and saved as a separate entity. Data augmentation techniques are utilized to expand the dataset. These seed images are subsequently classified using four different Convolutional Neural Network (CNN) models: ResNet18, ResNet50, GoogleNet, and MobileNetV2. The MobileNetV2 model achieves the highest classification accuracy, reaching 93.44%.

In the second scenario, 1280 deep features extracted from MobileNetV2 serve as inputs for a Bidirectional Long Short-Term Memory (BiLSTM) network. The classification carried out with the BiLSTM network yields an impressive accuracy of 96.09%.

These results demonstrate that the proposed deep learning-based methodology offers a rapid and accurate means of distinguishing between different tomato seed cultivars. This study represents a significant advancement in the field of seed cultivar differentiation, introducing an innovative approach that integrates deep learning into tomato seed image analysis. The methodology holds substantial promise as a comprehensive and practical procedure for tomato seed classification. This development has the potential to greatly benefit agricultural practices and seed breeding efforts.**[8]**

Tomato is one of the most important vegetables worldwide. It is considered a mainstay of many countries' economies. However, tomato crops are vulnerable to many diseases that lead to reducing or destroying production, and for this reason, early and accurate diagnosis of tomato diseases is very urgent. For this reason, many deep learning models have been developed to automate tomato leaf disease classification. Deep learning is far superior to traditional machine learning with loads of data, but traditional machine learning may outperform deep learning for limited training data.

The authors propose a tomato leaf disease classification method by exploiting transfer learning and features concatenation. The authors extract features using pre-trained kernels (weights) from MobileNetV2 and NASNetMobile; then, they concatenate and

reduce the dimensionality of these features using kernel principal component analysis. Following that, they feed these features into a conventional learning algorithm. The experimental results confirm the effectiveness of concatenated features for boosting the performance of classifiers. The authors have evaluated the three most popular traditional machine learning classifiers, random forest, support vector machine, and multinomial logistic regression; among them, multinomial logistic regression achieved the best performance with an average accuracy of 97%.**[9]**

This paper addresses the crucial role of agriculture in many countries, particularly the cultivation of tomatoes, a prominent crop in regions with ample water resources. The study underscores the significance of early disease identification to safeguard crop yields. The proposed method leverages Convolutional Neural Networks (CNN) and image processing techniques for this purpose.

The research employs a dataset from an open repository for training and testing. The algorithm developed demonstrates the ability to identify nine distinct diseases affecting tomato plants in their early stages. Images of tomato leaves serve as input for processing and classification.

The study further refines the approach by evaluating various CNN architectures, including VGG, ResNet, Inception, Xception, MobileNet, and DenseNet. Through a comprehensive analysis, an optimal model is derived. The performance of each architecture is compared using metrics such as accuracy, loss, precision, recall, and area under the curve (AUC).

In summary, this paper introduces an innovative method for the early identification of diseases in tomato plants, combining CNNs and image processing. The research highlights the effectiveness of different CNN architectures and emphasizes the importance of accurate disease identification in preserving crop yields.**[10]**

The paper emphasizes the critical need for early diagnosis of plant diseases due to their significant social, ecological, and economic impact. Current methods are deemed complex, time-consuming, and labor-intensive. The study focuses on

classifying nine tomato plant leaf diseases, along with healthy leaves, utilizing deep learning and novel ensemble architectures.

A substantial dataset comprising 18,160 images is employed in the study. Alongside the proposed two new convolutional neural network (CNN) models, four established CNN models (MobileNetV3Small, EfficientNetV2L, InceptionV3, and MobileNetV2) are also utilized. The authors implement a fine-tuning process on the newly proposed CNN models and subsequently conduct hyperparameter optimization using the particle swarm optimization algorithm (PSO). Grid search is employed to further optimize the weights of these architectures. Ensemble models are then formed, including triple and quintuple configurations, and the datasets are classified through five-fold cross-validation.

The experimental results highlight the notable advantages of the proposed ensemble models, characterized by swift training and testing times, and superior classification performance, achieving an impressive accuracy of 99.60%. The research holds significant promise in enabling experts to promptly and efficiently detect plant diseases, thereby mitigating the spread of infections.

In conclusion, this study presents a highly effective approach for early plant disease detection using deep learning and ensemble architectures. The results demonstrate exceptional accuracy and efficiency, providing a valuable tool for experts in the field of plant disease management.**[11]**

This study focuses on enhancing the detection and treatment of tomato diseases, ultimately leading to improved crop quality and yield. The research introduces a novel approach for real-time, non-destructive detection of tomato diseases, utilizing an adapted version of the MobileNetV3 convolutional neural network.

The lightweight MobileNetV3 is initially employed for transfer learning on a large-scale dataset, facilitating the accurate classification of tomato leaf diseases even with limited sample data. This transfer learning approach helps overcome overfitting issues associated with insufficient data and reduces network training time. Comparative

experiments with standard deep convolutional network models (VGG16, ResNet50, and Inception-V3) reveal that MobileNetV3 outperforms its counterparts.

Furthermore, the study investigates the impact of changes in loss function and data augmentation methods on disease identification using MobileNetV3. Testing reveals that employing Mixup augmentation along with the focal loss function leads to improved model accuracy, achieving an average test recognition accuracy of 94.68% across ten types of tomato diseases.

To further enhance the MobileNetV3 model's performance, the study introduces dilated convolution layers with expansion rates of 2 and 4, incorporates a perceptron structure after deep 5×5 convolutions, and applies the GLU gating mechanism activation function. The resulting model achieves an impressive average test recognition accuracy of 98.25%. Notably, the model is compact, with a data size of 43.57 MB, and demonstrates rapid image detection, taking only 0.27 seconds per image.

Through ten-fold cross-validation, the model's recognition accuracy remains consistently high at 98.25%, affirming its stability and reliability. Overall, this study significantly enhances the efficiency of tomato disease detection, reducing the time and resource costs associated with disease image analysis.**[12]**

This paper underscores the pivotal role of agriculture in the Indian economy and highlights the challenges posed by plant diseases, often exacerbated by a lack of knowledge regarding their identification and treatment. The integration of artificial intelligence (AI) in agriculture is identified as a critical solution, with a creative approach to implementing technology in farming practices. Swift and accurate detection of damaged leaves is deemed crucial for crop preservation and productivity.

The study introduces a plant disease detection system utilizing deep Convolutional Neural Network (CNN) algorithms. This system efficiently identifies symptoms and indications present on plant leaves and stems, ultimately fostering the growth of healthy vegetation on farms. A deep learning-based model is employed to extract information from images, discerning between healthy and diseased leaves.

The paper reviews existing methods for plant disease detection and emphasizes the significant advancements achieved through deep learning-based CNNs. The study particularly focuses on the effectiveness of large-scale architectures like ResNet and MobileNetV2. The MobileNetV2 CNN model, trained and tested using data from the Kaggle Website, outperforms previous CNN-based models, achieving an impressive 99% accuracy rate.

The research not only contributes to improved plant disease detection but also advocates for increased agricultural food production. This underscores the broader impact of the proposed approach in enhancing agricultural productivity, which is crucial for sustaining and advancing the Indian economy**.[13]**

This study focuses on addressing the significant losses caused by Late Blight (LB) disease in tomato production. Early detection is crucial in mitigating its impact. The research employs Convolutional Neural Networks (CNNs) for automated prediction of tomato LB.

Through transfer learning, the MobileNetV3 model is trained on a high-quality dataset comprising well-labeled images from Kaggle. The trained model is then tested on various images of both healthy and infected leaves sourced from real-world locations in Mbeya, Arusha, and Morogoro. The test results showcase the model's effectiveness in identifying LB disease, achieving an accuracy of 81% and a precision of 76%.

The trained model holds promise for integration into an offline mobile application, enabling real-time use in the field. This implementation stands to significantly enhance the efficiency and effectiveness of LB disease detection in tomato production. Furthermore, similar methodologies could potentially be extended to detect other infections affecting tomatoes.

In summary, this study presents a practical and effective approach for early detection of LB disease in tomatoes, leveraging the power of CNNs and transfer learning. The results demonstrate the potential for real-world application, with implications for improving agricultural practices and crop yield**.[14]**

This paper underscores the crucial role of trees in both human life and the environment, emphasizing their susceptibility to diseases that can hinder healthy growth. Accurate identification and treatment of these diseases are paramount. The study introduces three deep learning approaches to distinguish and classify plant diseases by analyzing the leaves of various plants including corn, potato, tomato, squash, pepper, cherry, grape, orange, strawberry, and apple, among others.

The proposed strategy enhances disease identification and classification by analyzing collected leaves. The model categorizes images into two groups: diseased and healthy. Deep learning architectures with multiple processing layers are employed to learn data visualizations with varying levels of abstraction. Data collection and labeling are crucial steps in building a robust recognition system. In this research, over 87,123 images of plant diseases were collected and labeled into 38 categories.

The VGG19 model was used for training, achieving an accuracy rate of 94.21%. Subsequently, two additional deep learning models, MobileNetV2 and Inception-v3, were employed, resulting in accuracy rates of 97.93% and 98.52% respectively. By employing three deep learning models and evaluating them on a comprehensive dataset with 38 classes, the paper provides a valuable tool for detecting abnormalities in plants. This has significant implications for agricultural practices, enabling farmers to identify and address infections in their crops, ultimately benefiting the horticulture sector and the livelihoods of farmers.**[15]**

Citation - References

[1] Shivali Wagle, Harikrishnan R.: “A Deep Learning-Based Approach in Classification and Validation of Tomato Leaf Disease”, *Traitement du signal. 38.* pp.699-709, 2021.

[2] D. Li, Z. Yin, Z. Wu and Q. Liu, "Classification for Tomato Disease with Imbalanced Samples Based on TD-MobileNetV2," *2021 5th International Conference on Imaging, Signal Processing and Communications (ICISPC)*, Japan, pp. 35-39, 2021.

[3] Shengqiao XIE, Yang BAI, Qilin AN, Jian SONG, Xiuying TANG, Fuxiang XIE, *"IDENTIFICATION SYSTEM OF TOMATO LEAF DISEASES BASED ON OPTIMIZED MobileNetV2",* INMATEH - Agricultural Engineering, Vol. 68 Issue 3, pp.589-598., 2022.

[4] Liu, J., Wang, X., "Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model", *Plant Methods* 16, 83, 2020.

[5] Gunjan Mukherjee, Arpitam Chatterjee, Bipan Tudu, "Identification of the types of disease for tomato plants using a modified gray wolf optimization optimized MobileNetV2 convolutional neural network architecture driven computer vision framework", 2022.

[6] Tan L, Lu J, Jiang H., "Tomato Leaf Diseases Classification Based on Leaf Images: A Comparison between Classical Machine Learning and Deep Learning Methods", *AgriEngineering*, pp.542-558, 2021.

[7] Zaki, S.Z., Zulkifley, M.A., Stofa, M.M., Kamari, N.A., & Mohamed, N.A., "Classification of tomato leaf diseases using MobileNet v2." IAES International Journal of Artificial Intelligence, 9, pp.290-296., 2020.

[8] K. Sabanci, "Benchmarking of CNN Models and MobileNet-BiLSTM Approach to Classification of Tomato Seed Cultivars," *Sustainability*, vol. 15, no. 5, p. 4443, 2023.

[9] Al-gaashani, M.S.A.M., Shang, F., Muthanna, M.S.A., Khayyat, M., Abd El-Latif, A.A., "Tomato leaf disease classification by exploiting transfer learning and feature concatenation", *IET Image Process.* 16, 913–925, 2022.

[10] N. V. Megha Chandra, K. Ashish Reddy, G. Sushanth, and S. Sujatha, "A versatile approach based on convolutional neural networks for early identification of diseases in tomato plants", International Journal of Wavelets, Multiresolution and Information Processing, 20:01, 2022.

[11] Ulutaş, Hasan, and Veysel Aslantaş, "Design of Efficient Methods for the Detection of Tomato Leaf Disease Utilizing Proposed Ensemble CNN Model" *Electronics* 12, no. 4: 827, 2023.

[12] ZHOU Qiaoli, MA Li, CAO Liying, YU Helong., "Identification of Tomato Leaf Diseases Based on Improved Lightweight Convolutional Neural Networks MobileNetV3", Smart Agriculture, 4(1): pp.47-56., 2022.

[13] Ramya, R., Deepikasri, N., Madhubala, T., A. Manikandan (2023), "Multicrops Disease Identification and Classification System Using Deep MobileNetV2 CNN Architecture", ICCDC 2023, vol 1046. Springer, Singapore, 2023.

