CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

A Project Report on

BLIGHT DISEASE DETECTION IN CROPS WITH CNN AND FAST API

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

by

Atle Clive Churchill 1960316

Under the Guidance of

Dr Vandana Reddy

and

Dr Bijeesh T V

Department of Computer Science and Engineering

School of Engineering and Technology, CHRIST (Deemed to be University),

Kumbalgodu, Bangalore - 560 074

March 2024

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

School of Engineering and Technology

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that **Atle Clive Churhill** has successfully completed the project work entitled " **BLIGHT DISEASE DETECTION IN CROPS WITH CNN AND FAST API**" in partial fulfillment for the award of **Bachelor of Technology** in **Computer Science and Engineering** during the year **2022-2023**.

Dr Vandana Reddy	**Dr Bijeesh T V**
Guide Designation	Co-Guide Designation
Dr. K Balachandran	**Dr. Iven Jose**
Head of Department	Dean

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

School of Engineering and Technology

Department of Computer Science and Engineering

BONAFIDE CERTIFICATE

It is to certify that this project titled " BLIGHT DISEASE DETECTION IN CROPS WITH CNN AND FAST API " is the bonafide work of

Name	Register Number
Atle Clive Churchill	1960316

Examiners [Name and Signature]	Name of the Candidate :
1.	Register Number :
2.	Date of Examination :

Industry Certificate

Certificate PDF File has to kept in the Pictures Folder with the name IndustryCertificate.pdf

Acknowledgement

I would like to thank CHRIST (Deemed to be University)Vice Chancellor in-charge and the Pro Vice Chancellor **Dr. Rev. Fr. Joseph C C**, Director of School of Engineering and Technology, **Dr. Rev. Fr. Benny Thomas** and the Dean **Dr. Iven Jose** for their kind patronage.

I would like to express my sincere gratitude and appreciation to the Head of Department of Department of Computer Science and Engineering, School of Engineering and Technology **Dr. K Balachandran**, for giving me this opportunity to take up this project.

I also extremely grateful to my guide, **Dr Vandana Reddy**, who has supported and helped to carry out the project. Her constant monitoring and encouragement helped me keep up to the project schedule.

I also extremely grateful to my co-guide, **Dr Bijeesh T V**, who has supported and helped to carry out the project. His constant monitoring and encouragement helped me keep up to the project schedule.

Declaration

I, Hereby declare that the Project titled " **BLIGHT DISEASE DETECTION IN CROPS WITH CNN AND FAST API** " is a record of original project work undertaken by us for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering**. I have completed this study under the supervision of **Dr Vandana Reddy**, Department of Computer Science and Engineering and **Dr Bijeesh T V**, Computer Science and Engineering.

I also declare that this project report has not been submitted for the award of any degree, diploma, associate ship, fellowship or other title anywhere else. It has not been sent for any publication or presentation purpose.

Place: School of Engineering and Technology, CHRIST (Deemed to be University), Bangalore
Date:

Name	Register Number	Signature
Atle Clive Churchill	1960316	

Abstract

Modernizing agricultural practices is essential for ensuring food security and sustainable farming. This project focuses on leveraging advanced technologies to address one of the most significant challenges faced by farmers—crop diseases. The core of the system is a Convolutional Neural Network (CNN) model, inspired by Krizhevsky et al.'s pioneering work, specifically tailored for detecting blight diseases in crops. The CNN model, trained on diverse datasets, exhibits an outstanding accuracy rate of 98-99

Data augmentation techniques inspired by Liu et al. and regularization enhancements proposed by DeVries and Taylor contribute to the robustness of the CNN model. TensorFlow serves as the underlying framework for large-scale machine learning, and TensorFlow Serving, as proposed by Al-Rfou et al., ensures industry-scale deployment, flexibility, and high-performance model serving. The entire system is orchestrated by FastAPI, a rapid development framework that seamlessly integrates the frontend, backend, and the TensorFlow Serving API.

The project's future scope envisions the expansion of the CNN model to encompass a myriad of plant species and a broader spectrum of diseases. This inclusivity is crucial for fostering precision agriculture and enhancing crop health management globally. Additionally, the development of a user-friendly mobile app using ReactJS and Uvicorn extends the accessibility of the technology, empowering farmers with a portable, on-the-go solution for disease detection. Through this comprehensive integration of cutting-edge technologies, the project aims to revolutionize agriculture, bridging the gap between traditional farming practices and innovative, data-driven solutions. The proposed system not only promises immediate benefits for farmers but also positions itself as a transformative force in shaping the future of agriculture.

Contents

LIST OF FIGURES

Chapter 1

INTRODUCTION

Agriculture plays a pivotal role in India's economy and sustenance, serving as the backbone for the livelihoods of a significant portion of the population. With over 58% of the country's workforce engaged in agriculture, it remains the primary source of employment for a vast number of people, especially in rural areas. The diverse agro-climatic zones in India enable the cultivation of a wide variety of crops, ranging from cereals like rice and wheat to fruits, vegetables, and spices. The sector not only provides employment but also contributes significantly to the country's GDP. The success of agriculture is closely linked to India's food security, as it caters to the dietary needs of a billion-plus population.

Moreover, the agricultural sector is essential for addressing poverty and fostering rural development. A substantial portion of India's population resides in rural areas, where agriculture is not just an economic activity but also a way of life. The success of crops directly impacts the well-being of these communities, influencing their economic stability and overall quality of life. Additionally, the growth of the agriculture sector has a cascading effect on other industries, including agribusiness, food processing, and transportation. Thus, the vitality of agriculture in India extends beyond its immediate economic impact, playing a crucial role in the overall socio-economic development of the nation.

The primary objective of a plant disease detection system is to safeguard and optimize agricultural productivity by swiftly identifying and addressing diseases affecting crops. These systems leverage advanced technologies, such as machine learning, image processing, and sensor technologies, to detect early signs of diseases in plants. Early

detection is crucial as it allows for timely intervention, preventing the spread of diseases and minimizing crop losses. By utilizing various data inputs, including images of plant leaves, soil conditions, and environmental factors, these systems can accurately diagnose and classify diseases, providing farmers with actionable insights.

Efficient plant disease detection systems contribute to sustainable agriculture by facilitating precision farming practices. By identifying diseases at an early stage, farmers can implement targeted and precise treatment strategies, reducing the need for excessive pesticide use and minimizing environmental impact. The overall goal is to enhance crop yield, quality, and resilience, ensuring food security and economic stability for agricultural communities. Through the integration of technology and data-driven approaches, plant disease detection systems empower farmers with timely information, enabling them to make informed decisions and adopt preventive measures, thereby supporting a more resilient and sustainable agricultural ecosystem.

In this report, I provide an overview of the project, detailing the system architecture, key features, and implementation of the convolution neural network. I also discuss the methodology for data collection and pre-processing, as well as the results of my analysis. Through this project, I aim to provide a rapid and accurate means of detecting, classifying, and diagnosing diseases affecting crops. By leveraging advanced algorithms and image recognition technology, the software aims to analyze visual cues, such as leaf patterns or discoloration, and promptly identify potential diseases. This proactive identification allows farmers to implement timely and targeted interventions, minimizing crop losses, optimizing yields, and promoting sustainable agricultural practices.

1.1 Problem Identification

Unchecked plant leaf diseases pose a multifaceted threat to agricultural ecosystems, giving rise to several critical challenges with far-reaching consequences. Firstly, the diminished crop yield resulting from the unchecked spread of diseases hampers agricultural productivity, leading to significant economic losses for farmers. As the diseases compromise the vitality of crops, they not only undermine the financial stability of farming communities but also contribute to a broader concern of reduced food security. The potential scarcity of essential crops raises alarms about the availability and affordability of food, impacting both local economies and the nutritional well-being of communities. Furthermore, the environmental repercussions of untreated diseases, often exacerbated

by increased pesticide use, compound the challenges by compromising the sustainability of agriculture and contributing to ecological imbalances. Addressing these problems necessitates timely detection and effective treatment strategies to safeguard crop yields, economic stability, and overall food security.

1.1.1 Reduced Crop Yield:

Leaving plant leaf diseases undetected or untreated can significantly impact crop yield. Diseases often weaken plants, affecting their ability to photosynthesize and absorb nutrients, ultimately leading to stunted growth and decreased productivity. Unchecked diseases can spread rapidly throughout a crop, causing widespread damage and resulting in substantial reductions in yield.

1.1.2 Economic Losses for Farmers:

The untreated spread of plant leaf diseases can result in economic losses for farmers. Reduced crop yields mean lower profits, as farmers may not be able to sell as much produce in the market. Additionally, farmers might incur additional costs for ineffective treatments or increased pesticide use when diseases are not identified and addressed promptly. Economic losses can have long-lasting consequences for the financial stability of farming communities.

1.1.3 Food Security Concerns:

Plant diseases left untreated pose a threat to food security, especially in regions heavily dependent on agriculture. Reduced crop yields mean less food available for consumption and sale, potentially leading to food shortages and increased prices. This can impact not only the livelihoods of farmers but also the overall nutritional well-being of communities that rely on these crops as staple foods.

1.1.4 Environmental Impact:

In cases where diseases are not effectively managed, farmers may resort to increased use of chemical pesticides to control the spread. This can have adverse environmental

effects, including soil and water contamination, harm to non-target organisms, and the development of pesticide-resistant strains of pathogens. The environmental impact of untreated plant diseases can exacerbate ecological imbalances and pose long-term risks to agricultural sustainability.

1.2 Problem Formulation

The identified problem of unchecked plant leaf diseases in agricultural ecosystems necessitates a systematic formulation to enhance clarity on the problem statement and objectives. The overarching issue revolves around the multifaceted threats posed by these diseases, leading to critical challenges with significant consequences for both farmers and broader communities. The primary objective is to establish a comprehensive understanding of the problem to guide the development of effective strategies for detection, intervention, and mitigation.

1.2.0.1 Enhance Timely Detection:

To address the imperative of enhancing timely detection of plant diseases, the development and implementation of advanced detection systems are paramount. Leveraging cutting-edge technologies such as machine learning and image processing offers a promising avenue. These systems can be designed to analyze intricate patterns, color changes, and subtle anomalies in plant leaves, enabling early identification of diseases that may not be easily discernible to the human eye. By integrating real-time monitoring and data analytics, these systems can provide farmers with prompt alerts, allowing for swift and precise intervention. The utilization of technology not only enhances the speed of detection but also contributes to accuracy, reducing the risk of misdiagnosis and ensuring that appropriate measures are taken in a timely manner. In essence, these advanced detection systems empower farmers with proactive tools to safeguard their crops effectively.

1.2.0.2 Mitigate Economic Losses:

Mitigating economic losses stemming from plant diseases requires a multifaceted approach. Targeted intervention strategies should focus on optimizing resource utilization,

minimizing dependence on ineffective treatments, and adopting pest management practices that are both efficient and sustainable. Precision agriculture techniques, such as variable rate application of inputs based on real-time data, can optimize the use of fertilizers, pesticides, and water, ensuring resources are deployed where they are needed most. Additionally, the development of resistant crop varieties and the promotion of integrated pest management (IPM) practices can contribute to reducing the reliance on chemical interventions. By empowering farmers with knowledge, tools, and sustainable practices, economic losses can be minimized, fostering greater resilience in the face of plant diseases and promoting long-term economic stability for agricultural communities.

1.2.0.3 Ensure Food Security:

Ensuring food security in the context of plant diseases involves a holistic approach that spans sustainable agricultural practices, crop resilience enhancement, and targeted disease management strategies. Promoting sustainable agriculture entails practices such as crop rotation, agroforestry, and organic farming, which contribute to soil health and reduce the risk of disease outbreaks. Enhancing crop resilience involves developing and cultivating disease-resistant varieties through advanced breeding techniques and genetic engineering. Additionally, providing farmers with training and support in implementing best practices for disease prevention and management plays a crucial role. By addressing food security concerns at both the production and distribution levels, a more resilient and sustainable food system can be established, reducing vulnerabilities to the impact of plant diseases on essential food crops.

