**POTATO LEAF DISEASE DETECTION USING DEEP LEARNING**

**A PROJECT REPORT**

*Submitted by*

**G Adithya [Reg No: RA2011027010019]**

**M. Lalith Kiran [Reg No: RA2011027010052]**

*Under the Guidance of*

**Dr. A. V. Kalpana**

(Assistant Professor, Department of Data Science and Business Systems)

*In partial fulfillment of the Requirements for the Degree*

*of*

**BACHELOR OF TECHNOLOGY**

## COMPUTER SCIENCE ENGINEERING

## with specialization in Big Data Analytics



**DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR – 603203**

**(Under Section 3 of UGC Act, 1956)**

SRM NAGAR, KATTANKULTATHUR – 603203

CHENGALPATTU DISTRICT

**NOVEMBER 2023**



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report titled “**POTATO LEAF DISEASE DETECTION USING DEEP LEARNING”** is the bonafide work of **“G. Adithya [Reg No: RA2011027010019] and M. Lalith Kiran [Reg No: RA2011027010052]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. A. V. Kalpana Dr. Lakshmi

**GUIDE HEAD OF THE DEPARTMENT**

Assistant Professor Dept. of DSBS

Dept. of DSBS

Signature of Internal Examiner Signature of External Examine

### Department of Data Science and Business Systems

**SRM Institute of science and technology**

**Own Work Declaration Form**

##### **Degree/ Course :**

**Student Name :**

##### **Registration Number :**

**Title of Work :**

We hereby attest that the evaluation complies with the requirements set out by the Education Committee, the University Website, and the Rules and Regulations against academic misconduct and plagiarism\*\*.

We certify that, with the exception of the places mentioned, everything of the work in this evaluation is original to me or us and that we have complied with the requirements listed below:

* Clearly cited all relevant sources and listed them all
* All quoted content (from books, the internet, etc.) was referenced and inserted inverted commas.
* Identified the origins of any images, data, etc. that I do not own
* Never used any of the previous or current reports or essays written by any other student.
* Acknowledged assistance from others (such as fellow students, technologists, statisticians, and outside sources) where appropriate.
* Compiling any additional standards for plagiarism listed on the university website or course handbook.

I am aware that any fraudulent claim made for this work will be punished in line with the rules and norms of the university.

| **DECLARATION:** |
| --- |
| I certify that, with the exception of the places where referencing is given, this evaluation is my or our own work and that I have adhered to the above-mentioned good academic practices. I am also aware of and understand the university's policy on academic misconduct and plagiarism. |
| If you are working in a group, please write your registration numbers and sign with the date for every student in your group. |

ii

**ACKNOWLEDGEMENT**

We sincerely appreciate **Dr. C. Miltonizihchevgan**, our esteemed Vice Chancellor, for serving as our guiding light in all of our undertakings.

We would like to thank **Dr. S. Ponnusamy**, our registrar, from the bottom of our hearts for his support.

We would like to sincerely thank **Dr. T. V. Gopal**, our dean of the College of Engineering and Technology, for bringing innovation to every execution.

We would like to sincerely thank **Dr. Revathi Venkataraman**, the Chairperson of the School of Computing, for giving us the courage to finish our course projects.

We would like to thank the course coordinators and professor **Dr.M.Lakshmi ,** head of the department of Data science and business systems, for their unwavering encouragement and support.

We are very appreciative of the help, timely suggestions, and direction provided by our course project instructor, **Mrs. A.V. Kalpana**, Assistant Professor, Department of DSBS, during the project.

We would like to express our appreciation to the Department of DSBS, our HoD, Professor **Dr. M. Lakshmi**, and my departmental colleagues for their support.

Finally, we would like to express our gratitude to our parents and close friends for their direct and indirect contributions to the accomplishment of our project. Above all, I am grateful to God for granting me the ability to finish my course assignment.