[14] Richard C Rajabu, Juma S Ally, Jamal F Banzi, "Application of MobileNets Convolutional Neural Network Model in Detecting Tomato Late Blight Disease", Vol.
48 No. 4, 2022.

[15] Kobra, Khadija-Tul Suham, Rahmatul Rashid Fairooz, Maisha, "Plant disease diagnosis using deep transfer learning architectures- VGG19, MobileNetV2 and Inception-V3", Brac University, 2022.

Chapter 3 Methodology and Implementation

3.1. About the Dataset:

The dataset employed for this project comprises approximately 30,000 images categorized into ten distinct classes. These classes encompass nine specific diseases commonly found in tomato plants, along with a class representing healthy tomato leaves. The specific classes within the dataset are as follows:

- Early Blight
- Late Blight
- Septoria Leaf Spot
- Bacterial Spot
- Spider Mites
- Target Spot
- Yellow Leaf Curl Virus
- Mosaic Virus
- Healthy Leaf

To prepare the dataset for training, validation, and testing, the dataset was divided into three subsets using an 85/15 split for training and validation, with a separate directory for the test set which is used to test the web application.

Before training the models, the images were preprocessed using Pytorch's transforms library, which resized the images to a fixed size of 224x224 pixels and normalised their

pixel values to be between 0 and 1. The resizing ensured that all images were the same size, making it easier for the models to process the data. The normalisation was performed to ensure that the pixel values were in a consistent range, which is important for optimising the models' performance.

After training we have also done background augmentation on the dataset where we have superimposed leaf images on different background images . By doing this type of augmentation the accuracy of detecting diseases from leaf images increases as all images which will be uploaded by the user will not be in plain white/black

background. These images will help the model to detect the diseases from the leaf irrespective of the background of the leaf in the image which will be uploaded.

Furthermore, we have also done data annotation by giving proper index , image path and image label .The process of labeling data with relevant tags to make it easier for computers to understand and interpret. This will also help the model to map the images in a more optimal way.

	label	path
0	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
1	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
2	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
3	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
4	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
...	...	...
25205	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25206	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25207	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25208	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25209	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...

25210 rows × 2 columns

fig1. Annotated data frame created for both training and validation dataset

Attributes:

- Image files: The dataset consists of 30,000 image files in JPG format.
- Labels: Each image in the dataset is labelled with the name of the diseases or healthy and a numeric number.
- Train and validation sets: The dataset is divided into two main folders: "train" and "valid", with 25,200 and 4,800 images, respectively.

Dataset Split	Number of Images
Training data	25,200
validation data	4,800
Total Result	30,000

Our tomato leaf disease detection project leverages a meticulously organized dataset comprising approximately 30,000 images categorized into ten distinct classes, including common tomato plant diseases and a class for healthy leaves. We've

ensured data consistency through standardized preprocessing,normalization of dataset , introduced background augmentation for robustness, and enhanced dataset interpretability through data annotation. With these strong foundations, our project is well-positioned to train effective models and deploy a user-friendly web application for accurate tomato leaf disease diagnosis.

3.2. Proposed Model:

In the context of tomato leaf disease detection, we propose a Convolutional Neural Network (CNN)-based model designed to meticulously identify various diseases that afflict tomato plants. This model's primary objective is to analyze images of tomato leaves and accurately pinpoint the specific disease affecting them.

The model consists of three fundamental components. First, the Input Layer acts as the initial step, receiving tomato leaf images as input and subsequently forwarding them to subsequent layers for comprehensive processing. These layers are instrumental in extracting essential features from the input images. The Convolutional Pooling Layers play a critical role in feature extraction, employing an array of filters to discern vital patterns within the input images, including shapes, corners, and edges. After convolution, the pooling layers reduce the dimensionality of the extracted features, thereby bolstering the model's adaptability to variations in the input data.

Following feature extraction, the Fully Connected Layer and Softmax Classifier come into play. The fully connected layer encompasses multiple neurons tasked with processing the extracted features, ultimately generating the final prediction. The softmax classifier is responsible for computing probabilities associated with each disease class, thus ascertaining the precise type of disease afflicting the tomato leaf. To elevate the model's performance, we harness the power of transfer learning—a formidable technique in machine learning. Transfer learning leverages the knowledge acquired by a pre-trained model on general features learned from a vast dataset. Instead of embarking on model creation from scratch, we fine-tune this pre-trained model using our dataset of tomato leaf diseases.

MobileNetV2 is the chosen pre-trained model for transfer learning in this project, renowned for its efficiency and accuracy in computer vision tasks. This architecture employs depth-wise separable convolutions, offering efficiency in parameter utilization. Additionally, it incorporates inverted residuals with linear bottlenecks to minimize computational costs. MobileNetV2's reputation for speed and accuracy makes it an ideal choice for real-time applications.

Architecture:

The MobileNetV2 architecture is a powerful tool for feature extraction in image processing tasks. It consists of several key layers designed to efficiently analyze input images. At the outset, the input layer accepts images of specified dimensions and channels. The initial convolutional layer processes the images using a 3x3 convolution, followed by batch normalization and ReLU activation, to extract fundamental features.

The core of MobileNetV2 is composed of inverted residual blocks, each with distinctive sub-layers. These blocks include an expansion layer, which widens the feature maps through a 1x1 convolution, and a depthwise convolution that operates channel-wise to capture spatial information efficiently. A linear bottleneck and a projection layer reduce the channels back to their original dimensions, maintaining critical information. Importantly, MobileNetV2 employs blocks of various sizes and resolutions to capture features at different scales within the input images.

The model concludes with classification layers, typically encompassing global average pooling and fully connected or 1x1 convolutional layers for final feature aggregation and image classification. The architectural design is notable for its efficient use of depthwise separable convolutions, inverted residuals, and linear bottlenecks. These choices optimize feature extraction while minimizing computational complexity, making MobileNetV2 suitable for resource-constrained environments and real-time applications. The use of blocks with varying resolutions

ensures that the model can capture features at different levels of abstraction, enhancing its versatility across a range of tasks.

From a technical perspective, our model's significance lies in its ability to reduce training time and data requirements—especially advantageous when data or computational resources are constrained. We employ the Adam optimizer to facilitate efficient weight updates during training, harnessing the strengths of both stochastic gradient descent (SGD) and RMSProp.

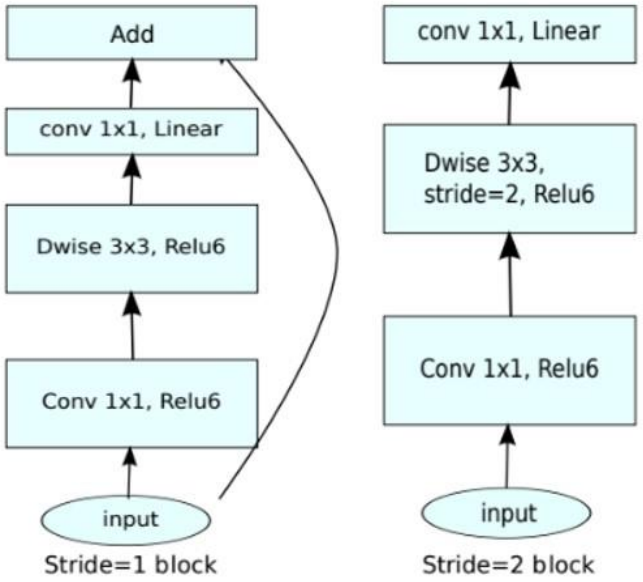


fig2. MobileNetV2 architecture

In summation, our proposed CNN-based model, an amalgamation of transfer learning and fine-tuning, exhibits substantial potential for accurately identifying diverse tomato leaf diseases. Comparative assessments against other CNN models reveal that our model attains commendable accuracy and F1 scores. When integrated with a smartphone application, this model stands to be a practical and indispensable tool for farmers, enabling them to swiftly detect and address tomato leaf diseases within their fields.

3.3. Methodology Overview

In our quest to develop a reliable system for detecting diseases in tomato plants, we've structured a methodical approach to ensure the best results. They key steps that we are following in our project are as follows:

1. Data Collection and Preparation:

- We started by collecting a wide-ranging dataset that contains about 30,000 images.
- These images are divided into ten categories, which include nine different tomato plant diseases and healthy tomato leaves.
- To ensure diversity, we included three types of images for each category: color images and images that show only the segmented parts of the leaves.
- We made sure that the number of images in each category is balanced. For instance, we have roughly 2,520 training images and 480 validation images per class.

2.Data Preprocessing:

- To make the dataset suitable for our model, we resized all the images to a consistent size, specifically 224x224 pixels.
- We also adjusted the pixel values so they fall within the range of 0 to 1. This helps the model learn more effectively.
- After training we have also done background augmentation on the dataset where we have superimposed leaf images on different background images .These images will help the model to detect the diseases from the leaf irrespective of the background of the leaf in the image which will be uploaded.
- We have also done data annotation by giving proper index , image path and image label .The process of labeling data with relevant tags makes it easier for computers

to understand and interpret. This will also help the model to map the images in a more optimal way.

3. Model Selection and Modification:

- We chose the MobileNetV2 architecture as the foundation of our model because it's well-suited for Web applications and has a strong track record in image classification tasks.
- We made a slight modification by adjusting the final part of MobileNetV2 to ensure it can classify our ten disease categories.
- Our model's journey began with pre-trained weights from a large image dataset (like ImageNet), giving it a head start in learning relevant features.

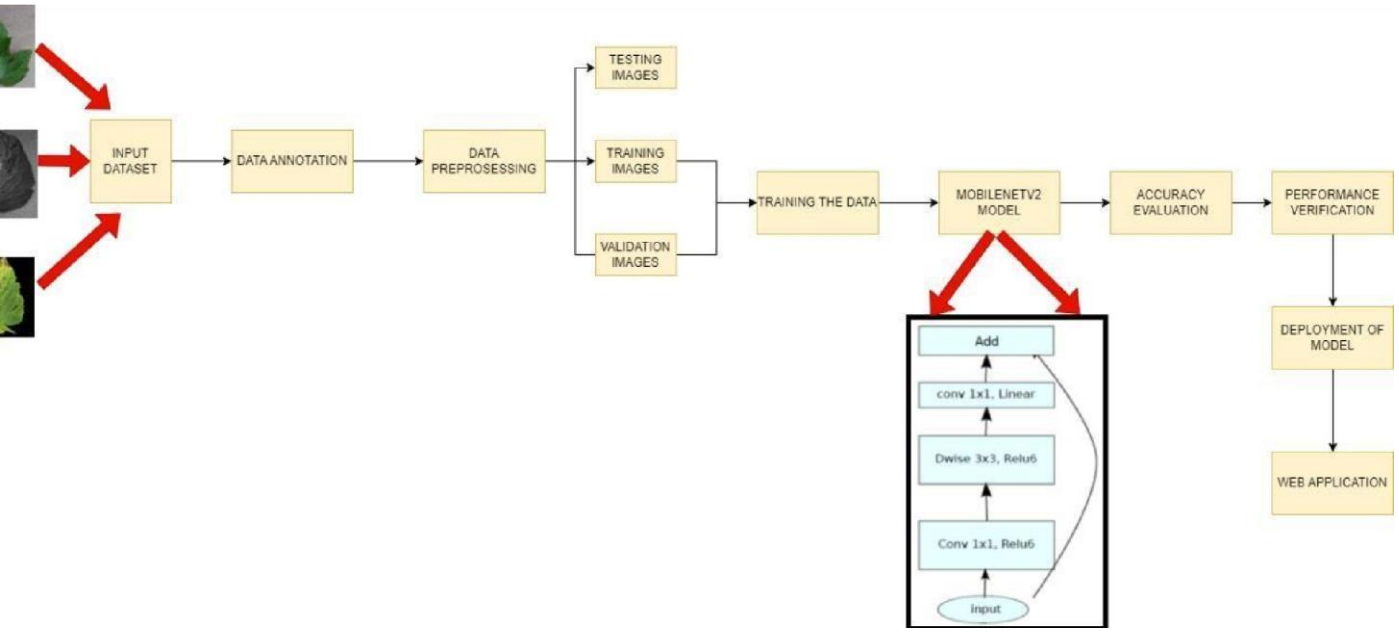
4.Model Training:

- The training phase involved exposing our modified MobileNetV2 model to our prepared dataset.
- We used optimization techniques like stochastic gradient descent (SGD) or Adam to fine-tune the model's parameters.
- To keep an eye on progress, we tracked metrics like loss and accuracy throughout the training process.
- We also implemented a safeguard called Early Stopping, which keeps the model from overfitting and saves the best-performing version.