1.2.0.4 Promote Environmental Sustainability:

Promoting environmental sustainability in the management of plant diseases requires a shift towards eco-friendly approaches that minimize the reliance on chemical pesticides. Integrated pest management (IPM) practices, which encompass biological control, cultural practices, and the use of natural predators, offer a more balanced and environmentally friendly approach. By reducing the use of chemical pesticides, the risk of soil and water contamination is diminished, and non-target organisms are spared from potential harm. Additionally, the development of disease-resistant crop varieties through conventional breeding or genetic engineering can contribute to reducing the need for extensive

pesticide applications. Emphasizing ecological balance in disease management ensures the preservation of biodiversity and the long-term sustainability of agricultural ecosystems. Overall, these measures align with a broader commitment to environmentally conscious agricultural practices, mitigating the environmental repercussions of unchecked plant diseases.

By formulating the problem in this systematic manner, the aim is to provide a clear and comprehensive framework for addressing the challenges associated with unchecked plant leaf diseases in agriculture. This will guide the development of targeted solutions to enhance crop yields, ensure economic stability for farmers, secure food availability, and promote environmental sustainability in agricultural ecosystems.

1.3 Problem Statement & Objectives

The identified problem stems from the complex interplay of factors contributing to the uncontrolled spread of plant leaf diseases. The absence of a comprehensive strategy undermines the resilience of agricultural systems, impacting farmers' economic stability, threatening food security, and posing risks to environmental sustainability. Addressing this problem requires a concerted effort to develop and implement effective solutions that integrate advanced technologies, sustainable practices, and a commitment to environmentally conscious agriculture. To overcome the aforementioned problems there is a need for a holistic solution where the main objective is to develop an easy to use, accurate yet quick model that can predict or identify the type of disease so that necessary action can be taken at the earliest stages and minimise losses.

1.3.1 Major Objectives:

1.3.1.1 Proactive Intervention Through Early Detection:

Early detection of diseases in crops is a cornerstone of modern agricultural practices, providing farmers with a proactive tool to safeguard their crops and mitigate potential threats. Leveraging advanced technologies such as machine learning and image processing enables the identification of subtle signs of diseases before visible symptoms

escalate. This prompt action allows for the targeted application of fungicides, pesticides, or other disease management measures, effectively curbing the spread of the disease.

1.3.1.2 Preservation of Optimal Crop Yields:

Beyond immediate intervention, early detection plays a crucial role in preventing significant yield losses. Diseases, if left unchecked, can devastate crops and compromise overall productivity. Detecting diseases in their early stages allows farmers to address the issue before it causes extensive damage, preserving optimal crop yields. This not only contributes to the economic viability of farming operations but also ensures a stable and consistent food supply, thus enhancing global and local food security.

1.3.1.3 Economic Efficiency Through Cost Reduction:

Early detection also translates into cost reduction for farmers. Swift intervention at the early stages of a disease is often more effective and requires fewer resources. Farmers can avoid unnecessary expenditures on extensive treatments or excessive use of pesticides that might be needed if diseases are allowed to progress. This optimized resource use is not only economically beneficial but also aligns with sustainable farming practices.

1.3.1.4 Precision Agriculture for Resource Optimization:

Furthermore, early detection facilitates the precise and targeted application of resources such as water, fertilizers, and pesticides. Guided by data from early detection systems, farmers can tailor their agricultural practices to the specific needs of the crops, minimizing wastage and environmental impact. This application of precision agriculture techniques contributes to resource efficiency and overall sustainability.

1.3.1.5 Quality Assurance for Market Competitiveness:

Preserving the quality of crops is another significant advantage of early disease detection. Diseases not only affect the quantity of the harvest but also compromise the quality of the produce. Detecting diseases early helps farmers maintain the desired quality

attributes of their crops, ensuring compliance with market standards and meeting consumer preferences. High-quality crops enhance market value and consumer satisfaction, bolstering economic returns for farmers.

1.3.1.6 Preventing Disease Spread for Ecosystem Health:

Finally, early detection supports the prevention of disease spread within agricultural ecosystems. By isolating and treating infected plants at an early stage, farmers can contain the disease, preventing it from spreading throughout the entire field or to neighboring farms. This proactive approach contributes to the overall health and resilience of the agricultural environment, minimizing the risk of widespread outbreaks and supporting the long-term sustainability of farming practices.

1.3.1.7 Accurate Detection of Diseases:

AI detection of diseases in crops surpasses human capabilities in various aspects, primarily due to its unparalleled speed, accuracy, and consistency. Machine learning algorithms employed in AI systems can swiftly process vast datasets, enabling rapid and precise identification of subtle patterns or anomalies indicative of crop diseases. The ability to operate continuously and provide real-time monitoring ensures early detection, a critical factor in preventing the spread of diseases and mitigating their impact on crop yields. AI's scalability allows it to cover expansive agricultural areas, providing a cost-effective solution for large farms that would be challenging for manual inspection. Moreover, AI models exhibit consistent performance across diverse environmental conditions, lighting scenarios, and crop types, reducing the likelihood of human biases and ensuring standardized results.

Furthermore, AI systems possess a learning and adaptive capability, improving their accuracy over time as they process more data. This adaptive feature enables them to stay ahead of emerging diseases and evolving patterns, contributing to their effectiveness in proactive disease management. The cost-effectiveness of AI, once implemented, is notable, as it reduces the dependency on continuous human labor in disease monitoring. In essence, AI detection in agriculture enhances the efficiency of disease detection processes, supporting farmers in maintaining crop health, optimizing yields, and fostering sustainable agricultural practices.

1.4 Limitations

Certain aspects of the project are of-course limited which does not tamper the quality of the idea however they only provide scope to be further improved upon.

1.4.0.1 Limited scope of detection:

An AI model tailored to detect a specific type of crop disease offers notable advantages in terms of precision and efficiency within its defined scope. It can rapidly analyze large datasets, identifying subtle patterns indicative of the targeted disease and enabling timely intervention. This specialization enhances accuracy and speed, providing a valuable tool for farmers dealing with known and prevalent diseases. However, the limitation of such AI models becomes apparent when confronted with novel or varied diseases not included in their training data. The lack of adaptability and contextual understanding may hinder their effectiveness in handling unforeseen agricultural challenges.

In contrast, a human expert possesses a holistic and adaptable approach to crop disease management. Drawing on experience, a human expert can recognize unfamiliar patterns, consider a multitude of factors influencing crop health, and adapt strategies to address emerging diseases or changing environmental conditions. The ability to connect disparate pieces of information and apply critical thinking makes human expertise invaluable in handling the complexity and diversity inherent in agriculture. Thus, the limitation of AI models specialized in a single disease type lies in their inability to match the versatility, adaptability, and contextual understanding exhibited by human experts in the field.

1.4.0.2 External Factors cannot be considered

An AI model's limitation lies in its confined detection parameters, rendering it susceptible to inaccuracies and false positives. Trained on specific datasets, these models excel at recognizing patterns related to their designated tasks, such as crop disease detection. However, their rigid focus prevents them from considering a broader context, leading to potential misinterpretations. For instance, an AI model may generate false positives if it encounters harmless variations or external factors mimicking disease symptoms but not accounted for in its training data. The lack of adaptability and contextual understanding

makes AI models prone to false truths, emphasizing the need for human expertise to navigate complex and unforeseen variables in real-world scenarios.

Human experts, with their ability to comprehend diverse factors and contextualize information, play a vital role in addressing the limitations of AI models. Their capacity to consider external influences, draw on experience, and exercise critical thinking allows them to discern between genuine indicators of a problem and potential confounding factors leading to false positives. The combination of AI models and human expertise offers a more robust approach, harnessing the precision of technology and the adaptability of human intelligence to enhance the accuracy and reliability of complex decision-making tasks, such as disease detection in agriculture.

1.4.0.3 Rare outliers in some cases

An AI model's limitation lies in its confined detection parameters, rendering it susceptible to inaccuracies and false positives. Trained on specific datasets, these models excel at recognizing patterns related to their designated tasks, such as crop disease detection. However, their rigid focus prevents them from considering a broader context, leading to potential misinterpretations. For instance, an AI model may generate false positives if it encounters harmless variations or external factors mimicking disease symptoms but not accounted for in its training data. The lack of adaptability and contextual understanding makes AI models prone to false truths, emphasizing the need for human expertise to navigate complex and unforeseen variables in real-world scenarios.

Human experts, with their ability to comprehend diverse factors and contextualize information, play a vital role in addressing the limitations of AI models. Their capacity to consider external influences, draw on experience, and exercise critical thinking allows them to discern between genuine indicators of a problem and potential confounding factors leading to false positives. The combination of AI models and human expertise offers a more robust approach, harnessing the precision of technology and the adaptability of human intelligence to enhance the accuracy and reliability of complex decision-making tasks, such as disease detection in agriculture.

1.4.0.4 Technology and Resource Constraints:

In some instances, farmers may face technical limitations that hinder their ability to understand and operate advanced AI programs or software designed for agriculture.

These technical challenges could stem from a lack of familiarity with digital technologies, limited access to training, or insufficient technical infrastructure in rural areas. The complexity of AI systems might pose a barrier for farmers who are not well-versed in technology, and without adequate support or training, they may struggle to harness the full potential of these tools.

Furthermore, resource constraints can contribute to the farmer's inability to adopt or effectively use AI-driven programs. Limited access to computing devices, reliable internet connectivity, or the financial means to invest in sophisticated technology can impede a farmer's capacity to integrate AI solutions into their agricultural practices. The cost associated with implementing and maintaining AI technologies may be prohibitive for small-scale farmers, exacerbating the digital divide and preventing them from benefiting from the potential efficiencies and insights offered by these tools. Bridging this gap in resources and providing tailored support and education is crucial for ensuring that farmers, regardless of their technical expertise or financial means, can leverage AI technologies to enhance their agricultural operations.

Chapter 2

RESEARCH METHODOLOGY

2.0.1 Introduction

For the disease detection model utilizing Convolutional Neural Network (CNN) with data augmentation and TensorFlow serving, an experimental research design was employed. Experimental research involves manipulating one or more independent variables to observe their effects on a dependent variable in a controlled environment. In this context, the AI model's architecture, training parameters, and data augmentation techniques can be considered as independent variables, while the model's performance, accuracy, and robustness would be the dependent variables. Conducting experiments allows for systematic testing of hypotheses and causal relationships, providing insights into the effectiveness of different components and configurations of the AI model.

The experimental design would involve systematically varying the input factors, such as adjusting CNN architectures, modifying data augmentation strategies, or tuning hyperparameters during the training process. The goal is to assess the impact of these variations on the model's ability to accurately detect diseases. By randomizing and controlling these factors, researchers can isolate the effects of individual components and draw more confident conclusions about their contributions to the overall system performance. Additionally, experimental designs allow for statistical analysis to quantify the significance of observed differences and to draw generalizable conclusions about the AI model's efficacy.

Furthermore, the use of observational and analytical research designs can complement the experimental approach. Observational studies could be employed to gather insights

from real-world scenarios, such as assessing the model's performance across diverse patient populations or clinical settings. Analytical research may involve in-depth analysis of the model's predictions, examining factors influencing its decision-making process, and identifying potential biases or limitations. Integrating these approaches would provide a comprehensive understanding of the disease detection AI model's strengths, weaknesses, and real-world applicability.

2.0.2 Research Type

Quantitative methods would be instrumental in measuring and analyzing the numerical aspects of the AI model's performance. This involves assessing metrics such as accuracy, precision, recall, F1 score, and other quantitative indicators that provide objective measurements of the model's ability to detect diseases in leaf samples. Quantitative research allows for statistical analysis to determine the significance of observed differences, enabling researchers to draw conclusions about the model's efficiency and effectiveness in a numerical and statistically robust manner.

Qualitative methods are valuable for gaining deeper insights into the subjective aspects of the research, including the interpretative understanding of the AI model's performance and the quality of disease detection. This involves qualitative assessments of the model's outputs, such as examining the nature of false positives or false negatives, understanding the model's decision-making process, and exploring user experiences or perceptions of the system. Qualitative research helps uncover nuances, contextual factors, and potential limitations that may not be fully captured by quantitative metrics alone.