G. Adithya

M. Lalith kiran

**ABSTRACT**

Potato cultivation faces significant challenges due to various leaf diseases that adversely affect crop yield and quality. This study presents a comprehensive approach for potato leaf disease Classification utilizing multiple deep-learning architectures. The project encompasses three distinct models: a Convolutional Neural Network (CNN), a Long Short-Term Memory network (LSTM), and a Single Convolutional Neural Network (SCNN). Each model is tailored to address specific aspects of disease detection, providing a multi-modal framework for accurate and reliable results. The CNN model is designed to classify potato leaf diseases based on RGB images. The architecture incorporates Fully connected layers are used for classification, whereas convolutional layers are used for feature extraction. The model is trained on a diverse dataset containing images of healthy leaves and those affected by early blight, late blight, and other diseases. For temporal analysis, we introduce an LSTM model that processes sequences of images to capture the dynamic progression of diseases over time. The LSTM model is trained on sequences of potato leaf images, providing insights into the temporal patterns of disease development. Additionally, we propose a SCNN model, focusing on spatial features for improved disease detection. This model utilizes convolutional layers with batch normalization for effective feature extraction and classification. A collection of pictures of both healthy and sick potato leaves is used to train and assess the models. For every model, performance measures are provided, such as accuracy, precision, recall, and F1 score. Furthermore, the CNN and SCNN models are compared with traditional image processing methods, demonstrating the superiority of deep learning approaches in potato leaf disease detection. To enhance practical utility, the CNN model is deployed as a user-friendly application for real-time disease prediction on single-leaf images. The application allows farmers to upload images for instant diagnosis and provides visualizations of disease probabilities along with an interpretable confusion matrix. This all-encompassing strategy advances automated identification of potato leaf diseases and provides farmers with a reliable way to track and successfully manage crop health. The integration of multiple deep learning architectures provides a versatile framework capable of addressing various challenges associated with potato leaf diseases.

**TABLE OF CONTENTS**

|  |  |  | TABLE OF CONTENTS |  |
| --- | --- | --- | --- | --- |
| CHAPTER NO. | | | TITLE | PAGE NO. |
|  | | |  |  |
|  | ABSTRACT | |  | iv |
|  | LIST OF FIGURES | |  | vii |
|  | LIST OF SYMBOLS, ABBREVIATIONS | | | viii |
| 1. | INTRODUCTION | |  | 1 |
|  | 1.1 | GENERAL |  | 1 |
|  | 1.2 | PURPOSE | | 1 |
|  | 1.2.1 | SCOPE |  | 2 |
|  | 1.2.2 | NEED FOR POTATO LEAF DISEASE DETECTION | | 3 |
|  | 1.3 | SUSTAINABLE TECHNOLOGY | | 4 |
|  | 1.4 | MOTIVATION | | 5 |
|  | 1.5 | PROBLEM STATEMENT | | 7 |
|  | 1.5 | RESEARCH OBJECTIVES | | 8 |
|  | 1.6 | POTATO LEAF DISEASE DETECTION USING TENSORFLOW AND DEEP LEARNING TECHNIQUES | | 9 |
| 2 | LITERATURE REVIEW | |  | 11 |
|  | 2.1 | REVIEW PAPERS ONPOTATO LEAF DISEASE DETECTION | | 11 |
| 3 | PROPOSED METHODOLOGY | | | 17 |
|  | 3.1 CONVOLUTION NEURAL NETWORK | | | 17 |
|  | 3.1.1 INTRODUCTION TO CONVOLUTION NEURAL NETWORK | | | 17 |
|  | 3.1.2 LAYERS IN CNN | | | 18 |
|  | 3.1.3 FEATURE EXTRACTION WITH CONVOLUTION LAYERS | | | 19 |
|  | 3.1.4 OPTIMIZATION TECNIQUES FOR CNN | | | 19 |
|  | 3.1.5 ADVANCEMENTS IN CNN | | | 20 |
|  | 3.2 LONG SHORT-TERM MEMORY | | | 21 |
|  | 3.2.1 INTRODUCTION TO LONG SHORT-TERM MEMORY | | | 21 |
|  | 3.2.2 LAYERS IN LONG SHORT-TERM MEMORY | | | 22 |
|  | 3.2.3 FEATURE EXTRACTION WITH LSTM | | | 23 |
|  | 3.2.4 OPTIMIZATION TECNIQUES FOR LSTM | | | 24 |
|  | 3.2.5 ADVANCEMENTS IN LSTM | | | 26 |
|  | 3.3 SPATIAL CONVOLUTIONAL NEURAL NETWORK  v | | | 27 |
|  | 3.3.1 INTRODUCTION TO SPATIAL CONVOLUTION NEURAL NETWORK | | | 27 |
|  | 3.3.2 LAYERS IN SCNN | | | 28 |
|  | 3.3.3 FEATURE EXTRACTION WITH SCNN | | | 29 |
|  | 3.3.4 OPTIMIZATION TECHNIQUES FOR SCNN | | | 30 |
|  | 3.3.5 ADVANCEMENTS IN SCNN | | | 31 |
| 4 | MODULE/ EMPIRICAL STUDY 10 | | | 33 |
|  | 4.1 PRE-PROCESSING TECHNIQUES | | | 33 |
|  | 4.2 SEGMENTATION | | | 33 |
|  | 4.3 FEATURE EXTRACTION | | | 34 |
|  | 4.4 FEATURE SELECTION | | | 34 |
|  | 4.5 EVALUATION | | | 35 |
|  | 4.6 FINAL PREDICTION | | | 35 |
| 5 | CONCLUSION | | | 37 |
| 6 | FUTURE ENHANCEMENTS | | | 38 |
| 7 | REFERENCES | | | 39 |
|  | APPENDIX | | | 41 |
|  | PAPER PUBLICATION STATUS | | | 68 |
|  |  |  | |  |