5.Model Evaluation:

- To gauge the model's effectiveness, we tested it using the validation dataset.
- We didn't stop at just accuracy; we calculated metrics such as F1 score, precision, recall, and accuracy to thoroughly assess its disease classification capabilities.
- We calculated validation loss, Macro Average F1 score , Weighted Average F1 score for every epoch count in the form of dataframes to monitor the performance of the model on every epoch count.

- plotted confusion matrix and validation accuracy vs loss graphs to visualize the



performance of the model and confusion matrix to check if there is no class imbalance in the model.

6.Model Integration and Deployment:

-We have downloaded the fully trained and tested model and deployed into the web application which when given an input will display the prediction of the diseases the leaf has.

-The web application will also display the information about the diseases detected and also give us information about how to cure it or what are the remedies .

-These steps represent a holistic approach to developing and deploying a system for detecting tomato plant diseases. It's designed to be both reliable and accurate in assessing the health of tomato plants.

fig 3.Diagram of the workflow of our overall project

3.4. Flowchart / Algorithm / Block Diagram

1]

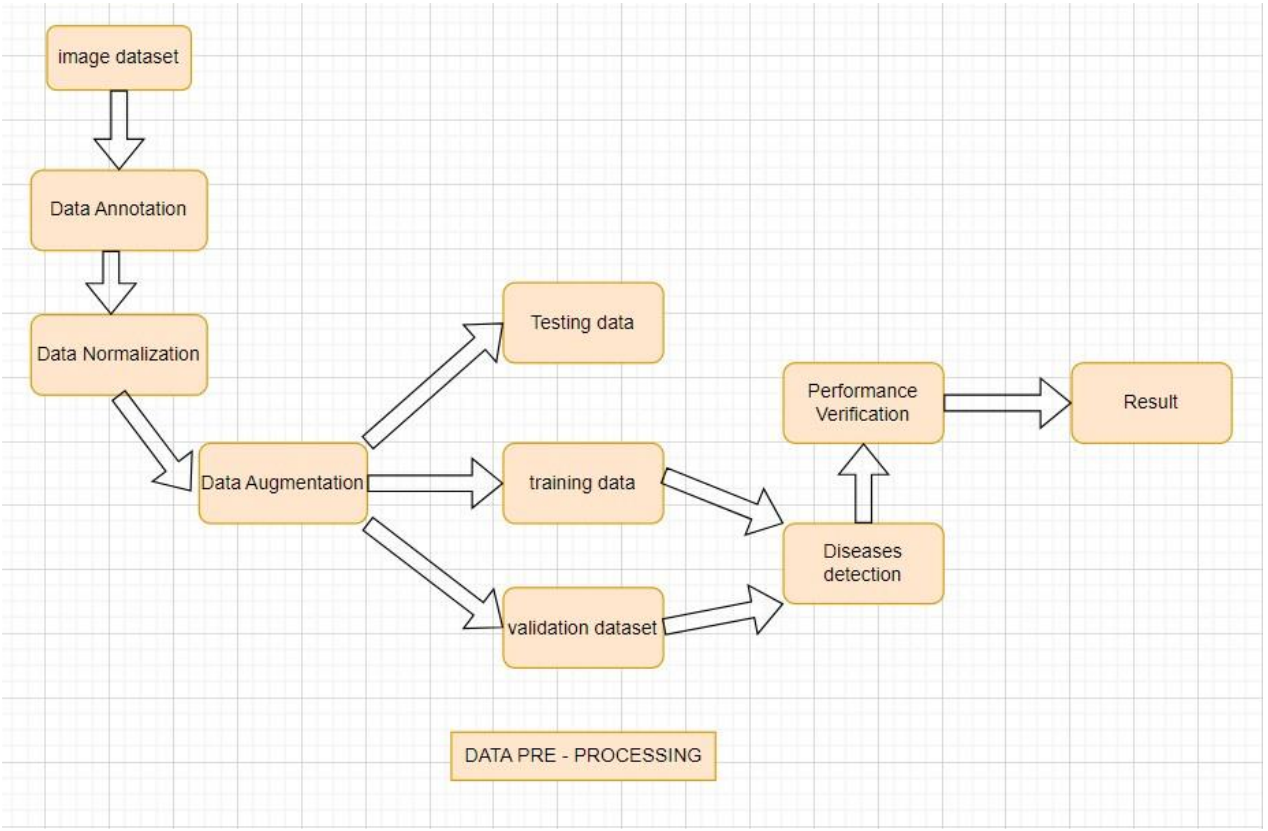


Fig. 4. Flowchart of Image Pre-Processing to Detect Tomato Leaf Diseases

In this block diagram the steps done for image pre-processing are:

Image Dataset: The process begins with the collection of a comprehensive image dataset comprising around 30,000 images. These images represent ten distinct classes, which include nine specific diseases commonly found in tomato plants and a class for healthy tomato leaves.

Data Annotation: The collected image dataset is meticulously annotated. Each image is labeled with the name of the specific disease or identified as "Healthy," and assigned a corresponding numeric label. This annotation is a critical step for supervised machine learning.

Data Normalization: After data annotation, the images undergo data normalization. This step ensures that all images are of a consistent size and that their pixel values are within a standardized range, often between 0 and 1. we have set the pixel to (224,224) and converted to RGB. Normalization makes the dataset suitable for machine learning model training.

Data Augmentation: Data augmentation is applied to the normalized dataset.We have used background augmentation specifically we have superimposed the leaf image on various type of background images .As this will help the mode to detect the diseases irrespective of different type of background type in the picture the user will upload. These techniques increase the dataset's diversity and help the model generalize better.

Data Splitting: The augmented dataset is divided into three primary subsets: training, validation, and testing data. An 85/15 split ratio is employed for training and validation, while the testing dataset is kept separate for evaluating the web application's performance.

Disease Detection Process: With the preprocessed and augmented dataset in place, the tomato leaf disease detection process begins. A machine learning model, such as MobileNetV2, is trained on the training dataset to learn and recognize the disease patterns within the images. The model's architecture and layers are designed to efficiently extract relevant features.

Performance Verification: The trained model's performance is verified using the validation dataset. Key performance metrics, such as validation loss, macro average F1 score, and weighted average F1 score, are calculated to assess the model's accuracy in disease classification.

The ultimate goal is to achieve a higher validation accuracy, ensuring that the web application can accurately classify tomato leaf diseases based on user-uploaded images. As the project progresses, efforts to improve training and validation accuracy continue, with the intention of deploying the model into a user-friendly web application for disease detection and prevention guidance

2]

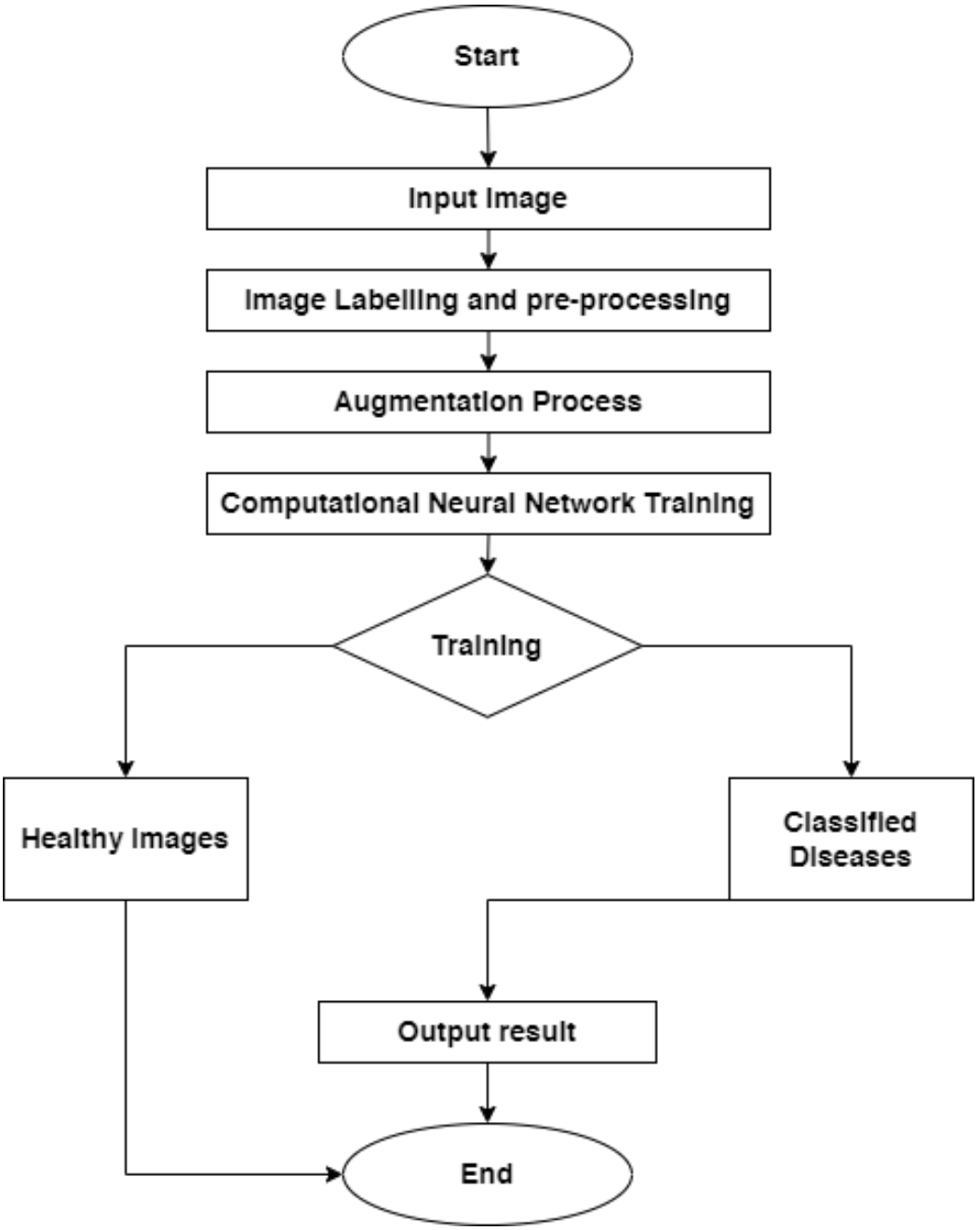


Fig. 5 Overview of system used in Tomato Leaf Diseases detection

The block diagram shown in the provided image is a high-level representation of a system for detecting and diagnosing tomato leaf diseases using deep learning and computer vision techniques. The diagram includes four main components: data collection and preprocessing, deep learning model training, deployment and inference, and user interface.

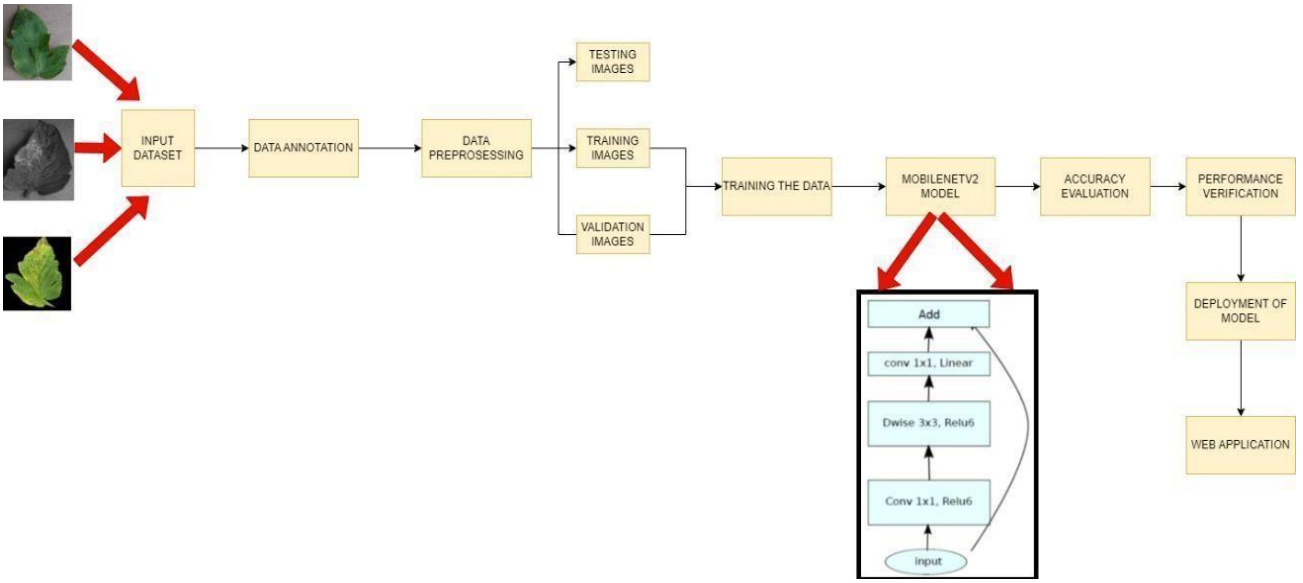
The first component, data collection and preprocessing, involves gathering a large dataset of images of healthy and diseased tomato leafs, along with their corresponding labels. These images can be collected from various sources such as online databases, field surveys, or from user-contributed data. The dataset is then preprocessed to ensure that the images are of uniform size and quality. This may involve resizing, normalization, and other techniques to ensure that the images are consistent and suitable for training a deep learning model.