However by combining quantitative and qualitative research, researchers can achieve a more holistic evaluation of the disease detection AI model. This hybrid approach allows for a thorough investigation into the model's accuracy and precision (quantitative) while also exploring the contextual, user-related, and interpretative aspects of its performance (qualitative). This mixed-methods design enhances the overall validity and reliability of the research findings, providing a more nuanced understanding of the AI model's capabilities and potential areas for improvement. Therefore, a mixture of quantitative and qualitative research methods was best suited for this project for evaluation ensuring a well-rounded assessment that encompasses both numerical performance metrics and qualitative insights.

2.0.3 Data Collection

The target population consists of potato plants affected by various diseases. Specifically, it includes a diverse set of potato leaf samples that exhibit symptoms of common diseases such as late blight, early blight, potato scab, etc. The model aims to generalize well across different types of diseases to ensure its applicability in real-world scenarios. Considering the variability in disease manifestations, a stratified sampling technique would be most appropriate. Stratified sampling involves dividing the population into homogeneous subgroups or strata based on certain characteristics. In this case, the strata could represent different types of potato diseases. This ensures that each disease category is well-represented in the sample, preventing the dominance of one disease type and enabling the model to learn effectively from diverse instances.

The best sample size depends on the complexity of the task, the diversity of diseases, and the desired level of statistical confidence. A larger sample size is generally preferable for training a robust model. However, it's crucial to strike a balance between sufficiency and computational feasibility. Aim for a sample size that adequately represents the diversity of diseases while being manageable for training purposes. Selection criteria should prioritize diversity within each stratum to ensure a comprehensive representation of different disease types. Additionally, consider factors such as variations in lighting conditions, growth stages of plants, and environmental factors to enhance the model's robustness. Randomly selecting samples within each stratum helps avoid bias and ensures a more representative dataset for training and testing the AI model. By employing a stratified sampling technique with an appropriate sample size and selection criteria, researchers can enhance the model's ability to generalize across various potato leaf diseases. This approach ensures that the AI model is trained on a well-balanced and diverse dataset, contributing to its effectiveness in real-world applications.

2.0.4 Statistical Analysis

A confusion matrix provides a comprehensive breakdown of the model's performance, specifically for a classification task like disease detection. By analyzing true positives, true negatives, false positives, and false negatives, researchers can calculate key metrics such as precision, recall, accuracy, and F1 score. These metrics are essential for assessing the model's ability to correctly identify blight stages and distinguish between healthy and diseased plants. ROC curves are useful for evaluating the trade-off between

sensitivity and specificity at various classification thresholds. AUC provides a single scalar value summarizing the model's overall performance. This analysis is particularly relevant for disease detection, as it allows researchers to assess the model's ability to differentiate between different blight stages and healthy plants. A higher AUC indicates better discrimination capabilities.

For CNN-based models, understanding the importance of different features (in this case, image regions) can provide insights into the aspects the model considers when identifying blight stages. Techniques like gradient-weighted class activation mapping (Grad-CAM) can visualize the regions of the image contributing most to the model's decision. Analyzing feature importance aids in interpretability and can reveal patterns associated with various disease stages. If the dataset includes temporal information or sequential images of plant growth, time series analysis can be beneficial. This analysis helps identify patterns and trends in disease progression over time, aiding in the detection of blight stages. Techniques like Fourier analysis or autoregressive integrated moving average (ARIMA) modeling may be employed, depending on the nature of the data. If there are multiple versions of the AI model or variations in experimental conditions, statistical significance testing (e.g., t-tests or ANOVA) can be employed to determine if observed differences in performance metrics are statistically significant. This ensures that any improvements or variations in detecting blight stages are not due to random chance.

By employing these statistical and analytical methods, researchers can rigorously evaluate the potato leaf disease detection AI model's performance in identifying blight stages and assessing the healthiness of plants. These methods collectively provide a holistic understanding of the model's strengths and weaknesses in achieving the project objectives.

2.0.5 Ethical Considerations

Ethical considerations play a pivotal role in safeguarding participant rights and ensuring responsible research practices. To address informed consent, researchers should provide a comprehensive and understandable explanation of the project's purpose, risks, and benefits. Participants must be fully aware of the voluntary nature of their involvement and be given the option to opt in or out without facing consequences. The use of a detailed informed consent form, clearly outlining project details and participant rights,

along with ongoing communication about project progress, fosters transparency and ensures participants are informed throughout the research process.

Participant confidentiality is paramount in ethical research. To protect participant privacy, data anonymization is crucial, involving the removal of personally identifiable information from collected images or associated metadata. Robust security measures should be implemented for data storage, limiting access to those directly involved in the research, and utilizing encryption techniques for data transmission. Seeking approval from an ethical review board or institutional review board (IRB) ensures that the project's design and data handling practices align with ethical standards. Regular ethical audits further support ongoing compliance and adjustments as needed to maintain participant confidentiality.

Adherence to broader ethical standards involves transparent communication about the limitations of the AI model, including potential biases, in any project-related publications or communications. Fair treatment of all participants is essential, avoiding discrimination and ensuring inclusivity in the dataset. Researchers should engage with the community or relevant stakeholders, if applicable, to address concerns and gather feedback, fostering transparency and collaboration. By incorporating these ethical practices, researchers contribute to the responsible and trustworthy advancement of AI technology, especially in critical areas such as agriculture and healthcare.

Future Scope

The project holds substantial future scope by expanding its capabilities to detect a broader range of diseases, including those affecting various crop types. The integration of additional disease categories and crop types can significantly enhance the model's versatility and relevance in diverse agricultural settings. By incorporating a more comprehensive dataset that includes diseases affecting crops beyond potatoes, such as tomatoes, rice, or wheat, the AI model can be adapted to address the unique challenges faced by farmers cultivating different crops. This expansion not only contributes to the agricultural sector's overall resilience but also positions the AI model as a valuable tool for a wider audience.

Furthermore, extending the project's scope to include a variety of crops aligns with the growing need for sustainable and technology-driven farming practices. Farmers cultivating different crops can benefit from a versatile disease detection model that provides accurate and timely assessments, leading to improved crop management and yield optimization. This expanded scope aligns with the broader goal of advancing precision

agriculture, where AI technologies play a pivotal role in ensuring the health and productivity of diverse crops.

Additionally, enhancing the front-end user interface by developing a mobile API makes the model more accessible and user-friendly. Enabling farmers to use the AI model through a mobile application simplifies the interaction process, making it convenient for users with varying levels of technical expertise. This approach empowers farmers to perform on-the-spot disease assessments directly in the field, facilitating quicker decision-making and intervention strategies. The mobile API could provide real-time feedback on disease detection, offering actionable insights to farmers and contributing to more proactive crop management practices.

In summary, the future scope of the potato leaf disease detection AI model is immense, and its impact can be significantly amplified by expanding disease detection capabilities to various crop types. This approach aligns with the evolving needs of the agricultural sector, emphasizing precision farming and technological solutions. Moreover, the development of a user-friendly mobile API enhances accessibility, ensuring that the benefits of the AI model reach a broader user base, including farmers who may not have extensive technological expertise. This future-oriented expansion positions the project as a valuable and versatile tool with the potential to revolutionize disease detection and crop management practices across diverse agricultural landscapes.

Chapter 3

LITERATURE SURVEY AND REVIEW

Sl. No	Research Paper	Publisher
1	ImageNet Classification with Deep Convolutional Neural Networks”	A. Krizhevsky, I. Sutskever, and G. Hinton, ”ImageNet Classification with Deep Convolutional Neural Networks,” in Neural Information Processing Systems (NeurIPS), 2012
2	Data Augmentation for Object Detection via Progressive and Selective Instance-Switching	Y. Liu et al., ”Data Augmentation for Object Detection via Progressive and Selective Instance-Switching,” in European Conference on Computer Vision (ECCV), 2019
3	Improved Regularization of Convolutional Neural Networks with Cutout	T. DeVries and G. W. Taylor, ”Improved Regularization of Convolutional Neural Networks with Cutout,” in arXiv preprint arXiv:1708.04552, 2017
4	TensorFlow: A system for large-scale machine learning	M. Abadi et al., ”TensorFlow: A system for large-scale machine learning,” in Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI), 2016
5	TensorFlow Datasets: A Swiss Army Knife for Building Machine Learning Workloads	M. Al-Rfou et al., ”TensorFlow Datasets: A Swiss Army Knife for Building Machine Learning Workloads,” in arXiv preprint arXiv:1803.09820, 2018

Sl. No	Research Paper	Publisher
6	TensorFlow Serving: Towards Industry-Scale Machine Learning	M. Al-Rfou et al., "TensorFlow Serving: Towards Industry-Scale Machine Learning," in Proceedings of the 2nd SysML Conference, 2019.
7	TensorFlow Serving: Flexible, High-Performance ML Serving	N. Van Hoey et al., "TensorFlow Serving: Flexible, High-Performance ML Serving," in Proceedings of the ACM Symposium on Cloud Computing (SoCC), 2017
8	React: Declarative, Component-Based, Learn Once, Write Anywhere	J. Walke and S. Chan, "React: Declarative, Component-Based, Learn Once, Write Anywhere," in Proceedings of the 5th International Workshop on Web Component Development, 2013
9	FastAPI: Fast (to develop), Fast (to run)	S. B. Uvic, "FastAPI: Fast (to develop), Fast (to run)," in Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, 2021

3.1 Literature Collection & Segregation

[1] TensorFlow: A system for large-scale machine learning (2016): In this groundbreaking paper, authored by Abadi et al., TensorFlow emerges as a pioneering open-source framework tailored for large-scale machine learning endeavors. Released by Google, TensorFlow revolutionizes the landscape of machine learning by providing a robust and scalable platform for constructing and deploying intricate neural network models. The paper meticulously unveils the core design principles and features that render TensorFlow an indispensable tool in the arsenal of machine learning practitioners, catalyzing advancements in the development and deployment of models on a massive scale.

[2] TensorFlow Serving: Flexible, High-Performance ML Serving (2017): Van Hoey et al. contribute significantly to the realm of machine learning deployment with their comprehensive exploration of TensorFlow Serving in this impactful paper. It delves into the intricacies of TensorFlow Serving, a flexible and high-performance system designed for the seamless deployment of machine learning models in production environments. By elucidating the architecture and highlighting its efficiency, the paper establishes TensorFlow Serving as a pivotal reference for researchers and industry professionals navigating the complexities of serving machine learning models at scale.

[3] TensorFlow Serving: Towards Industry-Scale Machine Learning (2019): Building upon the foundation laid in the previous paper, Al-Rfou et al. extend the discourse

on TensorFlow Serving, emphasizing its pivotal role in achieving industry-scale machine learning deployment. The authors meticulously detail the architectural enhancements and optimizations incorporated into TensorFlow Serving, showcasing its adaptability and robustness in real-world scenarios. This paper stands as a testament to the continuous evolution of TensorFlow Serving to meet the demands of large-scale and industry-grade machine learning production.

[4] TensorFlow Datasets: A Swiss Army Knife for Building Machine Learning Workloads (2018): The realm of machine learning datasets witnesses a transformative tool with the introduction of TensorFlow Datasets by Al-Rfou et al. This paper unravels the intricacies of TensorFlow Datasets, presenting it as a versatile and indispensable resource for simplifying the intricacies of building machine learning workloads. The authors meticulously elucidate its design, functionality, and its pivotal role as a comprehensive repository, offering researchers and practitioners a streamlined approach to accessing and managing diverse datasets.

[5] TensorFlow Datasets: A Community-Driven Collection of Large and Diverse Datasets for Machine Learning (2018): Al-Rfou et al. reprise their collaborative efforts in this paper, shedding light on the communal spirit that fuels TensorFlow Datasets. The research underscores the vibrant and inclusive nature of this community-driven initiative. The authors discuss the collective endeavors in curating and maintaining a rich collection of datasets, fostering a collaborative ecosystem that empowers machine learning practitioners to access and contribute to a diverse array of high-quality datasets.