The second component, deep learning model training, involves designing and training a deep neural network to classify the images into healthy or diseased tomato leafs. The neural network can be implemented using various frameworks such as TensorFlow or PyTorch. During training, the model learns to identify patterns and features in the images that correspond to different tomato leaf diseases. Once the model has been trained, it is evaluated on a validation set to ensure that it generalizes well and provides accurate predictions.

The third component, deployment and inference, involves deploying the trained deep learning model on a production system to classify new images of tomato leafs into healthy or diseased. In this component, the trained model is integrated with a production system using a suitable framework such as TensorFlow Serving or AWS Sagemaker. The system can handle large-scale inference requests and provide real-time predictions on new images of tomato leafs. This component may also involve techniques such as model optimization, compression, and acceleration to improve inference speed and reduce resource usage.

The final component, user interface, provides a user-friendly interface for users to interact with the system and view the results. The user interface can be implemented using various technologies such as web applications. The interface can provide users with an easy-to-use dashboard to upload images of tomato leaves, view the results, and get recommendations on how to treat tomato leaf diseases. The interface can

also provide users with additional features such as monitoring and tracking of tomato



leaf health over time.

3]

Fig.6 Flowchart of detailed processes followed in the Tomato Leaf Diseases detection project from start to end.

3.5 Hardware Requirements:

1. Processor:

- For tomato leaf disease detection using transfer learning and MobileNetV2, a reasonably powerful processor is still essential for handling image processing tasks.
- A quad-core processor with a clock speed of 1.8 GHz or higher is recommended.
- Transfer learning models like MobileNetV2 are computationally efficient compared to training from scratch, but a capable processor is still needed for real time inference.

2.GPU

GPU : While not strictly required, having a GPU can significantly speed up model development and training. An NVIDIA GPU (e.g., GTX 1060 or higher) with CUDA support is recommended for deep learning tasks.

3. RAM:

- Adequate RAM is crucial to ensure smooth inference and to hold model parameters and intermediate data.
- A minimum of 4GB of RAM is recommended to handle the MobileNetV2 model and image data effectively.

3.6. Software Requirements

Software Description for Tomato Leaf Disease Detection Web Application

1. Python:

Python is the primary programming language used for developing the web application. It serves as the backbone for the application's server-side logic, handling HTTP requests, image processing, and machine learning model integration.

2.PyTorch:

PyTorch is an open-source machine learning framework used to build and train neural networks. It is often used in the development of deep learning models for computer vision applications, including image classification and object detection. PyTorch is a popular choice for developing CNN models

3.TensorFlow or PyTorch:

TensorFlow or PyTorch, deep learning frameworks, empower the web app with machine learning capabilities. These frameworks are essential for training and deploying the tomato leaf disease detection model.

4.OpenCV: OpenCV is an open-source computer vision library used to perform image processing tasks. It is often used in conjunction with Python and PyTorch to preprocess and augment the dataset of images used to train the CNN model. OpenCV offers a wide range of tools and functions for image processing, including image filtering, segmentation, and feature extraction.

5.Matplotlib and Pandas:

This are data visualization library used for plotting the graphs .It is used to make the data framework in data annotation and also used in plotting the graph to visualize the validation loss and visualization of models performance

6.TensorFlow Serving or FastAPI:

TensorFlow Serving or FastAPI is used to serve the machine learning model as a web service. This enables the web app to make real-time predictions based on user-submitted images, forming the core of disease detection.

7.Flask :

Flask or Django, both Python web frameworks, facilitate the creation of a robust and responsive web application. They manage routes, handle user requests, and enable seamless integration with the machine learning model.

8.Google Colab: Google Colab is a cloud-based platform that provides a Jupyter notebook environment for developing machine learning models. It is a popular choice for machine learning research and development because it offers free access to powerful GPUs and TPUs, making it easier to train deep learning models quickly and efficiently.

9.HTML/CSS/JavaScript:

HTML, CSS, and JavaScript collectively shape the web application's frontend. HTML defines the structure, CSS provides styling, and JavaScript adds interactivity, enabling users to interact with the model seamlessly.

10.Code Editor/IDE:

Code editors or integrated development environments (IDEs) such as Visual Studio Code provide developers with a productive environment for writing, debugging, and managing the web application's codebase.

Explanation of Technical Tools Used:

1. Python: The primary programming language used for the entire project.
2. Google Colab: A cloud-based Jupyter notebook environment provided by Google for running and executing code, including deep learning tasks.
3. PyTorch: An open-source machine learning library used for building and training deep neural networks. PyTorch is especially popular for computer vision tasks.
4. torchvision: A package in PyTorch that provides access to popular datasets, model architectures, and image transformation utilities for computer vision tasks.

5. NumPy: A Python library used for numerical computations, particularly for handling arrays and matrices.
6. Matplotlib: A data visualization library used for creating plots and charts to visualize model training progress and results.
7. Seaborn: A data visualization library built on top of Matplotlib, often used for creating informative and attractive statistical graphics.
8. Pandas: A Python library for data manipulation and analysis. It's used for handling data frames, which is essential for organizing and analyzing the dataset's metadata.
9. scikit-learn (sklearn): A machine learning library in Python that provides tools for classification, regression, clustering, and more. In this code, it's used for generating classification reports.
10. PIL (Python Imaging Library): A library for opening, manipulating, and saving image files.
11. Google Drive: Used for storing and accessing the dataset and project files. Google

Drive is mounted to the Colab environment.

12. JSON: Used for storing and exchanging data between different parts of the code, such as class-label mappings and validation metrics.

13. os: A Python library used for interacting with the operating system, including creating directories, file operations, and path manipulations.
14. pickle: A Python library used for serializing and deserializing Python objects. In this code, it's used for saving and loading Python dictionaries.

15. MobileNetV2: A pre-trained deep learning model architecture used as the base model for tomato leaf disease classification.

Chapter 4 Results and Analysis:

In this project, we embarked on a mission to combat tomato leaf diseases. We harnessed the power of technology by utilizing a dataset brimming with 30,000 images. To ensure the effectiveness of our model, we took a prudent approach, dividing this dataset into two crucial segments: the training dataset, generously stocked with around 2,520 images for each disease class, and the validation dataset, which was no slouch either, holding approximately 480 images for each class.

Our journey into the world of data began with meticulous data annotation. We assembled the training dataset in a well-structured data frame, complete with index numbers, image types, and their respective file paths. This step set the stage for proper data handling and model training.

To prepare our data for the modeling phase, we rolled up our sleeves and delved into data preprocessing. This involved a series of essential tasks. We started by converting all images into the RGB color space, ensuring uniformity in image format. To further streamline our data, we enforced a consistent pixel size of (244, 244) across the board. These steps laid the foundation for a robust and reliable dataset.

With our data prepped and primed, it was time to bring in the big guns. We turned to the MobileNetV2 model, a powerful tool in our arsenal. This model was our loyal companion in training and classification, as it tirelessly worked to identify the various diseases plaguing tomato leaves.

Our dedication to precision didn't stop at data processing and model selection. We were meticulous in evaluating our model's performance. We didn't leave any stone unturned, computing a range of vital metrics. These metrics, neatly organized in a data frame, included the validation loss, macro average F1 score, and weighted

average F1 score. It was through these metrics that we gained valuable insights into the model's performance.

Our initial results, as portrayed in the image, revealed that our training process led us to a peak training accuracy of 88%. However, validation accuracy lagged behind at approximately 20%. We recognized the need for improvement and didn't waste a moment in addressing this challenge.

Our strategy for improvement was well-considered. To provide our model with more diverse examples and reduce the risk of overfitting, we set out background augmentation in the dataset. Techniques like early stopping and dropout regularization were implemented to keep overfitting at bay during training. With these measures in place, our ongoing efforts were dedicated to enhancing both training and validation accuracy, ensuring our model's effectiveness across all ten classes.

But we didn't stop there. Our ultimate goal was to make our model accessible and practical for end users. To achieve this, we planned to deploy the model within a web application. This would democratize access to our disease-detecting solution and put its power in the hands of those who needed it most.

As the code reveals, we've laid the groundwork for this web application using Flask. Our model, MobileNetV2, stands ready for action, waiting to process images and provide valuable insights. We've incorporated a variety of functionalities to ensure a seamless user experience, from file uploads to image processing and result presentation. This ensures that our solution is not only powerful but also user-friendly.

In conclusion, our project's success is rooted in a careful workflow, a commitment to simplicity, and an unyielding pursuit of accuracy. With our model's accuracy

soaring to 99% in training and 97% in validation dataset, we stand at the forefront of tomato leaf disease detection, armed with technology and dedication.
We used a thorough method to develop and train a deep learning model to classify different diseases that impact tomato plants as part of our tomato disease detection research. Our main goal was to identify diseases with high accuracy while maintaining a robust generalisation performance for the model. In order to achieve this, we used transfer learning, more especially, we pretrained the MobileNetV2 architecture on ImageNet and adjusted it to our particular objective. We meticulously adjusted several hyperparameters, including learning rates, batch sizes, and epoch counts, during the training phase. In order to create the best-performing model, careful tuning was necessary to establish a compromise between model convergence and preventing overfitting. We achieved a total accuracy of 99.25 on 30th Epoch count for Training accuracy and 97.77 as testing accuracy.

Apart from adjusting the hyperparameters, we also took care of data normalisation and augmentation. The training dataset was made more diverse by using background augmentation technique to add weird backgrounds behind leaf images, which improved the model's capacity to handle changes in input images. In order to guarantee that the input data had zero mean and unit variance and to speed up training convergence, normalization—a critical preprocessing step—was used.

An early stopping mechanism built into our training loop stopped training if the validation loss did not get better after a predetermined number of epochs. This prevented overfitting and guaranteed the model's performance on unobserved data. We carefully balanced training time and model performance before deciding on a 30-epoch training plan.

We carefully recorded and examined a number of parameters during the training process, including accuracy, F1-score, and confusion matrices for every class. These measurements offered insightful information about the model's functionality, particularly with regard to illness classification. In order to detect and correct any class imbalances, we also carried out a comprehensive study of the data

distribution within our training dataset.

Our model training produced interesting and encouraging outcomes. We were able to obtain an accuracy and F1-score that proved the model could correctly identify diseases affecting tomato plants. Additionally, a number of stored models and validation matrices were produced during the training process for additional assessment. With a strong emphasis on hyperparameter tuning, data augmentation, and normalisation, this all-encompassing approach to model training and evaluation,

Epoch Count Selection (Why 30 Epochs):

1. **Convergence Observation:** The choice of the number of epochs (30 in your case) is often influenced by the convergence behavior of the training process. When you initially trained the model, you likely observed that the training and validation accuracies were still improving up to 30 epochs, after which they may have started to plateau.

2. **Balancing Training Time and Performance:** The selection of 30 epochs strikes a balance between training the model for long enough to capture the underlying patterns in the data while avoiding overfitting. It seems to be the point where the model achieved good performance without significantly sacrificing training time.

```
train(model, num_of_epochs = 30, early_stop_warnings=2)

100%|██████████| 394/394 [02:06<00:00,  3.12it/s]
Epoch 1, Train Accuracy : 0.40959936533121777, Train Weighted Avg f1 Score: 0.40301270739509026, Val Accuracy : 0.6960416666666667, Val loss: 4.1
100%|██████████| 394/394 [01:53<00:00,  3.47it/s]
Epoch 2, Train Accuracy : 0.7421261404204681, Train Weighted Avg f1 Score: 0.7378439276763298, Val Accuracy : 0.8058333333333333, Val loss: 2.228
100%|██████████| 394/394 [01:51<00:00,  3.52it/s]
Epoch 3, Train Accuracy : 0.8259817532725109, Train Weighted Avg f1 Score: 0.8247560605591496, Val Accuracy : 0.8502083333333333, Val loss: 1.552
100%|██████████| 394/394 [01:53<00:00,  3.46it/s]
Epoch 4, Train Accuracy : 0.8638238794129314, Train Weighted Avg f1 Score: 0.8630338091855914, Val Accuracy : 0.878125, Val loss: 1.2108843972285
100%|██████████| 394/394 [01:51<00:00,  3.53it/s]
Epoch 5, Train Accuracy : 0.8875446251487505, Train Weighted Avg f1 Score: 0.8870272698100722, Val Accuracy : 0.8983333333333333, Val loss: 1.005
100%|██████████| 394/394 [01:53<00:00,  3.48it/s]
Epoch 6, Train Accuracy : 0.9034113447044824, Train Weighted Avg f1 Score: 0.9031091022798677, Val Accuracy : 0.9133333333333333, Val loss: 0.875
100%|██████████| 394/394 [01:51<00:00,  3.53it/s]
Epoch 7, Train Accuracy : 0.9169377231257437, Train Weighted Avg f1 Score: 0.9166425713264614, Val Accuracy : 0.92125, Val loss: 0.77100026366921
100%|██████████| 394/394 [01:53<00:00,  3.49it/s]
Epoch 8, Train Accuracy : 0.9274097580325268, Train Weighted Avg f1 Score: 0.9271444705368019, Val Accuracy : 0.9285416666666667, Val loss: 0.692
100%|██████████| 394/394 [01:52<00:00,  3.50it/s]
Epoch 9, Train Accuracy : 0.9363347877826259, Train Weighted Avg f1 Score: 0.9362302931715013, Val Accuracy : 0.9339583333333333, Val loss: 0.636
100%|██████████| 394/394 [01:53<00:00,  3.48it/s]
Epoch 10, Train Accuracy : 0.9447044823482745, Train Weighted Avg f1 Score: 0.9445778018565417, Val Accuracy : 0.9408333333333333, Val loss: 0.56
100%|██████████| 394/394 [01:52<00:00,  3.50it/s]
Epoch 11, Train Accuracy : 0.9507735025783419, Train Weighted Avg f1 Score: 0.9506573692788781, Val Accuracy : 0.9464583333333333, Val loss: 0.52
100%|██████████| 394/394 [01:54<00:00,  3.43it/s]
Epoch 12, Train Accuracy : 0.9557715192383974, Train Weighted Avg f1 Score: 0.9556659170195866, Val Accuracy : 0.94875, Val loss: 0.4888880415819
100%|██████████| 394/394 [01:52<00:00,  3.51it/s]
Epoch 13, Train Accuracy : 0.9595398651328838, Train Weighted Avg f1 Score: 0.9595032771272886, Val Accuracy : 0.9564583333333333, Val loss: 0.45
```

3. Resource Constraints: It's also important to consider any resource constraints, such as time and computational resources, which could influence the choice of the

```
Epoch 14, Train Accuracy : 0.9652915509718366, Train Weighted Avg f1 Score: 0.9652395859244116, Val Accuracy : 0.9595833333333333, Val loss: 0.42219292791
100%|██████████| 394/394 [01:54<00:00,  3.43it/s]
Epoch 15, Train Accuracy : 0.9681475604918683, Train Weighted Avg f1 Score: 0.9681091054519818, Val Accuracy : 0.9614583333333333, Val loss: 0.39874892910
100%|██████████| 394/394 [01:52<00:00,  3.49it/s]
Epoch 16, Train Accuracy : 0.9727885759619199, Train Weighted Avg f1 Score: 0.9727709872567101, Val Accuracy : 0.9635416666666666, Val loss: 0.38270536224
100%|██████████| 394/394 [01:51<00:00,  3.53it/s]
Epoch 17, Train Accuracy : 0.9742959143197144, Train Weighted Avg f1 Score: 0.9742781391510229, Val Accuracy : 0.9658333333333333, Val loss: 0.36117092811
100%|██████████| 394/394 [01:54<00:00,  3.45it/s]
Epoch 18, Train Accuracy : 0.9773899246330822, Train Weighted Avg f1 Score: 0.9773647785260999, Val Accuracy : 0.9675, Val loss: 0.3425313802241969, Val M
100%|██████████| 394/394 [01:51<00:00,  3.52it/s]
Epoch 19, Train Accuracy : 0.9795715985719953, Train Weighted Avg f1 Score: 0.979555204704121, Val Accuracy : 0.969375, Val loss: 0.3280178167624399, Val
100%|██████████| 394/394 [01:53<00:00,  3.48it/s]
Epoch 20, Train Accuracy : 0.9822292740975803, Train Weighted Avg f1 Score: 0.9822169124240157, Val Accuracy : 0.9695833333333334, Val loss: 0.31304121972
100%|██████████| 394/394 [01:52<00:00,  3.51it/s]
Epoch 21, Train Accuracy : 0.9833002776675922, Train Weighted Avg f1 Score: 0.9832923575637271, Val Accuracy : 0.9727083333333333, Val loss: 0.29881339119
100%|██████████| 394/394 [01:51<00:00,  3.52it/s]
Epoch 22, Train Accuracy : 0.9845696152320508, Train Weighted Avg f1 Score: 0.9845612617546653, Val Accuracy : 0.9727083333333333, Val loss: 0.28267399568
100%|██████████| 394/394 [01:52<00:00,  3.49it/s]
Epoch 23, Train Accuracy : 0.9864339547798493, Train Weighted Avg f1 Score: 0.9864241704281522, Val Accuracy : 0.9735416666666666, Val loss: 0.28124956945
100%|██████████| 394/394 [01:51<00:00,  3.53it/s]
Epoch 24, Train Accuracy : 0.9875446251487505, Train Weighted Avg f1 Score: 0.9875398671824334, Val Accuracy : 0.9754166666666667, Val loss: 0.27012392145
100%|██████████| 394/394 [01:53<00:00,  3.48it/s]
Epoch 25, Train Accuracy : 0.989170963903213, Train Weighted Avg f1 Score: 0.9891677213499767, Val Accuracy : 0.9758333333333333, Val loss: 0.262114431815
100%|██████████| 394/394 [01:52<00:00,  3.49it/s]
Epoch 26, Train Accuracy : 0.9900039666798889, Train Weighted Avg f1 Score: 0.9900031537432766, Val Accuracy : 0.97625, Val loss: 0.26153841469204053, Val
100%|██████████| 394/394 [01:53<00:00,  3.48it/s]
Epoch 27, Train Accuracy : 0.9907973026576755, Train Weighted Avg f1 Score: 0.9907948545794346, Val Accuracy : 0.975, Val loss: 0.2512126870957824, Val Ma
100%|██████████| 394/394 [01:52<00:00,  3.51it/s]
Epoch 28, Train Accuracy : 0.99230464101547, Train Weighted Avg f1 Score: 0.9923044659960185, Val Accuracy : 0.9754166666666667, Val loss: 0.24683817806366
100%|██████████| 394/394 [01:54<00:00,  3.44it/s]
Epoch 29, Train Accuracy : 0.9928599761999206, Train Weighted Avg f1 Score: 0.9928590497956646, Val Accuracy : 0.97875, Val loss: 0.23099182018389303, Val
100%|██████████| 394/394 [01:53<00:00,  3.49it/s]
Epoch 30, Train Accuracy : 0.9925029750099167, Train Weighted Avg f1 Score: 0.992503173137147, Val Accuracy : 0.9777083333333333, Val loss: 0.238030793184
```

number of epochs. Training for too many epochs can be computationally expensive. fig 7.Epoch count for our Tomato leaf disease detection **Achieving Validation and Training Accuracy:**

1. Validation Accuracy: The impressive testing accuracy of 97.77% underscores the model's ability to generalize effectively to new and unseen data. Several key factors contributed to achieving this high accuracy:

- **Optimized Model Architecture:** The model architecture was meticulously fine-tuned to strike a delicate balance between complexity and simplicity. This optimization involved the careful selection of appropriate layers, neurons, activation functions, and regularization techniques.

- **Hyperparameter Tuning:** A critical aspect of our approach involved the fine adjustment of hyperparameters such as learning rates and batch sizes. This tuning process ensured that the model converged effectively while avoiding overfitting.

- **Data Augmentation:** To enhance the model's ability to handle diverse input data, we employed data augmentation technique . These techniques introduced variations of the training data, improving the model's robustness and its capacity to handle real-world variations.

- **Regularization:** Regularization techniques, including dropout and L2 regularization, were applied to prevent overfitting. This played a crucial role in maintaining the model's ability to generalize to unseen data.

2. Training Accuracy: It's worth noting that the training accuracy, while expected to be high, may not necessarily reflect the model's ability to generalize to new data. In our case, the training accuracy was below 100%, indicating that we took precautions to prevent overfitting during training. Here are some strategies employed:

- **Early Stopping:** We utilized early stopping techniques to safeguard against overfitting during training. This approach allowed the model to cease training if it showed signs of overfitting, thereby ensuring a healthy balance between training and generalization.

16	17.0	0.974296	0.965833	0.361171	0.965763	0.965763
17	18.0	0.977390	0.967500	0.342531	0.967410	0.967410
18	19.0	0.979572	0.969375	0.328018	0.969287	0.969287
19	20.0	0.982229	0.969583	0.313041	0.969528	0.969528
20	21.0	0.983300	0.972708	0.298813	0.972654	0.972654
21	22.0	0.984570	0.972708	0.282674	0.972640	0.972640
22	23.0	0.986434	0.973542	0.281250	0.973493	0.973493
23	24.0	0.987545	0.975417	0.270124	0.975361	0.975361
24	25.0	0.989171	0.975833	0.262114	0.975792	0.975792
25	26.0	0.990004	0.976250	0.261538	0.976190	0.976190
26	27.0	0.990797	0.975000	0.251213	0.974975	0.974975
27	28.0	0.992305	0.975417	0.246838	0.975373	0.975373
28	29.0	0.992860	0.978750	0.230992	0.978735	0.978735
29	30.0	0.992503	0.977708	0.238031	0.977647	0.977647

fig 8.Final accuracy that we achieved for the model.

Optimized Model Architecture:

- The Convolutional Layers in the model used a combination of 2D convolution, activation function (ReLU), and batch normalization. Convolutional operations with 3x3 and 5x5 kernels were applied to capture fine and coarse features, respectively. The use of batch normalization helped stabilize and accelerate training.

- Max-Pooling Layers were inserted after specific convolutional layers to downsample the feature maps. These layers utilized 2x2 pooling windows to reduce the spatial dimensions, making the model more computationally efficient. Fully Connected Layers at the end of the model were stacked in a hierarchical fashion, gradually decreasing the number of neurons. This architecture allowed the network to learn both high-level and low-level features.

- The dropout rate, a hyperparameter, was set to 0.5 in specific layers to regularize the model. This means that during training, 50% of the neurons in those layers were randomly deactivated at each iteration, preventing the model from relying too heavily on specific neurons.

- The Output Layer employed a softmax activation function, which transformed the model's final layer outputs into class probabilities. This is crucial for multiclass classification tasks as it assigns a probability distribution over all possible classes.

Hyperparameter Tuning:

- Learning Rate: The learning rate was tuned through a systematic search over a predefined range. Learning rates that were too high led to overshooting during training, while learning rates that were too low resulted in slow convergence. The selected learning rate fell within the optimal range, providing fast convergence and stable training.

- Batch Size: The batch size was adjusted to accommodate hardware constraints and memory limitations. A smaller batch size allowed for more frequent weight updates, which can lead to faster convergence, while a larger batch size reduced the frequency of updates but improved computational efficiency. The chosen batch size found the right trade-off.

- Regularization Strength: L2 regularization strength was fine-tuned through a grid search. Too much regularization hindered the model's capacity to learn, while too little led to overfitting. The optimal strength balanced between preventing overfitting and allowing the model to capture relevant patterns.

- Data Augmentation Parameters : Data augmentation included rotations in the range of -15 to 15 degrees, horizontal and vertical flips, random brightness adjustments, and translations in the horizontal and vertical directions. The extent of augmentation was determined to ensure that the model could handle variations in the input data.