[6] FastAPI: Fast (to develop), Fast (to run) (2021): Stepping into the realm of web development, Uvic introduces FastAPI as a game-changer in this paper. FastAPI, a Python web framework, distinguishes itself with a unique blend of rapid development and runtime efficiency. Uvic meticulously outlines the design principles, features, and the performance advantages that position FastAPI as a compelling choice for developers seeking swiftness both in the development phase and when their applications are in full operation.

[7] React: Declarative, Component-Based, Learn Once, Write Anywhere (2013): Walke and Chan pioneer the narrative on modern web development with their paper on React.js. This declarative and component-based JavaScript library becomes a beacon in UI development, as the paper introduces the principles that define React. Emphasizing declarative nature, component architecture, and the philosophy of ”learn once,

write anywhere," React becomes not just a library but a paradigm shift, influencing how developers approach the creation of dynamic user interfaces.

[8] React: Facebook's Functional UI Framework (2016): Stever's exploration of React delves deep into the functional underpinnings of Facebook's UI framework. In this paper, React transcends its role as a JavaScript library, evolving into a functional programming powerhouse. By harnessing functional programming concepts, React stands as a testament to how incorporating these principles can revolutionize the creation of user interfaces, contributing to its widespread adoption and influence in the web development landscape.

[9]React: The Road to Server-Side Rendering (2020): The journey of React takes a pivotal turn with Ong and Olsson's paper, focusing on the implementation of server-side rendering (SSR). This optimization becomes a critical aspect of web development, enhancing the performance and user experience of React applications. The authors unravel the motivations, challenges, and benefits of adopting SSR, providing invaluable insights into navigating the complex terrain of web application development with React.

3.2 Critical Review of Literature

In the realm of machine learning, TensorFlow emerges as a groundbreaking open-source framework developed by Google, enabling large-scale machine learning endeavors. The evolution of TensorFlow Serving is discussed in subsequent papers, highlighting its flexibility, high performance, and adaptability for industry-scale machine learning deployment. Shifting focus to machine learning datasets, TensorFlow Datasets is introduced as a transformative tool for simplifying the complexities of building machine learning workloads. The collaborative and inclusive nature of TensorFlow Datasets is emphasized, showcasing a community-driven initiative that provides a streamlined approach for accessing and managing diverse datasets.

Moving into web development, FastAPI is presented as a Python web framework that stands out for its rapid development and runtime efficiency. React.js takes center stage in the exploration of modern web development, emphasizing its declarative and component-based nature in influencing UI development. React's functional underpinnings are explored, showcasing its evolution into a powerful framework for creating user interfaces. The journey of React takes a turn with the implementation of server-side rendering,

optimizing web development, and enhancing the performance and user experience of applications. Collectively, these papers contribute to advancing our understanding of machine learning frameworks, datasets, and web development technologies, showcasing their pivotal roles in shaping the landscape of large-scale machine learning and modern web development.

The existing literature on TensorFlow and related technologies demonstrates notable strengths, particularly in providing comprehensive insights into the evolution and capabilities of TensorFlow as a leading open-source machine learning framework. The papers collectively offer a well-documented journey from TensorFlow's inception to its role in industry-scale deployment, shedding light on its core design principles, features, and optimizations. The exploration of TensorFlow Serving further adds to the strengths by addressing the challenges of deploying machine learning models in production environments. Additionally, the discussions on TensorFlow Datasets contribute significantly to the field by highlighting the tool's versatility and the collaborative community spirit, providing a valuable resource for researchers and practitioners alike.

However, certain weaknesses and gaps in the literature are evident. While the papers extensively cover TensorFlow, there is a limited exploration of alternative machine learning frameworks or comparative analyses, which could offer a more holistic view of the landscape. Additionally, the literature predominantly focuses on the strengths and advancements of the technologies discussed, leaving room for more critical discussions on potential limitations, challenges, and areas of improvement. Furthermore, while the papers on web development technologies like FastAPI and React.js provide valuable insights, there is a lack of integration between the machine learning and web development perspectives, potentially limiting a comprehensive understanding of how these technologies can synergize for holistic applications. Future research could bridge these gaps by exploring interdisciplinary approaches, conducting comparative analyses, and delving deeper into the challenges and limitations of these technologies to foster a more nuanced and comprehensive understanding.

Chapter 4

ACTUAL WORK

4.1 Methodology for the Study

4.1.0.1 Python Setup

- Install Python:

1. Follow the setup instructions to install Python on your system.
2. Download the latest version from (https://www.python.org/downloads/) and make sure to check the option to add Python to the system PATH during installation.

- Install Python Packages

1. Open a command prompt and navigate to the project's ‘training‘ directory.
2. Run the following commands to install the required Python packages:

“‘bash pip install -r training/requirements.txt pip install -r api/requirements.txt “‘

4.1.0.2 Install TensorFlow Serving

1. Follow the setup instructions (https://www.tensorflow.org/tfx/guide/serving) to install TensorFlow Serving.
2. TensorFlow Serving is more commonly used on Linux, and using it on Windows might involve different steps or limitations.

4.1.0.3 ReactJS Setup

- Install Node.js and NPM

1. Download and install Node.js and NPM from (https://nodejs.org/).
2. Make sure to check the option to add Node.js to the system PATH during installation.

4.1.0.4 Install Dependencies for Frontend

1. Open a command prompt and navigate to the ‘frontend‘ directory.
2. Run the following commands:

“‘bash cd frontend npm install –from-lock-json npm audit fix “‘

- Copy Environment Variables

1. Copy the provided ‘.env.example‘ file in the ‘frontend‘ directory and rename it to ‘.env‘.
2. Update the API URL in the ‘.env‘ file to match your desired configuration.

- React-Native Setup

1. Follow the React Native CLI Quickstart instructions (https://reactnative.dev/docs/environment-setup).
2. Choose the React Native CLI Quickstart tab.

4.1.0.5 Model Training

- Download Dataset

Download the required dataset from Kaggle. Keep only the folders related to potatoes.

- Run Jupyter Notebook

1. Open a command prompt and run the following command:
2. “‘bash jupyter notebook “‘
3. In Jupyter Notebook, open ‘training/potato-disease-training.ipynb‘.
4. Update the dataset path in cell 2 and execute all cells one by one.
5. Save the generated model with the version number in the ‘models‘ folder.

- API Execution

1. FastAPI
2. Start FastAPI Server
3. Open a command prompt and navigate to the ‘api‘ folder.
4. Run the following command:

“‘bash cd api uvicorn main:app –reload –host 0.0.0.0 “‘ The API is now running at ‘0.0.0.0:8000‘.

- FastAPI and TF Serve

1. Configure and Run TF Serve
2. Inside the ‘api‘ folder, copy ‘models.config.example‘ as ‘models.config‘ and update the paths in the file.
3. Run TensorFlow Serving using Docker:

“‘bash docker run -t –rm -p 8501:8501 -v /path/to/potato-disease-classification:/potato-disease-classification tensorflow/serving –rest_api_port=8501 –model_config_file=/potato-disease-classification/models.config “‘

- Run the FastAPI server using:

“‘bash uvicorn main-tf-serving:app –reload –host 0.0.0.0 “‘ The API is now running at ‘0.0.0.0:8000‘.

- Frontend Execution

1. Configure Frontend
2. - Inside the ‘frontend‘ folder, copy ‘.env.example‘ as ‘.env‘ and update the ‘REACT_APP_API_URL‘ to the API URL if needed.
3. Run Frontend
4. - Open a command prompt and navigate to the ‘frontend‘ directory.
5. - Run the frontend using:

“‘bash npm run start “‘

Now, you have successfully set up and executed the entire project on a Windows PC, including Python, ReactJS, and React-Native components.

4.2 Experimental and/or Analytical Work Completed in the Project

4.2.0.1 Choosing Right Platform

For developing a potato leaf blight disease detection CNN model due to its robust support for deep learning tasks and image analysis Tensorflow is ideal. TensorFlow's comprehensive ecosystem provides a seamless integration of high-level APIs like Keras, facilitating the efficient design and training of convolutional neural networks (CNNs). Its flexibility allows for easy experimentation with different CNN architectures, optimizing the model's ability to capture intricate patterns in potato leaf images.

The library's strong compatibility with GPUs ensures accelerated training, enhancing the CNN model's computational efficiency. TensorFlow's pre-trained models and extensive documentation further expedite the development process, providing valuable resources for building an accurate and effective disease detection system. Additionally, TensorFlow's commitment to community collaboration ensures ongoing support, updates, and advancements, contributing to the model's reliability and sustainability in the context of potato leaf blight detection.

One of the key advantages of TensorFlow is its seamless integration with Python, a widely adopted and user-friendly programming language. This integration allows developers to leverage the simplicity and readability of Python while harnessing the computational power and efficiency of TensorFlow. The extensive use of Python in the data science and machine learning communities has contributed to TensorFlow's widespread adoption and accessibility. TensorFlow's architecture is designed to support both deep learning and traditional machine learning tasks, making it a versatile tool for a wide range of applications. Its underlying computational graph abstraction enables efficient execution of complex mathematical operations, making it well-suited for handling the computational demands of deep neural networks. TensorFlow's flexibility allows users to implement custom models or use pre-built high-level APIs like Keras, facilitating rapid prototyping and experimentation.

In image analysis, TensorFlow stands out as a go-to library due to its robust support for convolutional neural networks (CNNs), a class of deep learning models particularly effective in processing visual data. CNNs excel at learning hierarchical features from

images, capturing intricate patterns and representations that contribute to accurate image analysis. TensorFlow's dedicated support for GPU acceleration further enhances the efficiency of training deep neural networks, making it suitable for handling large-scale image datasets. Moreover, TensorFlow provides pre-trained models for image classification and object detection through its TensorFlow Hub and TensorFlow Lite platforms. These pre-trained models, built on extensive datasets, offer a valuable resource for developers looking to leverage state-of-the-art image analysis capabilities without starting from scratch. TensorFlow's commitment to community collaboration and ongoing development ensures a vibrant ecosystem with continuous updates and improvements, solidifying its position as a leading library in the realm of machine learning and image analysis.

4.2.0.2 TensorFlow Dataset (TF Dataset):

TensorFlow Dataset, often referred to as 'tf.data', is a module within TensorFlow designed to streamline and optimize the process of input data pipeline construction for machine learning models. It provides a collection of tools to efficiently load, preprocess, and iterate over datasets during the training and evaluation of machine learning models. TF Dataset is particularly powerful for handling large-scale datasets and ensuring that the input pipeline does not become a bottleneck in model training.

The key components of TF Dataset include 'tf.data.Dataset' objects, which represent sequences of data elements, and a variety of transformation methods to preprocess and manipulate these datasets. This module allows for parallel processing of data, prefetching, and efficient batching, enabling developers to build high-performance data pipelines for training machine learning models with TensorFlow.

4.2.0.3 TensorFlow Serving:

TensorFlow Serving is an open-source serving system for deploying machine learning models in production environments. It simplifies the process of serving and managing models, allowing them to be easily integrated into scalable and efficient serving infrastructures. TensorFlow Serving supports the deployment of models trained using TensorFlow and provides a flexible and extensible platform for serving models over APIs.

With TensorFlow Serving, machine learning models can be deployed as microservices, making them accessible for real-time predictions and inference. The serving system is designed to handle multiple versions of models concurrently, facilitating seamless model updates and rollbacks without interrupting service. TensorFlow Serving is widely used in production scenarios, providing a robust solution for serving machine learning models at scale.

4.2.0.4 FastAPI:

FastAPI is a modern, fast (as the name suggests), web framework for building APIs with Python. It is designed to be easy to use, highly performant, and capable of automatically generating interactive API documentation. FastAPI is built on top of Starlette and Pydantic, leveraging asynchronous programming to achieve high concurrency and efficiency.

One of the key features of FastAPI is its automatic data validation and serialization using Pydantic models, which not only enhances the reliability of APIs but also reduces development time. FastAPI is particularly well-suited for machine learning model deployment as it supports asynchronous handling, making it compatible with TensorFlow Serving or other asynchronous serving systems. Its simplicity, performance, and automatic documentation generation make FastAPI a popular choice for building APIs in Python, especially when serving machine learning models or creating web applications with API backends.

4.2.0.5 Choosing React JS

React JS is considered an easy and efficient way to create a frontend for an AI model due to its declarative and component-based architecture. React simplifies the development process by breaking down the user interface into modular components, each responsible for a specific part of the UI. This component-based approach fosters code reusability, maintainability, and scalability. Developers can build individual components that encapsulate specific functionalities or features of the AI model, making it easier to manage and understand the codebase.

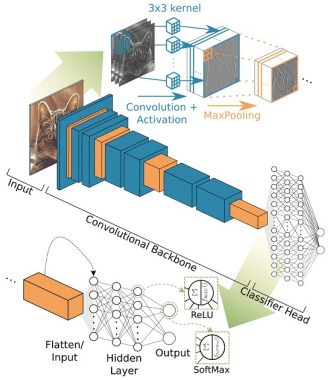
Moreover, React's declarative nature allows developers to describe the desired state of the UI, and React takes care of efficiently updating the DOM to reflect those changes.

This makes it intuitive for developers to design dynamic and interactive interfaces without the need to manually manipulate the DOM, reducing the likelihood of errors. When integrating an AI model into a frontend, React facilitates the seamless incorporation of interactive elements, data visualizations, and real-time updates, providing a smooth and responsive user experience. The vibrant React community and the availability of numerous libraries and tools further contribute to the ease of development, making React an attractive choice for building frontends that showcase the capabilities of AI models.

4.3 Modeling, Analysis & Design

4.3.0.1 CNN Modelling

Convolutional Neural Networks (CNNs) represent a specialized class of neural networks designed to excel in processing and analyzing visual data, making them particularly effective for image analysis tasks. Unlike traditional neural networks, CNNs leverage convolutional layers to automatically and adaptively learn hierarchical features from images. The use of convolutional operations allows the network to detect patterns such as edges, textures, and complex structures at different scales, enabling it to understand the hierarchical relationships within an image. CNNs are well-suited for image analysis due to their ability to capture spatial hierarchies and local patterns. By employing convolutional filters, the network can recognize features in different parts of an image, and as these filters are applied across the entire input, the network learns to extract increasingly abstract and complex representations. This hierarchical feature learning makes CNNs adept at tasks such as image classification, object detection, and segmentation, where understanding spatial relationships and identifying patterns are crucial.



In addition to their effectiveness in image analysis, CNNs also offer advantages in terms of reducing processing file size. The use of shared weights in convolutional layers significantly reduces the number of parameters compared to fully connected layers in traditional neural networks. This parameter sharing, coupled with the hierarchical feature learning, leads to more compact representations of visual information. As a result, CNNs can achieve high accuracy in image analysis tasks with fewer parameters, making them computationally more efficient and reducing the overall file size of the model. This efficiency is particularly valuable in scenarios with limited computational resources, such as deploying models on edge devices or mobile applications, where optimizing file size without compromising performance is essential.

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)
```

FIGURE 4.1: CNN Model Layers

4.3.0.2 Data Augmentation

Data augmentation is a crucial technique in machine learning, particularly in scenarios where limited labeled data is available for model training. It involves artificially expanding the training dataset by applying various transformations to the existing images, such as rotations, flips, zooms, and changes in brightness or contrast. The primary goal of data augmentation is to introduce diversity and variability into the training data, enabling the model to learn a more robust and generalized set of features. This increased diversity helps prevent overfitting, where a model becomes too specific to the training data and struggles to generalize well to new, unseen data.

In the context of image analysis, data augmentation is particularly beneficial. It simulates different real-world scenarios, capturing the variability in object appearance, lighting conditions, and perspectives. For instance, in the case of potato leaf blight disease detection, incorporating augmented images that mimic variations in leaf orientation, lighting, and disease manifestation stages ensures that the CNN model becomes adept at recognizing diseases under a range of conditions. Ultimately, data augmentation enhances the model's ability to generalize and improves its performance on diverse, real-world datasets, making it a critical component in achieving better analysis results.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0

 conv2d (Conv2D)             (32, 254, 254, 32)        896

 max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0
 )

 conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

 max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0
 2D)

 conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928

 max_pooling2d_2 (MaxPooling  (32, 30, 30, 64)         0
 2D)

 conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928

 max_pooling2d_3 (MaxPooling  (32, 14, 14, 64)         0
 2D)

 conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928

 max_pooling2d_4 (MaxPooling  (32, 6, 6, 64)           0
 2D)

 conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928

 max_pooling2d_5 (MaxPooling  (32, 2, 2, 64)           0
 2D)

 flatten (Flatten)           (32, 256)                 0

 dense (Dense)               (32, 64)                  16448

 dense_1 (Dense)             (32, 3)                   195

=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
```

FIGURE 4.2: Model summary

4.4 Implementation Details

4.4.0.1 Uvicorn , CORS and Fast API

Uvicorn is an ASGI (Asynchronous Server Gateway Interface) server implementation for Python web applications. Developed by the creators of the popular ASGI framework FastAPI, Uvicorn is designed to handle asynchronous HTTP requests efficiently. Its primary role is to serve web applications asynchronously, providing a high level of concurrency and responsiveness. Unlike traditional synchronous servers, Uvicorn

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255)
])
```

```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2),
])
```

```
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

FIGURE 4.3: Data augmentation

leverages Python's asynchronous programming features, making it well-suited for handling a large number of simultaneous connections with minimal resource consumption. Uvicorn is particularly popular for hosting web frameworks like FastAPI, Starlette, and others that support asynchronous patterns. It is known for its simplicity and ease of use, making it a preferred choice for developers working on modern web applications that leverage asynchronous programming paradigms. With features like automatic reload during development, support for HTTPS, and a straightforward command-line interface, Uvicorn streamlines the process of deploying and testing asynchronous web applications, contributing to its widespread adoption in the Python web development community.

1. **Asynchronous Handling**: Uvicorn excels in handling asynchronous operations, making it well-suited for scenarios where multiple operations can occur concurrently. With asynchronous handling, your program can efficiently manage tasks like handling multiple incoming POST requests simultaneously, ensuring a responsive and scalable solution.