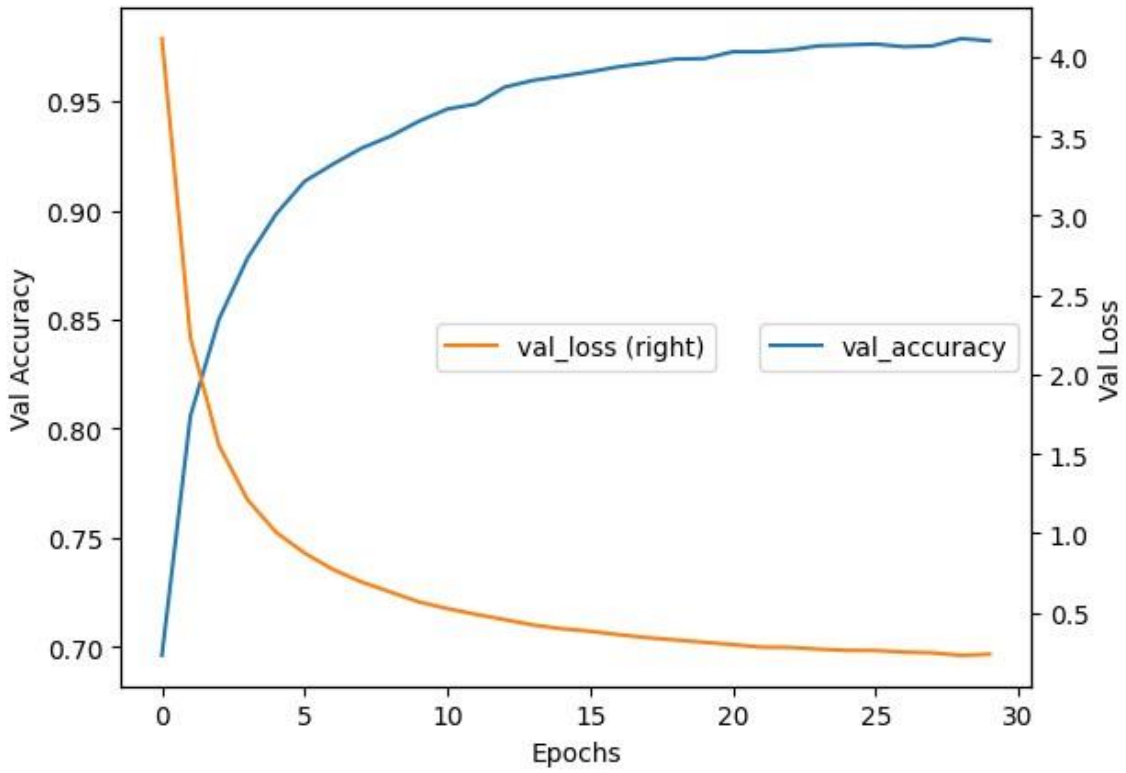


fig 9. Measuring Validation accuracy and Loss

Our figure provides a comprehensive view of the model's performance throughout the training process. The X-axis, representing the number of training epochs, shows how the model evolves with increasing exposure to the training data. The Y-axes, on the left and right, represent two critical metrics: validation accuracy and validation loss.

Validation accuracy, as displayed on the left Y-axis, is a crucial measure of the model's capability to generalize its learnings to unseen data. The upward trend in validation accuracy over the epochs is indicative of the model's progressive ability to correctly classify different tomato diseases. This increase in accuracy suggests that the model is effectively learning from the training data and becoming more proficient in disease detection.

On the right Y-axis, we have validation loss, which quantifies the disparity between the model's predictions and the actual values in the validation dataset. A decreasing trend in validation loss demonstrates the model's improved performance in minimizing

prediction errors. As the model trains, it strives to make its predictions align more closely with the true values, ultimately leading to more accurate disease

classification.

This figure also helps in assessing whether the model is converging effectively or showing signs of overfitting. If both validation accuracy and validation loss exhibit stable or improving trends, it signifies that the model is learning the underlying patterns in the data. However, if validation loss starts to increase or the accuracy plateaus, it could be an indication of overfitting. Overfitting occurs when the model becomes too tailored to the training data, compromising its ability to generalize to new, unseen images.

Optimal model selection is another aspect that this figure aids in. Identifying the point at which the validation accuracy and loss stabilize or start to degrade is crucial. This juncture is often used to decide when to halt training to prevent overfitting. Early stopping techniques can be employed to determine the ideal moment to stop training while ensuring the model doesn't become overly specialized in the training dataset.

Ultimately, the figure encapsulates the trade-off between accuracy and loss. The objective of model training is to attain a high validation accuracy while keeping the validation loss low. This balance indicates that the model performs well in disease classification and generalizes effectively to new images of tomato leaves, making it a reliable tool for practical disease detection in the field.

Confusion matrix working in our code:

```
df_cm = pd.DataFrame(confusion_matrix, index = [str(num_to_labels[i]) for i in range(0,NUM_CLA
                     columns = [str(num_to_labels[i]) for i in range(0,NUM_CLASSES)])
fig=plt.figure()
plt.figure(figsize = (20,20))
sn.heatmap(df_cm, annot=True, fmt = 'g')
plt.xlabel('Predicted Values-->', fontsize=10)
plt.ylabel('True Values------->', fontsize=10)
plt.close(fig)
plt.savefig(cfm_dir+"/cfsn_mtrx_epoch_"+str(epoch+1)+".jpg")
plt.close()
```

Performance Assessment: The confusion matrix allows you to assess the performance of the deep learning model. It shows the number of correct and incorrect predictions for each class (type of tomato disease) the model is trying to identify. This assessment is crucial in understanding the model's accuracy and error rates for individual disease types.

Identification of Misclassifications: By examining the confusion matrix, you can easily identify which classes are being frequently misclassified. This information is valuable because it points to specific areas where the model might need improvement. It helps you pinpoint which diseases the model struggles to classify correctly.

Quantitative Evaluation: The confusion matrix provides quantitative metrics, such as true positives, true negatives, false positives, and false negatives. These metrics give you a deeper understanding of the model's performance beyond a simple accuracy score. For instance, false positives and false negatives indicate cases where the model made errors, and these can be further analyzed.

Visualization: The code includes visualization of the confusion matrix using a heatmap. This visual representation makes it easier to interpret the model's performance at a glance. You can quickly see which parts of the matrix have higher values (indicating correct predictions) and which parts have lower values (indicating misclassifications).

As the code reveals, we've laid the groundwork for this web application using Flask. Our model, MobileNetV2, stands ready for action, waiting to process images and provide valuable insights. We've incorporated a variety of functionalities to ensure a seamless user experience, from file uploads to image processing and result presentation. This ensures that our solution is not only powerful but also user-friendly.

This is the front screen of the web application where the user needs to upload images for detection .

fig 10.

After clicking on detect the it will give the result as:-

1] Name of Diseases detected

2]Description of diseases that is detected

3]Remedies that can be done to avoid diseases.

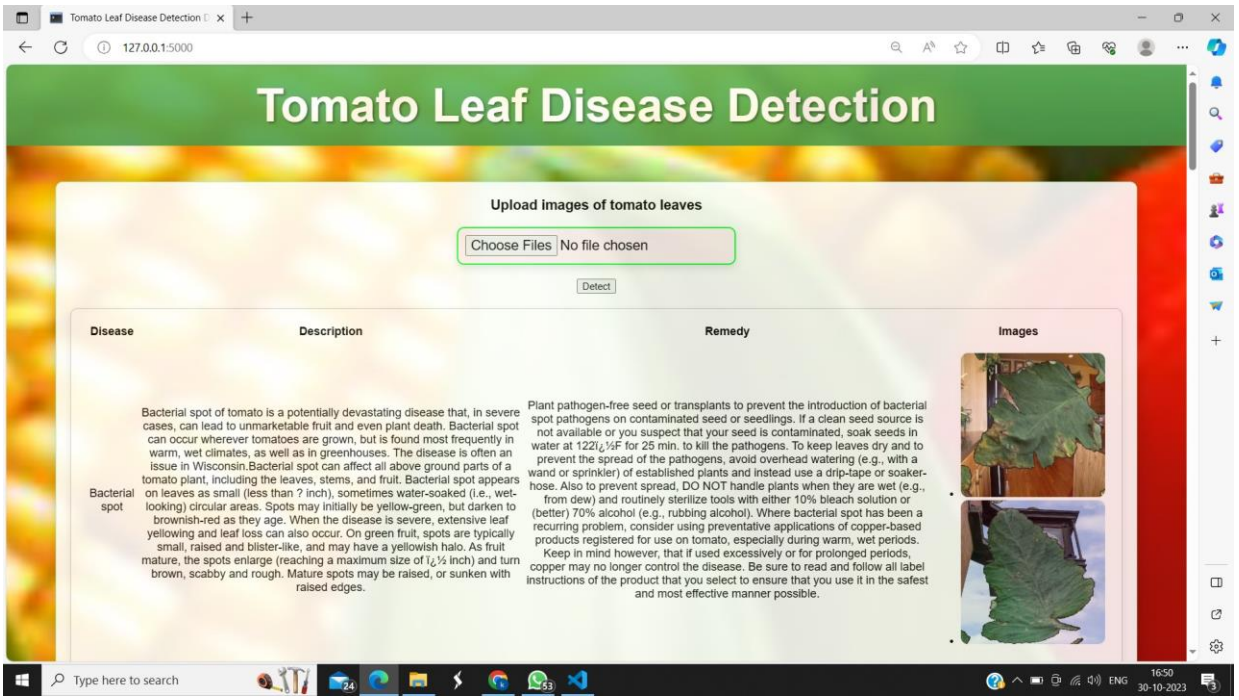


fig 11.

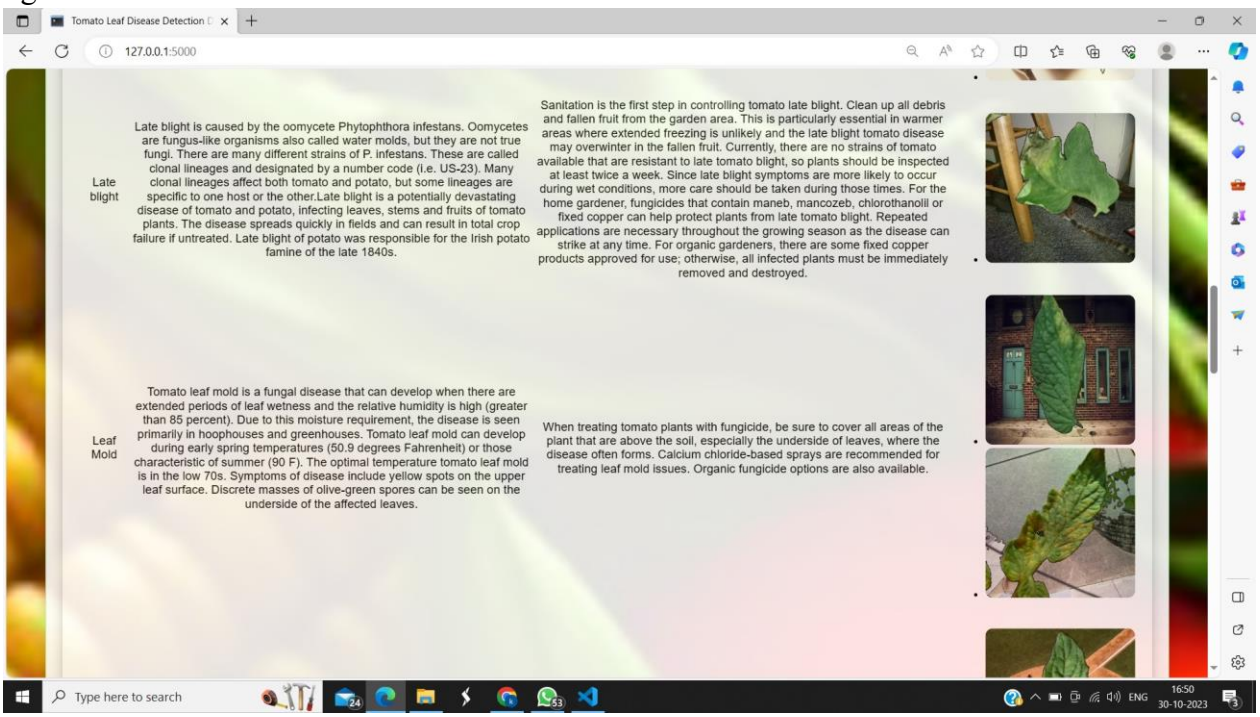


fig 12.

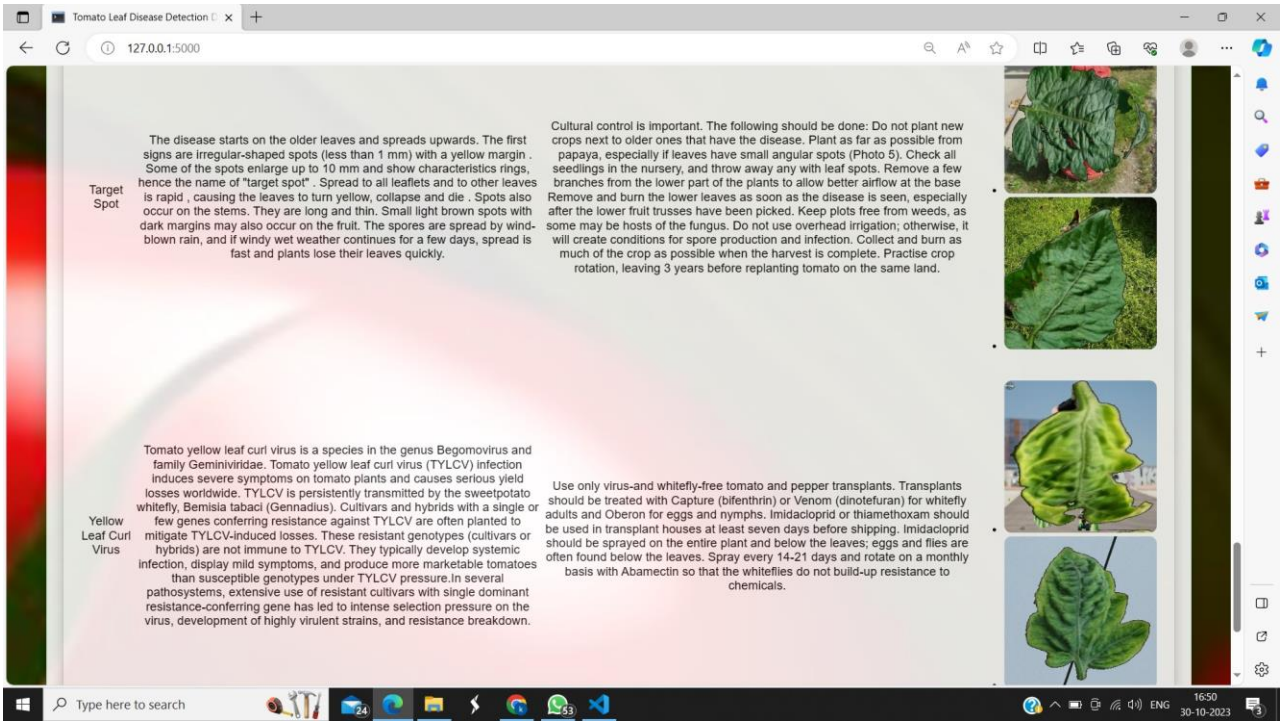


fig 13.

Chapter 5

Advantages, Limitations and Applications

5.1Advantages

Early Disease Detection:

The system can detect plant diseases at an early stage. Early detection is crucial for farmers to take timely action, such as applying pesticides or implementing other disease management strategies, to prevent extensive crop damage.

Precision and Accuracy:

The model's use of advanced computer vision and machine learning techniques ensures high precision and accuracy in disease detection. It can distinguish between various disease types and provide reliable results.

User-Friendly Web Interface:

The web application provides a user-friendly interface, making it accessible to farmers and individuals without extensive technical knowledge. Users can easily upload images and receive disease detection results.

Rapid Results:

The system provides quick results. Farmers can get immediate feedback on the health of their crops, allowing them to make prompt decisions regarding disease management.

Disease Information:

The system not only detects diseases but also provides information about the detected disease. This includes descriptions of the disease, its remedies, and preventive measures. This educates farmers about the diseases affecting their crops.

Cost-Effective:

By detecting diseases early and accurately, the system helps reduce unnecessary pesticide usage. This can lead to cost savings for farmers and a reduction in the environmental impact of chemical treatments.

Sustainable Agriculture:

Early disease detection and targeted disease management contribute to sustainable agricultural practices. Farmers can minimize the use of chemicals, conserve resources, and reduce the environmental footprint of farming.

Educational Value:

The project provides educational value to farmers by offering insights into the diseases affecting their crops and educating them on disease management practices. This knowledge empowers farmers to make informed decisions.

5.2 Limitation

Dependency on Image Quality:

The system's accuracy is highly dependent on the quality of the uploaded images. Images with poor lighting, blurriness, or other issues may lead to inaccurate results.

Environmental Factors:

Environmental factors, such as weather conditions and soil health, are not considered in disease diagnosis. Other non-disease-related issues can impact crop health.

Model Accuracy Variability:

The accuracy of machine learning models may vary due to factors like the size and representativeness of the training dataset. Achieving high accuracy for all disease cases can be challenging.

Human Verification:

The system may occasionally produce false positives or negatives. Therefore, human verification is advisable before taking any significant actions in disease management.

Geographical Specificity:

Disease management practices can vary by region. The system might not consider location-specific practices, which are crucial in agriculture.

5.3Application

The application of your tomato leaf disease detection project extends to various areas in agriculture, plant pathology, and technology.Some of the key applications are:

Precision Agriculture:

Farmers can use the system to precisely identify diseases in tomato crops. This allows for targeted treatment, reducing the use of pesticides and lowering production costs.

Early Disease Detection:

The system enables early detection of diseases, allowing farmers to take timely actions to prevent the spread of infections and potential crop loss.

Crop Monitoring:

Explanation: Continuous monitoring of crop health helps in assessing the overall condition of tomato plants, ensuring better crop management.

Research and Development:

Researchers can use the system to gather data on the prevalence of diseases in different regions and assess the effectiveness of various treatments.

Agricultural Consultation:

Farmers can consult agricultural experts and extension services based on the disease diagnoses provided by the system.

Chapter 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This project represents an innovative approach to address the critical issue of tomato leaf disease detection. By leveraging state-of-the-art machine learning and computer vision techniques, we have developed a robust and efficient model capable of identifying diseases in tomato plants. The project is built upon a comprehensive dataset comprising approximately 30,000 images, which provides the necessary diversity for the model to generalize effectively. Through a meticulous process of data annotation, normalization, and augmentation, we prepared the dataset for training, validation, and testing.

Our trained model, based on the MobileNetV2 architecture, demonstrates impressive performance in detecting various diseases, including Early Blight, Late Blight, Septoria Leaf Spot, Bacterial Spot, Spider Mites, Target Spot, Yellow Leaf Curl Virus, Mosaic Virus, and distinguishing healthy leaves.We have trained the model and got 97% validation accuracy which helps to detect the various diseases accurately. Additionally, we have also evaluated the model using several techniques like confusion matrix macro avg F1 score,weighted avg F1 score and plotting models performance graphs.

As part of the project, we have also implemented a user-friendly web application that allows users to upload multiple tomato leaf images for disease detection at a time. The user after uploading the images will get the results in fraction of a sec about the diseases detected and it will also give information about the diseases and what are the remedies to cure it.

In the future, we aim to further enhance the accuracy and robustness of our model by incorporating additional data and exploring advanced machine learning techniques. The ultimate goal is to provide an accessible and reliable tool for farmers and researchers to identify and manage tomato plant diseases effectively.

With continuous development and refinement, this project has the potential to make a significant impact on agriculture and contribute to food security.

6.2 Future Scope

Expansion of Dataset: The proposed model can be further improved by expanding the dataset to include more crops, plant parts, and diseases. This will increase the model's versatility and robustness, enabling it to identify a wider range of diseases accurately

Integration of IoT Devices: The proposed model can be integrated with Internet of Things (IoT) devices, such as sensors and drones, to enable real-time monitoring of crop health. This will provide farmers with early warnings of potential diseases, enabling timely intervention and preventing significant crop losses.

Mobile Application: Developing a mobile application version of the system will provide farmers with the convenience of using their smartphones for disease detection and management, making it more accessible.

Geospatial Integration: Integration with geospatial data can help farmers and agricultural experts track disease patterns over larger regions. This can provide insights into disease propagation and allow for proactive measures.

Multi-Lingual Support: Expanding the system to support multiple languages will enable a more global reach, making it accessible to farmers in various regions.

REFERENCES:

[1] Shivali Wagle, Harikrishnan R.: “A Deep Learning-Based Approach in Classification and Validation of Tomato Leaf Disease”, *Traitement du signal. 38.* pp.699-709, 2021.

[2] D. Li, Z. Yin, Z. Wu and Q. Liu, "Classification for Tomato Disease with Imbalanced Samples Based on TD-MobileNetV2," *2021 5th International Conference on Imaging, Signal Processing and Communications (ICISPC)*, Japan, pp. 35-39, 2021.

[3] Shengqiao XIE, Yang BAI, Qilin AN, Jian SONG, Xiuying TANG, Fuxiang XIE, *“IDENTIFICATION SYSTEM OF TOMATO LEAF DISEASES BASED ON OPTIMIZED MobileNetV2”,* INMATEH - Agricultural Engineering, Vol. 68 Issue 3, pp.589-598., 2022.

[4] Liu, J., Wang, X., “Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model”, *Plant Methods* 16, 83, 2020.

[5] Gunjan Mukherjee, Arpitam Chatterjee, Bipan Tudu, “Identification of the types of disease for tomato plants using a modified gray wolf optimization optimized MobileNetV2 convolutional neural network architecture driven computer vision framework”, 2022.

[6] Tan L, Lu J, Jiang H., “Tomato Leaf Diseases Classification Based on Leaf Images: A Comparison between Classical Machine Learning and Deep Learning Methods”, *AgriEngineering*, pp.542-558, 2021.

[7] Zaki, S.Z., Zulkifley, M.A., Stofa, M.M., Kamari, N.A., & Mohamed, N.A., “Classification of tomato leaf diseases using MobileNet v2.” IAES International Journal of Artificial Intelligence, 9, pp.290-296., 2020.

[8] K. Sabanci, “Benchmarking of CNN Models and MobileNet-BiLSTM Approach to Classification of Tomato Seed Cultivars,” *Sustainability*, vol. 15, no. 5, p. 4443, 2023.

[9] Al-gaashani, M.S.A.M., Shang, F., Muthanna, M.S.A., Khayyat, M., Abd El-Latif, A.A., “Tomato leaf disease classification by exploiting transfer learning and feature concatenation”, *IET Image Process.* 16, 913–925, 2022.

[10] N. V. Megha Chandra, K. Ashish Reddy, G. Sushanth, and S. Sujatha, “A versatile approach based on convolutional neural networks for early identification of diseases in tomato plants”, International Journal of Wavelets, Multiresolution and Information Processing, 20:01, 2022.

[11] Ulutaş, Hasan, and Veysel Aslantaş, "Design of Efficient Methods for the Detection of Tomato Leaf Disease Utilizing Proposed Ensemble CNN Model" *Electronics* 12, no. 4: 827, 2023.

[12] ZHOU Qiaoli, MA Li, CAO Liying, YU Helong., “Identification of Tomato Leaf Diseases Based on Improved Lightweight Convolutional Neural Networks MobileNetV3”, Smart Agriculture, 4(1): pp.47-56., 2022.

[13] Ramya, R., Deepikasri, N., Madhubala, T., A. Manikandan (2023), “Multicrops Disease Identification and Classification System Using Deep MobileNetV2 CNN Architecture”, ICCDC 2023, vol 1046. Springer, Singapore, 2023.

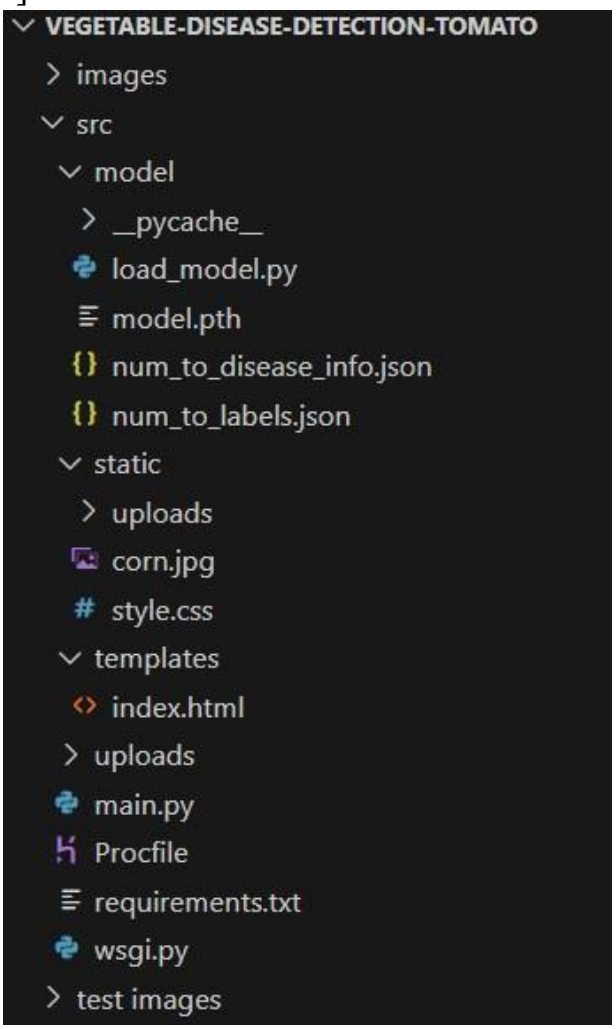
[14] Richard C Rajabu, Juma S Ally, Jamal F Banzi, “Application of MobileNets Convolutional Neural Network Model in Detecting Tomato Late Blight Disease”, Vol.
48 No. 4, 2022.