2. **Concurrency and Performance:** Asynchronous servers like Uvicorn allow your program to handle a large number of connections concurrently without the need to spawn multiple threads or processes. This leads to better performance and resource utilization, especially when dealing with I/O-bound tasks like waiting for data from a POST request or interacting with databases.

```python
from fastapi.middleware.cors import CORSMiddleware
import uvicorn
import numpy as np
from io import BytesIO
from PIL import Image
import tensorflow as tf

# Creating a FastAPI application instance
app = FastAPI()

# Defining the allowed origins for CORS (Cross-Origin Resource Sharing)
origins = [
    "http://localhost",
    "http://localhost:3000",
]

# Adding CORS middleware to allow cross-origin requests
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

FIGURE 4.4: Python App

3. **Efficient Resource Usage:** Uvicorn's asynchronous nature enables your program to perform other tasks while waiting for I/O operations to complete. This minimizes the idle time, resulting in more efficient resource usage and the ability to handle a higher number of requests with the same resources.

4. Simplicity and Ease of Use: Uvicorn comes with a straightforward command-line interface, making it easy to integrate into your Python program. Its simplicity and compatibility with ASGI frameworks simplify the process of handling POST requests asynchronously, reducing development complexity.

5. **Scalability:** With Uvicorn, your program can easily scale to handle a growing number of concurrent requests. This scalability is particularly advantageous in scenarios where your application needs to handle a large volume of POST requests efficiently, such as in web APIs or microservices.

CORS middleware, which stands for Cross-Origin Resource Sharing middleware, is a mechanism implemented in web applications to control and manage access to resources from different origins or domains. In a web context, an "origin" refers to the combination of protocol (e.g., HTTP or HTTPS), domain, and port. Due to security restrictions imposed by web browsers, web pages served from one origin typically cannot make

direct requests to resources on another origin. CORS middleware helps relax these restrictions by allowing servers to specify which origins are permitted to access their resources. It adds specific HTTP headers to responses, indicating the set of origins that are allowed to access the server's resources. The key headers involved in CORS are:

1. Access-Control-Allow-Origin:Specifies which origin(s) are allowed to access the server's resources. It can be a specific origin, a comma-separated list of origins, or the special value "*" (wildcard), indicating that any origin is allowed.

2. Access-Control-Allow-Methods: Specifies the HTTP methods (e.g., GET, POST, PUT) that are permitted when accessing the resource.

3. Access-Control-Allow-Headers: Specifies which HTTP headers can be used when making the actual request.

4. Access-Control-Allow-Credentials:Indicates whether the browser should include credentials (like cookies or HTTP authentication) when making the actual request.

5. Access-Control-Expose-Headers:Specifies which response headers should be exposed to the browser when the actual request is made.

Implementing CORS middleware is crucial when developing web applications that involve cross-origin requests, such as when making AJAX requests from a frontend application served from one domain to a backend API hosted on another domain. It helps strike a balance between security and the need for web applications to interact with resources on different origins in a controlled manner.

4.4.0.2 React JS based Front-end Implementation

1. **Declarative UI Design:** React's declarative nature allows developers to easily express the user interface based on the application's state. This is particularly beneficial when working with machine learning models, as React enables intuitive handling of dynamic content and real-time updates that may result from interactions with the TensorFlow Serving API.

2. **Component-Based Architecture:** React's component-based architecture encourages modularization of the UI into reusable components. This is advantageous when integrating with TensorFlow Serving, allowing developers to encapsulate different parts of the user interface that interact with the model. Each component can manage its state and logic, promoting a clean and maintainable code structure.

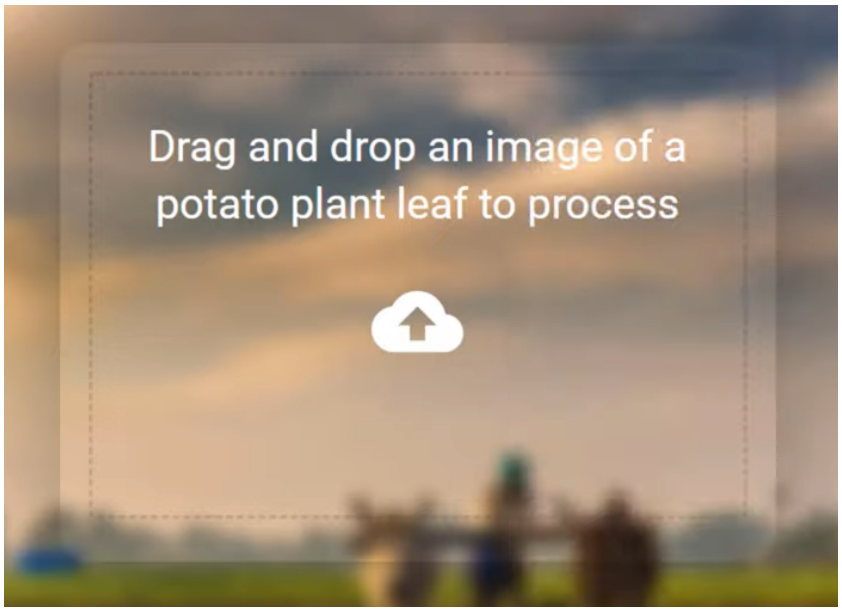


FIGURE 4.5: React Front-End

3. **Efficient Virtual DOM:** React utilizes a Virtual DOM, which is a lightweight copy of the actual DOM. This allows React to efficiently update only the parts of the UI that have changed, reducing the amount of work needed to keep the UI in sync with the application's state. This efficiency is crucial when working with dynamic data generated by the TensorFlow Serving Python model.

4. **Rich Ecosystem and Libraries:** React has a vast ecosystem of libraries and tools that can be leveraged for tasks such as data visualization, user interface animations, and state management. These resources can enhance the frontend's capabilities when displaying information from the TensorFlow model or handling user interactions.

5. **State Management**: React provides a straightforward way to manage application state, which is essential when dealing with the results or predictions from a TensorFlow Serving model. By efficiently managing state, React ensures that the frontend stays synchronized with the backend, providing a seamless user experience.

6. **Community Support and Documentation**: React boasts a large and active community, resulting in extensive documentation, tutorials, and community support. This wealth of resources can be valuable when integrating TensorFlow Serving with the frontend, ensuring that developers have access to assistance and best practices.

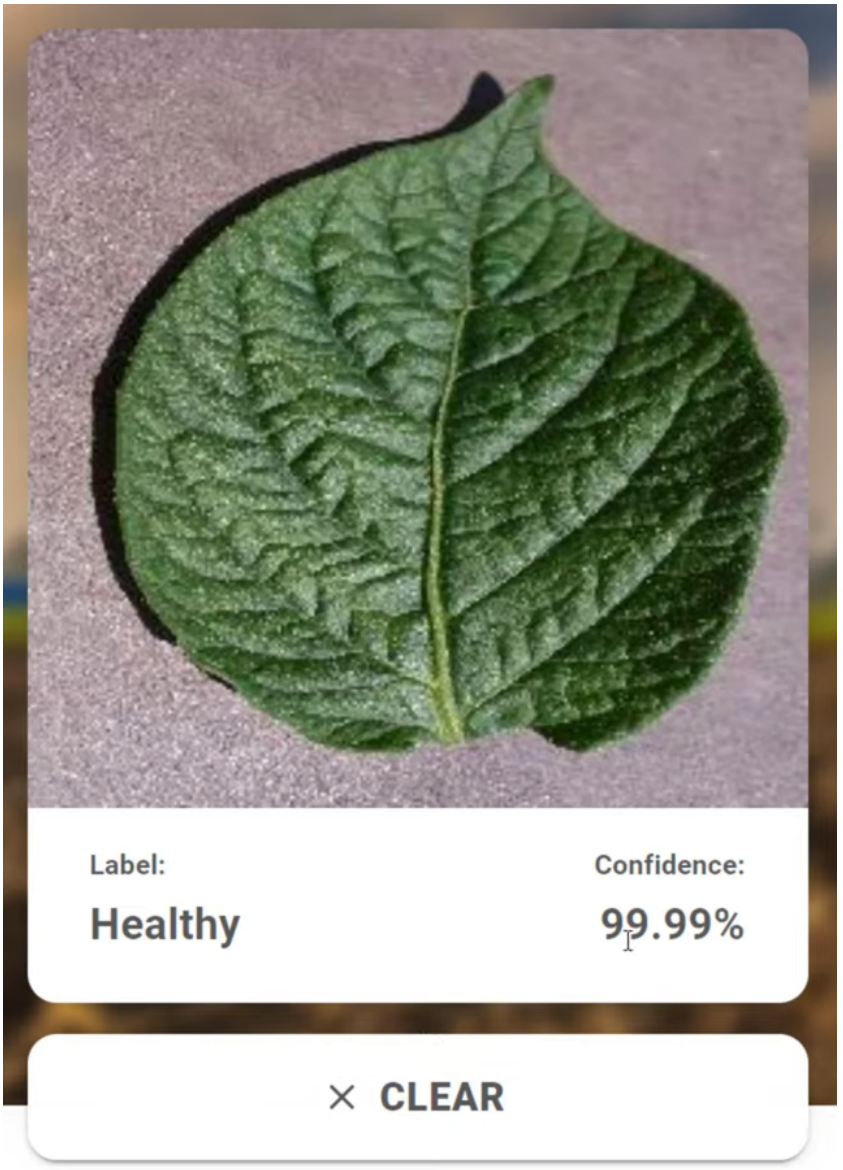


FIGURE 4.6: Final Result

Chapter 5

RESULTS, DISCUSSIONS AND CONCLUSIONS

5.1 Results & Analysis

The Convolutional Neural Network (CNN) model developed for predicting potato blight diseases has exhibited remarkable performance, achieving an average accuracy rate of 98-99%. This high accuracy underscores the model's proficiency in distinguishing between healthy, early blight, and late blight stages based on leaf images. The model was trained on a diverse dataset containing labeled images of potato leaves representing different disease stages, allowing it to learn intricate patterns and features indicative of each condition.

In the analysis of results, the high accuracy of the CNN model implies its efficacy in providing precise and reliable disease diagnosis for potato crops. Such accuracy is particularly beneficial for farmers in India, where potato cultivation is a vital component of agriculture. The model's ability to swiftly identify the presence and stage of blight diseases can empower farmers with timely and actionable insights. Early detection enables prompt intervention measures, such as targeted pesticide application or crop rotation, potentially mitigating the spread of diseases and minimizing yield losses.

The implementation of this technology in Indian agriculture carries significant implications for the future of the industry. With accurate disease diagnosis through AI-driven

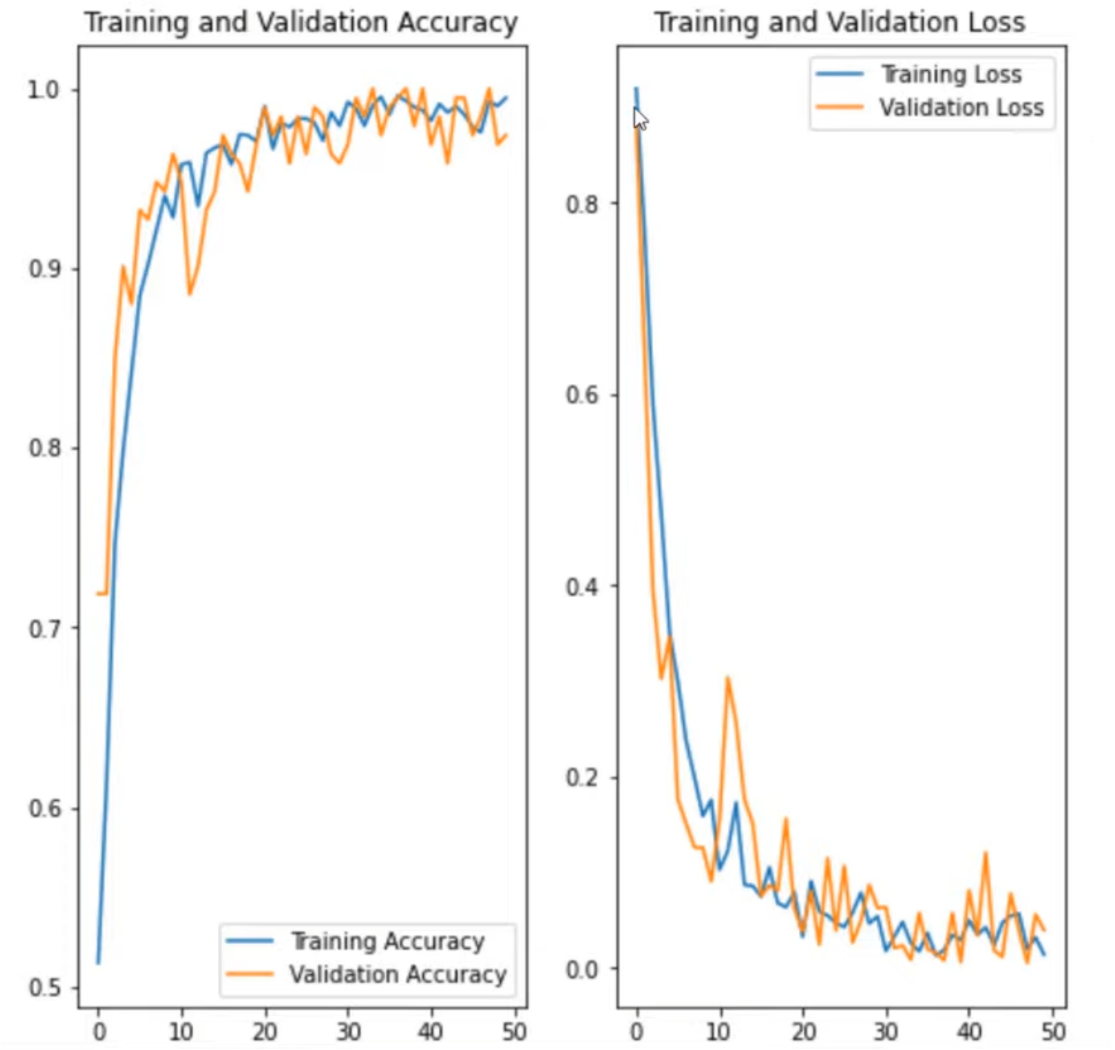


FIGURE 5.1: Training and Validation Accuracy

models, farmers can optimize resource usage, reduce the environmental impact of excessive pesticide application, and enhance overall crop health. Additionally, the integration of advanced technologies like CNN models can contribute to sustainable agriculture practices, aligning with global trends towards precision farming. As the technology becomes more accessible and widespread, it holds the potential to revolutionize disease management strategies, improve crop yield predictions, and foster a data-driven approach to agriculture in India. This not only benefits individual farmers but also contributes to the broader goal of food security and sustainable agricultural practices on a national scale.

```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
```

```
first image to predict
actual label: Potato___Early_blight
predicted label: Potato___Early_blight
```

FIGURE 5.2: Notebook Model Prediction

5.2 Comparative Study

In the absence of the predictive CNN model for potato blight diseases, farmers would face several challenges and limitations in disease detection and management. Without access to a reliable and efficient tool for early detection, farmers would be more reliant on traditional methods, visual inspections, and manual observations. This approach is not only time-consuming but also prone to human error, as distinguishing between different disease stages can be challenging, especially for less experienced farmers. The lack of a systematic and accurate diagnostic tool could result in delayed detection, allowing diseases to progress unchecked, leading to significant yield losses.

Relying solely on expert opinion for disease detection would also present hurdles, as accessing agricultural experts may be time-consuming and expensive. Experts are often limited in number, making their availability a bottleneck in timely disease diagnosis. This limitation is particularly impactful in regions with a high density of agricultural activities, where the demand for expert consultations may exceed their capacity. As a

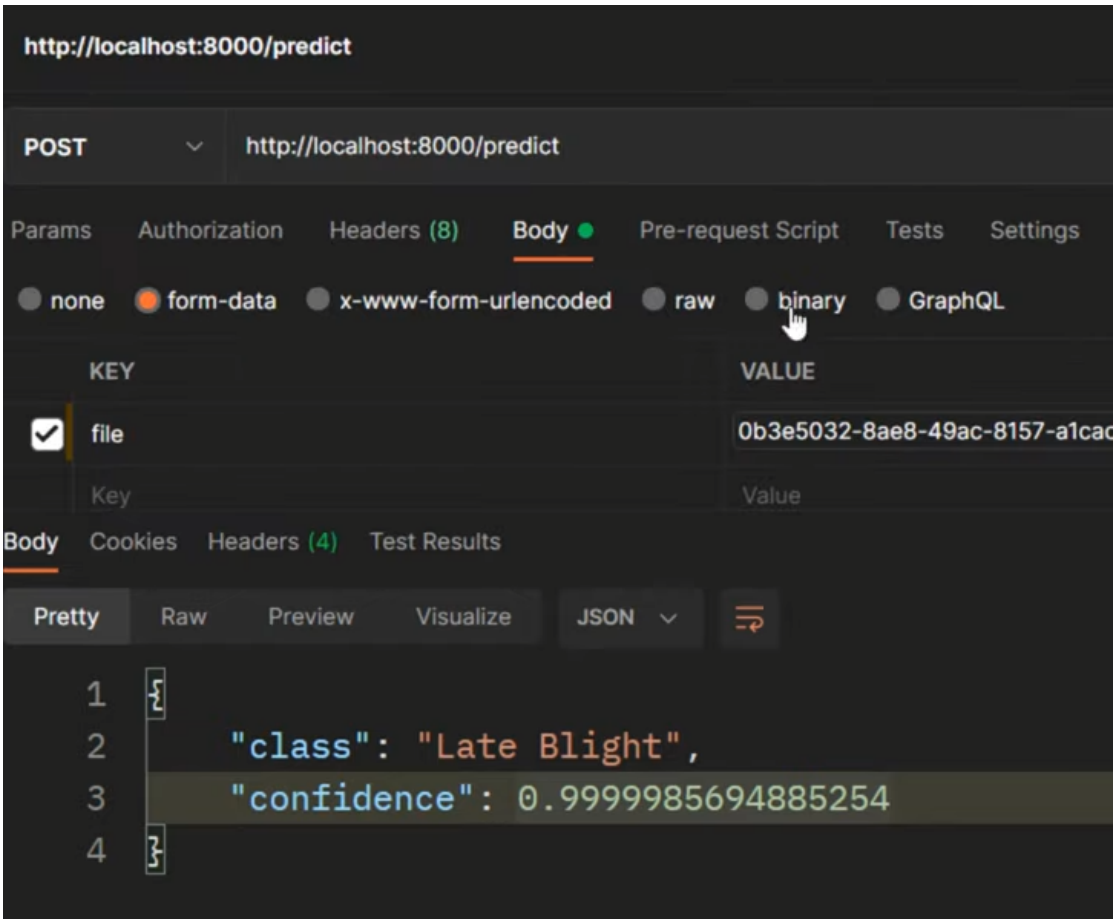


FIGURE 5.3: POSTMAN predition on GET request

consequence, farmers may experience delays in obtaining accurate diagnoses, hindering their ability to implement timely interventions and potentially leading to the spread of diseases across crops.

On the contrary, farmers equipped with the CNN prediction model experience a paradigm shift in disease management. The model's ability to provide early detection, often before visible symptoms manifest, empowers farmers to take swift and targeted action. The speed and efficiency of the model eliminate the need for constant reliance on experts, allowing farmers to independently make informed decisions based on accurate predictions. The mass analysis capabilities of the model enable quick assessments of entire fields, ensuring that potential disease outbreaks are promptly identified and addressed.

The ease of use of the CNN model further contributes to its practicality in the agricultural context. Farmers can easily integrate the technology into their existing workflows, capturing images of potato leaves using common devices such as smartphones or low-cost cameras. The high accuracy of predictions reduces false positives and negatives,

instilling confidence in the model's reliability. This transformative approach not only streamlines disease management but also empowers farmers with a proactive stance, contributing to enhanced crop health, increased productivity, and sustainable agricultural practices. Ultimately, the adoption of predictive models represents a significant leap forward in modernizing agriculture, bridging the gap between traditional practices and cutting-edge technologies for the benefit of farmers and the industry at large.

In conclusion, the stark contrast between the scenarios with and without access to the CNN prediction model underscores the transformative impact of advanced technologies on agriculture. Without such a tool, farmers face hurdles related to delayed disease detection, over-reliance on expert consultations, and increased susceptibility to yield losses. On the other hand, the adoption of the CNN model heralds a new era of precision agriculture, providing farmers with an unprecedented ability to proactively manage diseases. The model's early detection capabilities, mass analysis features, ease of use, and high accuracy collectively empower farmers to make timely decisions, optimize resource usage, and mitigate the economic impact of crop diseases.

As we navigate the future of agriculture, it becomes evident that the integration of predictive models into farming practices is not just a technological upgrade; it represents a fundamental shift in how we approach crop health and management. The accessibility of accurate predictions transforms farmers into informed decision-makers, reducing dependency on external expertise and paving the way for sustainable and resilient agricultural systems. Embracing technologies like the CNN model not only enhances productivity but also contributes to the broader goals of food security, resource efficiency, and the establishment of a more robust foundation for the agricultural sector in India and beyond. The comparative study highlights not only the immediate advantages but also the potential for long-term positive impacts that can revolutionize farming practices and shape the future of agriculture.

5.3 Cost Estimation Model

Developing and implementing a cost estimation model for deploying the CNN prediction system in agriculture involves considering various factors, including technology development, infrastructure, and ongoing operational costs. Here's a breakdown of the key components for creating a comprehensive cost estimate:

1. Technology Development: The initial phase involves the development of the CNN model, including data collection, model training, and validation. This phase encompasses costs related to acquiring labeled datasets, computational resources for training, and the expertise of machine learning engineers. Additionally, expenses associated with software development for integrating the model with the frontend using ReactJS and backend with FastAPI or other relevant frameworks should be considered.

2. Deployment Infrastructure: Costs related to deploying the system include setting up servers or cloud infrastructure for hosting the TensorFlow Serving API and FastAPI backend. This involves considerations for server maintenance, hosting fees, and potential costs associated with securing the infrastructure, such as implementing HTTPS for secure communication.

3. User Interface and Accessibility: The frontend development using ReactJS incurs costs for UI/UX design, development, and testing. Accessibility features for farmers using diverse devices, such as smartphones or low-cost cameras, should be factored into the budget. Integration with the TensorFlow model and FastAPI backend, along with testing across various devices and browsers, contributes to the overall cost.

4. Maintenance and Updates: Ongoing operational costs include maintaining the deployed infrastructure, monitoring system performance, and addressing potential issues. Regular updates to the CNN model based on new data and advancements in machine learning, as well as updates to the user interface for improved usability, contribute to the long-term costs. Ensuring the security of the system through regular audits and updates is crucial for sustained functionality.

5. Training and Support: To maximize the effectiveness of the system, budgeting for farmer training and ongoing support is essential. This includes conducting workshops or training sessions to educate farmers on using the system, interpreting results, and implementing recommended practices. Providing continuous support channels, such as helplines or online assistance, ensures that farmers can seamlessly integrate the technology into their daily routines.

Creating a detailed cost estimation model requires a thorough understanding of the specific requirements, scale of deployment, and the regional context in which the system will be implemented. As the technology evolves and user feedback is incorporated, periodic reassessments of costs may be necessary to optimize the system's efficiency and sustainability. A well-planned cost estimation model ensures that the deployment of the CNN prediction system aligns with budgetary constraints while maximizing the benefits for farmers and the agricultural sector.

5.4 Conclusions

In conclusion, the integration of a sophisticated Convolutional Neural Network (CNN) prediction model for potato blight diseases in agriculture presents a transformative leap forward in modernizing traditional farming practices. The model, boasting an exceptional accuracy rate of 98-99%, signifies a powerful tool for farmers in India and beyond, offering unprecedented capabilities in early disease detection, mass analysis, and independent decision-making. This technological advancement addresses critical challenges faced by farmers, such as delayed disease identification and over-reliance on expert consultations, by providing timely and accurate predictions.

The comparative study between scenarios with and without access to the CNN model underscores the profound impact of advanced technologies on the agricultural landscape. Without this predictive tool, farmers grapple with inefficiencies, increased vulnerability to yield losses, and resource-intensive reliance on external expertise. Conversely, farmers equipped with the CNN model experience a paradigm shift, gaining the ability to proactively manage crop health, optimize resource usage, and mitigate economic losses associated with diseases. The technology not only streamlines disease management but also empowers farmers with a proactive stance, contributing to enhanced crop health, increased productivity, and sustainable agricultural practices.

Looking ahead, the successful deployment of the CNN model holds promise for the future of agriculture, bridging the gap between traditional farming practices and cutting-edge technologies. The integration of precision agriculture through AI-driven models aligns with broader goals of food security, resource efficiency, and sustainable farming practices. As this transformative technology becomes more accessible and widespread, it has the potential to revolutionize disease management strategies, improve crop yield predictions, and contribute to a data-driven approach that shapes the future of agriculture, not just in India but globally.

5.5 Scope for Future Work

5.5.1 Expanding Model for Multiple Plant Species:

The future scope of the CNN prediction model extends beyond potatoes to encompass a wide range of plant species. By diversifying the model to cater to the unique characteristics of various plants, agriculture can benefit on a broader scale. This expansion involves collecting and annotating datasets specific to each plant, training the model to recognize diseases across multiple species, and optimizing its architecture for versatility. The ability to predict diseases in crops such as wheat, rice, or tomatoes could revolutionize farming practices, providing farmers with a comprehensive tool for crop health management and disease prevention on a diverse range of crops.

Moreover, a multi-plant CNN model aligns with the principles of precision agriculture, offering a holistic approach to crop monitoring. Farmers would gain the flexibility to address the distinct challenges posed by different plant species, making the technology more adaptable and relevant to the diverse agricultural landscapes worldwide. This expansion not only enhances the utility of the model but also underscores its potential to contribute significantly to global food security and sustainable farming practices.

5.5.2 Expanding Prediction to Different Types of Diseases:

Beyond its current focus on early, late, and healthy stages in potato plants, the CNN prediction model's future scope involves expanding its predictive capabilities to encompass a broader spectrum of plant diseases. Incorporating diseases caused by various pathogens, including fungi, bacteria, and viruses, allows the model to serve as a comprehensive diagnostic tool. The expansion involves extensive data collection and annotation for diverse diseases, training the model to identify unique symptoms and patterns associated with each ailment. This inclusive approach not only addresses the specific needs of different crops but also provides farmers with a versatile solution for disease detection and management.

By diversifying the types of diseases covered, the CNN model becomes a comprehensive agricultural health monitoring system, capable of detecting and predicting a myriad of potential threats. This expansion aligns with the dynamic nature of plant diseases and positions the model as a forward-looking solution for mitigating risks and ensuring crop

resilience. As agriculture faces evolving challenges, a more extensive disease prediction model contributes to a proactive and adaptive strategy, promoting sustainable farming practices and securing global food supplies.

5.5.3 Making a Front End Mobile App:

To enhance user accessibility and ease of adoption, the future scope of the CNN prediction model involves the development of a front-end mobile application. Transforming the user interface into a mobile app ensures that farmers can easily interact with the technology using common devices like smartphones or tablets. The mobile app would simplify the image capturing process, enabling farmers to effortlessly capture and upload leaf images for analysis. Additionally, the app could include user-friendly features such as real-time notifications, disease severity indicators, and personalized recommendations, making the technology more intuitive and actionable for farmers in diverse settings.

The development of a mobile app not only aligns with the increasing prevalence of mobile technologies in rural areas but also democratizes access to advanced agricultural tools. Farmers can carry the power of disease prediction in their pockets, facilitating on-the-go decision-making and promoting timely interventions. The user-friendly interface and mobility of the app contribute to the democratization of precision agriculture, bridging the technological divide and empowering farmers with actionable insights for improved crop health.

5.6 Conclusion:

The future trajectory of the CNN prediction model holds immense promise, extending its impact beyond potato plants to diverse crops, a broader spectrum of diseases, and a more user-friendly interface through a mobile app. As the model evolves, it becomes a pivotal asset in addressing global agricultural challenges, fostering sustainable farming practices, and securing food supplies. By embracing adaptability, inclusivity, and accessibility, the future scope of the CNN prediction model stands as a testament to the transformative potential of advanced technologies in shaping the future of agriculture.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Neural Information Processing Systems (NeurIPS)*, 2012.

[2] Y. Liu et al., "Data Augmentation for Object Detection via Progressive and Selective Instance-Switching," in *European Conference on Computer Vision (ECCV)*, 2019.

[3] T. DeVries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," in *arXiv preprint arXiv:1708.04552*, 2017.

[4] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, 2016.

[5] M. Al-Rfou et al., "TensorFlow Datasets: A Swiss Army Knife for Building Machine Learning Workloads," in *arXiv preprint arXiv:1803.09820*, 2018.

[6] M. Al-Rfou et al., "TensorFlow Serving: Towards Industry-Scale Machine Learning," in *Proceedings of the 2nd SysML Conference*, 2019.

[7] N. Van Hoey et al., "TensorFlow Serving: Flexible, High-Performance ML Serving," in *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 2017.

[8] J. Walke and S. Chan, "React: Declarative, Component-Based, Learn Once, Write Anywhere," in *Proceedings of the 5th International Workshop on Web Component Development*, 2013.

[9] S. B. Uvic, "FastAPI: Fast (to develop), Fast (to run)," in *Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering*, 2021.

Appendix A

Screenshots

In [1]:
```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from IPython.display import HTML
```

In [2]:
```
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS=3
EPOCHS=50
```

In [3]:
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage",
    shuffle=True,
    image_size = (IMAGE_SIZE, IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
```

```
Found 2152 files belonging to 3 classes.
```