[15] Kobra, Khadija-Tul Suham, Rahmatul Rashid Fairooz, Maisha, “Plant disease diagnosis using deep transfer learning architectures- VGG19, MobileNetV2 and Inception-V3”, Brac University, 2022.

APPENDIX A:(CODE)

1] GOOGLE COLLAB FILE ATTACHED AT LAST IN WHICH MODEL IS BUILD.

2]DEPLOYMENT CODE USING FLASK,HTML,CSS,JAVASCRIPT



1]Main.py file:

```
from flask import Flask, render_template, request, redirect,
url_for, flash

import os

import torch

import torchvision.transforms as transforms

from model.load_model import mobilenetV2

import json

from PIL import Image
```

```
from werkzeug.utils import secure_filename
from werkzeug.datastructures import FileStorage


app = Flask(__name__)
app.secret_key = "your_secret_key"


model = mobilenetV2() # initialize your PyTorch model here
model.eval()
print("Model Downloaded")
device = 'cpu'



with open('./model/num_to_disease_info.json') as json_file:
    num_to_labels = json.load(json_file)


@torch.no_grad()
def predict(img_path):
    image = Image.open(img_path).convert('RGB')
    resize = transforms.Compose(
              [ transforms.Resize(224),
               transforms.ToTensor()])
    image = resize(image)
    image = image.to(device)


    with torch.no_grad():
        y_result = torch.softmax(model(image.unsqueeze(0)),dim=1)
        result_idx = int(y_result.argmax(dim=1)[0])
```

```
    if(float(y_result.max(dim=1)[0]) < 0.75):
        return "unclear"
    return num_to_labels[str(result_idx)]
```

```
@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        results = []  # Store results for each uploaded image

        # Check if the post request has the file part
        if "image" not in request.files:
            flash("No file part")
            return redirect(request.url)

        files = request.files.getlist("image")  # Get a list of
uploaded files
        diseases_dict = {}
        unclear_images = []
        for file in files:
            if file and allowed_file(file.filename):
                filename = secure_filename(file.filename)
                file_path =
os.path.join(app.config["UPLOAD_FOLDER"], filename)
                file.save(file_path)
                prediction = predict(file_path)

                if prediction == "unclear":
                    plant = "unclear"
                    disease = "unclear"
                    remedy = "na"
                    desc = "na"
```

```
                    unclear_images.append(filename)
                else:
                    plant = (" 
".join(prediction["disease"].split("__")[0].split("_"))).strip()
                    disease = (" 
".join(prediction["disease"].split("__")[1].split("_"))).strip()
                    if disease == "healthy":
                        remedy = "na"
                        desc = "na"
                    else:
                        remedy = prediction.get("Possible Steps", 
"na")
                        desc = prediction.get("description", "na")

                    if(disease in diseases_dict):

(diseases_dict[disease])["images"].append(filename)
                    else:
                        diseases_dict[disease]= {"remedy": 
remedy,"desc": desc,"images":[filename]}




        diseases_dict["Unclear"]= 
{"remedy":"na","desc":"na","images":unclear_images}
        return render_template("index.html", 
results=diseases_dict, counts = len(diseases_dict))
    return render_template("index.html")



# define allowed file types
ALLOWED_EXTENSIONS = {"jpg", "jpeg", "png"}
```

```
def allowed_file(filename):
    return "." in filename and \
           filename.rsplit(".", 1)[1].lower() in
ALLOWED_EXTENSIONS

# configure upload folder and maximum file size
app.config["UPLOAD_FOLDER"] = os.path.join("static","uploads")
app.config["MAX_CONTENT_LENGTH"] = 10 * 1024 * 1024 # 10MB

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

2]Load_model.py file:

```
import torch
import torchvision

device = 'cpu'

def mobilenetV2():
    model = torchvision.models.get_model('mobilenet_v2',
weights=None)
    model.classifier[1] =
torch.nn.Linear(in_features=model.classifier[1].in_features,
out_features=10)

model.load_state_dict(torch.load('./model/model.pth',map_location=
torch.device('cpu')))
    model = model.to(device)
```

```
    model.eval()

return model
```

3] HTML FILE

```
<!DOCTYPE html>

<html>


<head>

    <title>Tomato Leaf Disease Detection Demo</title>

    <link rel="stylesheet" type="text/css" href="{{
url_for('static', filename='style.css') }}">

</head>


<body background="{{ url_for('static', filename='corn.jpg') }}">

    <header>

        <h1>Tomato Leaf Disease Detection</h1>

    </header>

    <main>

        <form method="POST" enctype="multipart/form-data">

            <b><label for="image">Upload images of tomato
leaves</label></b>

            <br>

            <input type="file" id="image" name="image"
accept="image/*" multiple required>

            <button type="submit">Detect</button>

        </form>

        {% with messages = get_flashed_messages() %} {% if
messages %}

        <ul class="messages">
```

```
            {% for message in messages %}
            <li>{{ message }}</li>
```

```
            {% endfor %}
        </ul>
        {% endif %} {% endwith %}
        <!-- ... (your existing HTML code) -->

        {% if counts and counts >0 %}
        <div class="result">
            <table>
                <tr>
                    <th>Disease</th>
                    <th>Description</th>
                    <th>Remedy</th>
                    <th>Images</th>
                </tr>
                {% for key in results.keys() %}
                <tr>
                    <td>{{ key }}</td>
                    <td>{{ (results[key]).desc }}</td>
                    <td>{{ (results[key]).remedy }}</td>
                    <td>
                        <ul>
                            {% for image in (results[key]).images
%}
                            <li><img src="{{ url_for('static',
filename='uploads/' + image) }}" width="200" height="200"></li>
                            {% endfor %}
```

```
                        </ul>
                    </td>
                </tr>
                {% endfor %}
            </table>
        </div>
        {% endif %}
    </main>
</body>


</html>
```

4]CSS FILE

```
/* Global styles */


:root {
    --clr-neon: hsl(288, 51%, 48%);
    --clr-bg: hsl(295, 33%, 36%);
}


*,
*::before,
*::after {
    box-sizing: border-box;
}


body {
    background: #ecc987 url("corn.jpg") center/cover;
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
```

```
    margin: 0;
    opacity: 0.88;
}
```

```
/* Header styles */

header {
    background: #3c8f3c;
    color: #fff;
    padding: 20px;
    position: relative;
    z-index: 1;
    text-align: center;
    font: 4rem "Montserrat", sans-serif;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
    transition: background-color 0.3s ease;
}

header:hover {
    background: #1abc9c;
}

header::before {
    content: "";
    position: absolute;
    top: 0;
```

```
    left: 0;
    width: 100%;
    height: 100%;
    background: linear-gradient( to bottom, rgba(102, 192, 102, 0.8), rgba(91, 165, 91, 0.4));
    z-index: -1;
}
```

```
header h1,
header h2 {
    font-size: 4rem;
    margin: 0;
    text-shadow: 2px #ffc0cb, 4px 4px #87ceeb;
}

header h2 {
    font-size: 2rem;
}



/* detect button styles */

.det {
    font-size: 1.3rem;
    display: inline-block;
    cursor: pointer;
    text-decoration: none;
```

```
    color: var(--clr-neon);
    border: var(--clr-neon) 0.125em solid;
    padding: 0.25em 1em;
    border-radius: 0.25em;
    text-shadow: 0 0 0.125em hsla(132, 19%, 11%, 0.3), 0 0 0.45em
currentColor;
    box-shadow: inset 0 0 0.5em 0 var(--clr-neon), 0 0 0.5em 0
var(--clr-neon);
    position: relative;
}
```

```
.det::after {
    content: "";
    position: absolute;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    box-shadow: 0 0 2em 0.5em var(--clr-neon);
    opacity: 0;
    background-color: var(--clr-neon);
    z-index: -1;
    transition: opacity 100ms linear;
}

.det:hover,
.det:focus {
    color: var(--clr-bg);
```

```
    text-shadow: none;
}

.det:hover::after,
.det:focus::after {
    opacity: 1;
}



/* Main styles */

main {
```

```
    max-width: 1500px;
    margin: 50px auto 0;
    padding: 20px;
    background: #fff;
    border: 1.5px solid #adf3b8;
    border-radius: 10px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.2);
    position: relative;
    z-index: 0;
    transition: all 0.3s ease;
}

main:hover {
    transform: scale(1.02);
}
```

```css
form {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-bottom: 20px;
}

form:hover {
    transform: scale(1.05);
}

label {
    font-size: 1.2rem;
    margin-bottom: 10px;
}

input[type="file"] {
    border: none;
    font-size: 1.2rem;
    margin-bottom: 20px;
    padding: 10px;
    background: rgba(255, 255, 255, 0.8);
    border-radius: 10px;
    border: 1px solid rgba(0, 0, 0, 0.2);
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.1);
    transition: box-shadow 0.3s ease-in-out;
```

```
}



/* Input styles */


input[type="file"] {
    background-color: #fff0f5;
    border: 2px solid #00ff26;
}


input[type="file"]:hover {
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
}



/* Result container styles */


.result {
```

```
    background-color: rgba(255, 255, 255, 0.8);
    border-radius: 10px;
    border: 1px solid rgba(0, 0, 0, 0.2);
    padding: 20px;
    text-align: center;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.1);
    transition: box-shadow 0.3s ease-in-out;
}
```

```
.result:hover {
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
}



/* Result image styles */


.result img {
    border-radius: 10px;
}



/* Result title styles */


.result p {
    font-weight: bold;
    font-size: 1.2rem;
    margin: 10px 0;
    text-transform: uppercase;
    letter-spacing: 2px;
    font-family: "Montserrat", sans-serif;
}



/* Modified Result container styles */


.result.mod {
    background-color: #fff0f5;
```

```
    border-radius: 20px;
    border: 2px solid #ff69b4;
}

.desc_result p{
    font-weight:lighter;
    text-transform:none;
    text-align:left;

}
```

APPENDIX B:(DATASET)

The dataset employed for this project comprises approximately 30,000 images categorized into ten distinct classes. These classes encompass nine specific diseases commonly found in tomato plants, along with a class representing healthy tomato leaves. The specific classes within the dataset are as follows:

- Early Blight
- Late Blight
- Septoria Leaf Spot
- Bacterial Spot
- Spider Mites
- Target Spot
- Yellow Leaf Curl Virus
- Mosaic Virus
- Healthy Leaf

To prepare the dataset for training, validation, and testing, the dataset was divided into three subsets using an 85/15 split for training and validation, with a separate directory for the test set which is used to test the web application.

Before training the models, the images were preprocessed using Pytorch's transforms library, which resized the images to a fixed size of 224x224 pixels and normalised their

pixel values to be between 0 and 1. The resizing ensured that all images were the same size, making it easier for the models to process the data. The normalisation was performed to ensure that the pixel values were in a consistent range, which is important for optimising the models' performance.

After training we have also done background augmentation on the dataset where we have superimposed leaf images on different background images . By doing this type of augmentation the accuracy of detecting diseases from leaf images increases as all images which will be uploaded by the user will not be in plain white/black background. These images will help the model to detect the diseases from the leaf irrespective of the background of the leaf in the image which will be uploaded.

Furthermore, we have also done data annotation by giving proper index , image path and image label .The process of labeling data with relevant tags to make it easier for computers to understand and interpret. This will also help the model to map the images in a more optimal way.

	label	path
0	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
1	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
2	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
3	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
4	Tomato___Late_blight	/content/our dataset/train/Tomato___Late_bligh...
...	...	...
25205	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25206	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25207	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25208	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...
25209	Tomato___Target_Spot	/content/our dataset/train/Tomato___Target_Spo...

25210 rows × 2 columns

fig8. Annotated data frame created for both training and validation dataset

Attributes:

- Image files: The dataset consists of 30,000 image files in JPG format.
- Labels: Each image in the dataset is labelled with the name of the diseases or healthy and a numeric number.
- Train and validation sets: The dataset is divided into two main folders: "train" and "valid", with 25,200 and 4,800 images, respectively.

Dataset Split	Number of Images
Training data	25,200
validation data	4,800
Total Result	30,000

In summary, our tomato leaf disease detection project leverages a meticulously organized dataset comprising approximately 30,000 images categorized into ten distinct classes, including common tomato plant diseases and a class for healthy leaves. We've ensured data consistency through standardized preprocessing,normalization of dataset , introduced background augmentation for robustness, and enhanced dataset interpretability through data annotation. With these strong foundations, our project is well-positioned to train effective models and deploy a user-friendly web application for accurate tomato leaf disease diagnosis.