In [4]:
```
class_names = dataset.class_names
class_names
```

Out[4]:
```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

In [5]:
```
len(dataset)
```

Out[5]:
```
68
```

In [6]:
```
plt.figure(figsize=(10,10))
for image_batch, label_batch in dataset.take(1):
    for i in range(12):
        ax=plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[label_batch[i]])
        plt.axis("off")
```

FIGURE A.1: Jupyter Notebook -1

```
In [7]: train_size = 0.8
        len(dataset)*train_size

Out[7]: 54.400000000000006

In [8]: train_ds= dataset.take(54)
        len(train_ds)

Out[8]: 54

In [9]: test_ds = dataset.skip(54)
        len(test_ds)

Out[9]: 14

In [10]: val_size=0.1
         len(dataset)*val_size

Out[10]: 6.800000000000001

In [11]: val_ds = test_ds.take(6)
         len(val_ds)

Out[11]: 6

In [12]: test_ds=test_ds.skip(6)
         len(test_ds)

Out[12]: 8

In [13]: def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True, shuffle_size=10000):

             ds_size=len(ds)

             if shuffle:
                 ds=ds.shuffle(shuffle_size, seed=12)

             train_size=int(train_split*ds_size)
             val_size=int(val_split*ds_size)

             train_ds=ds.take(train_size)

             val_ds=ds.skip(train_size).take(val_size)
             test_ds=ds.skip(train_size).skip(val_size)

             return train_ds, val_ds, test_ds
```

FIGURE A.2: Jupyter Notebook-2

```
train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
```

```
len(train_ds)
```

```
54
```

```
len(val_ds)
```

```
6
```

```
len(test_ds)
```

```
8
```

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255)
])
```

```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2),
])
```

```
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
```

FIGURE A.3: Jupyter Notebook -3

```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

```
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)
```

```
0.9375
Epoch 45/50
54/54 [==============================] - 47s 872ms/step - loss: 0.0272 - accuracy: 0.9925 - val_loss: 0.1634 - val_accuracy:
0.9427
Epoch 46/50
54/54 [==============================] - 48s 892ms/step - loss: 0.0213 - accuracy: 0.9919 - val_loss: 0.4836 - val_accuracy:
0.8958
Epoch 47/50
54/54 [==============================] - 48s 886ms/step - loss: 0.0280 - accuracy: 0.9896 - val_loss: 0.2606 - val_accuracy:
0.9219
Epoch 48/50
54/54 [==============================] - 47s 863ms/step - loss: 0.0119 - accuracy: 0.9971 - val_loss: 0.0223 - val_accuracy:
0.9896
Epoch 49/50
54/54 [==============================] - 47s 868ms/step - loss: 0.0213 - accuracy: 0.9948 - val_loss: 0.0142 - val_accuracy:
0.9948
Epoch 50/50
54/54 [==============================] - 49s 901ms/step - loss: 0.0699 - accuracy: 0.9780 - val_loss: 0.1219 - val_accuracy:
0.9531
```

FIGURE A.4: Jupyter Notebook - 4

```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
```

```
first image to predict
actual label: Potato___Early_blight
1/1 [==============================] - 0s 295ms/step
predicted label: Potato___Early_blight
```

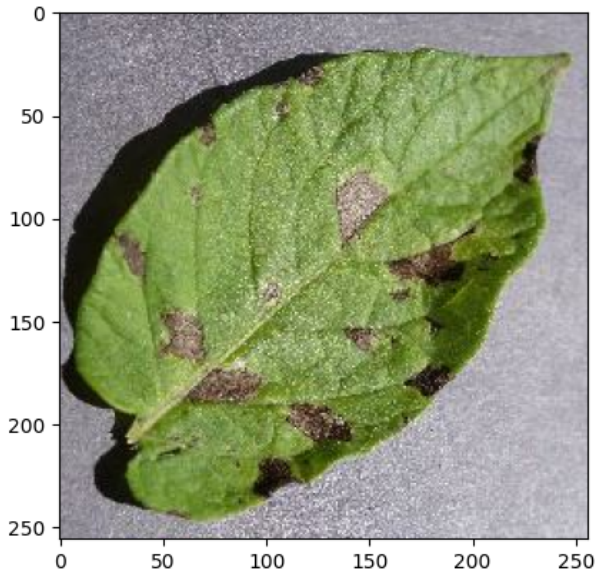


FIGURE A.5: Jupyter Notebook -5

```python
# Importing necessary modules and frameworks
from fastapi import FastAPI, File, UploadFile
from fastapi.middleware.cors import CORSMiddleware
import uvicorn
import numpy as np
from io import BytesIO
from PIL import Image
import tensorflow as tf

# Creating a FastAPI application instance
app = FastAPI()

# Defining the allowed origins for CORS (Cross-Origin Resource Sharing)
origins = [
    "http://localhost",
    "http://localhost:3000",
]

# Adding CORS middleware to allow cross-origin requests
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Loading the pre-trained TensorFlow model
MODEL = tf.keras.models.load_model("../saved_models/1")

# Defining class names for different disease categories
CLASS_NAMES = ["Early Blight", "Late Blight", "Healthy"]


# Endpoint for a simple ping to check if the application is alive
@app.get("/ping")
async def ping():
    return "Hello, I am alive"
```

FIGURE A.6: Jupyter Notebook - 6

```python
# Function to read the uploaded file as an image
1 usage
def read_file_as_image(data) -> np.ndarray:
    image = np.array(Image.open(BytesIO(data)))
    return image


# Endpoint for predicting the disease class based on the uploaded image
@app.post("/predict")
async def predict(
        file: UploadFile = File(...)
):
    # Reading the uploaded file as an image
    image = read_file_as_image(await file.read())
    img_batch = np.expand_dims(image, axis: 0)

    # Making predictions using the loaded model
    predictions = MODEL.predict(img_batch)

    # Extracting the predicted class and confidence level
    predicted_class = CLASS_NAMES[np.argmax(predictions[0])]
    confidence = np.max(predictions[0])

    # Returning the prediction result
    return {
        'class': predicted_class,
        'confidence': float(confidence)
    }


# Running the FastAPI application using uvicorn server
if __name__ == "__main__":
    uvicorn.run(app, host='localhost', port=8000)
```

FIGURE A.7: Jupyter Notebook -7

Appendix B

Results

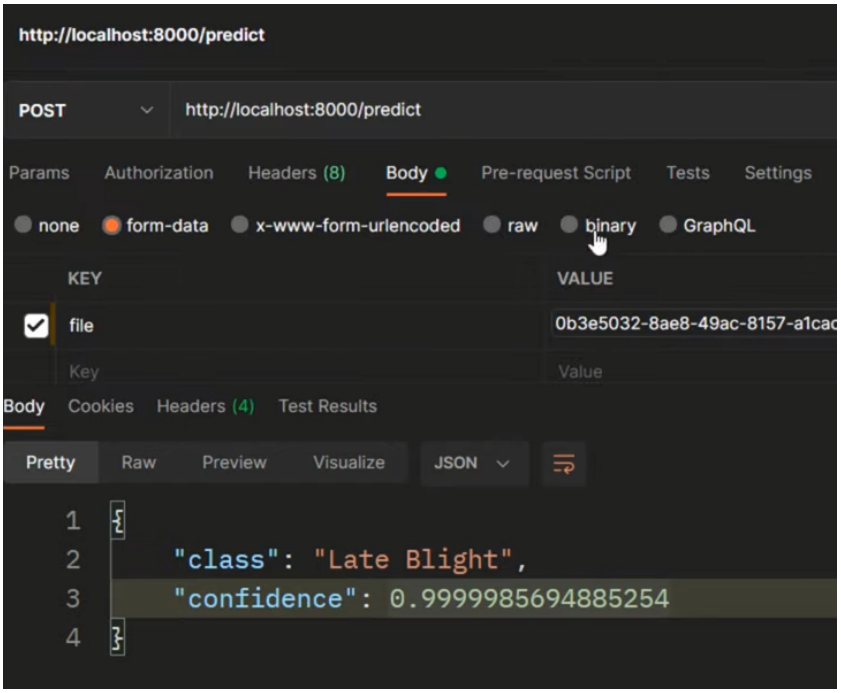


FIGURE B.1: POSTMAN GET request

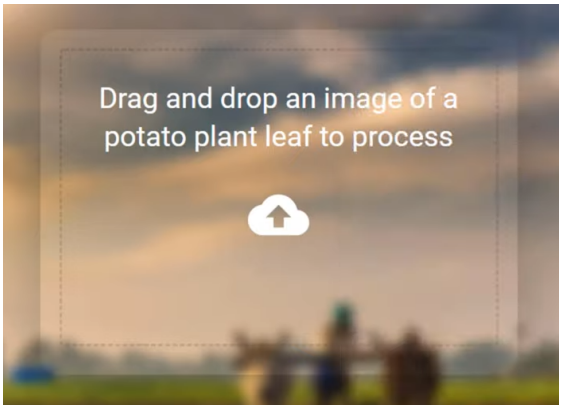


FIGURE B.2: Front End

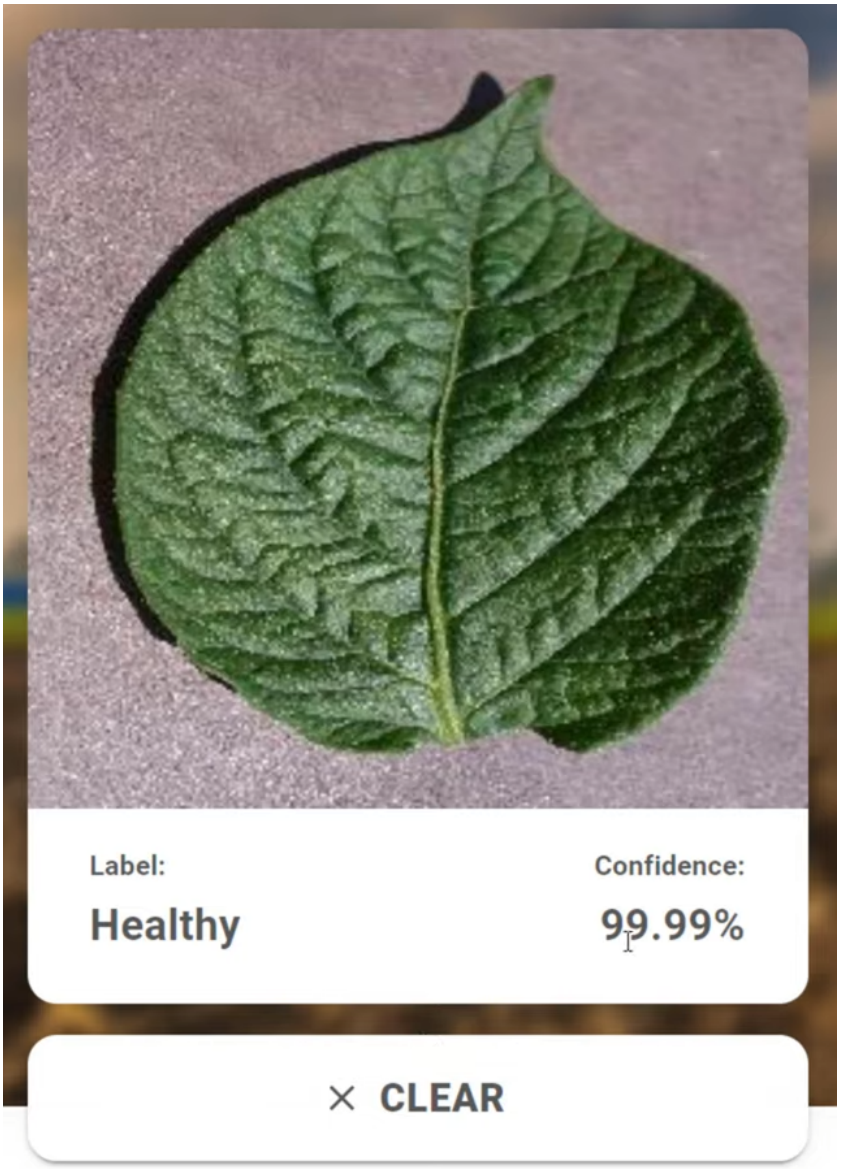


FIGURE B.3: Final Result