**LIST OF FIGURES**

3.1 Architecture Diagram for CNN…………………………………………...17

3.2 Architecture Diagram for LSTM…………………………………………...22

3.3 Architecture Diagram for SCNN…………………………………………...28

**ABBREVIATIONS**

**AI** Artificial Intelligence

**CNN**  Convolution neural network

**SCNN** Single Convolutional Neural Network

**LSTM** Long Short-Term Memory network

**LIST OF SYMBOLS**

^ Conjunction

**CHAPTER 1**

**INTRODUCTION**

**1.1 General:**

Plant diseases can cause large financial losses as well as possible food shortages by seriously threatening agricultural output and quality. To limit the damage, prompt detection and diagnosis are crucial. Although visual examination is a classic procedure, it is subjective, error-prone, and time-consuming.

Deep learning techniques-based automated systems for plant disease identification have garnered significant attention in recent years. Using images of afflicted leaves, Deep learning models such as convolutional neural networks (CNNs), have demonstrated potential in the identification and diagnosis of plant diseases. These models use large datasets for training in order to identify patterns and traits linked to certain diseases.

If a disease is identified, treatment recommendations can be made according to the kind and degree of the illness. In order to provide a CNN-based system for the detection and classification of common plant illnesses from leaf photos, our study investigates the application of several Deep learning techniques for the diagnosis of plant leaf diseases. Furthermore, we present a system that, by referencing scientific literature and expert expertise, recommends suitable corrective actions for every condition identified.

With this strategy, farmers are able to respond quickly and effectively, preventing the spread of disease and minimising crop losses. Our technique offers major improvements in plant disease detection in terms of efficiency and accuracy.

**1.2 PURPOSE**

The main objective of the potato leaf disease detection project is to solve important issues with agricultural practises, particularly with regard to potato farming. The main goal is to create a reliable, cutting-edge system that can precisely recognise and categorise diseases that impact potato plants using visual clues found in leaf photos. This project is significant because it has the potential to transform plant health monitoring and give farmers a proactive tool for early disease detection.

The objective of the research is to increase the efficiency and accuracy of sickness diagnosis by utilizing cutting-edge deep learning techniques, especially Convolutional Neural Networks (CNN). In order to minimise crop losses and reduce the spread of infections, early detection is essential for limiting the impact of illnesses on potato crops. Food security and higher agricultural output follow from this. Beyond short-term profits, the project aims to have a revolutionary effect on agricultural sustainability. Through the research, farmers will be able to diagnose diseases with more accuracy and dependability, leading to more prudent and focused use of resources like fertilisers and pesticides. This lessens the impact on the environment and helps to ensure the long-term financial viability of potato cultivation.

Moreover, the research is in line with larger initiatives in precision agriculture, which uses technology to optimise farming methods. By integrating an advanced disease detection system, farmers can gain practical insights that enable them to make well-informed decisions regarding disease management tactics. This includes the potential for on-the-ground real-time surveillance, which would allow for quick reactions to any new threats.

The project's goals are basically multifaceted: it addresses current issues with potato crop disease detection, increases agricultural productivity, encourages sustainability through wise resource management, and fits with the larger goal of utilising technology for precision agriculture. In the end, this project's success could have a favourable effect on both the livelihoods of farmers and the security of food supply worldwide.

**1.2.1 SCOPE**

The detection of potato leaf disease project spans a wide range of possible impact areas and objectives. The project's primary goal is to significantly improve agricultural disease management practices in potato farming by providing a state-of-the-art tool for the early detection and classification of leaf diseases. This could lead to more sustainable farming methods, improved yield, and a decrease in crop losses.

The project's purview encompasses technological innovations, utilizing cutting-edge methods like Convolutional Neural Networks (CNNs) for image analysis. In addition to meeting urgent agricultural needs, research into deep learning in disease detection advances the field of precision agriculture and smart farming.

The project also has effects on environmental sustainability. The system has the potential to minimize the environmental impact of conventional disease management practices by reducing the use of pesticides and other agricultural inputs through the facilitation of more precise and targeted interventions.

Scalability-wise, the project's results might be modified for wider use in different crops and geographical areas, which would lead to a more complete approach to plant disease detection in a range of agricultural contexts. This scalability makes the project more significant than it is in its current context and positions it as a tool that could revolutionize global agriculture.

Furthermore, by investigating real-time disease detection, pathways for the developed system's integration into more expansive agricultural technologies are opened up, resulting in chances for data-driven decision-making and automation in plant health management. This is consistent with the growing movement in agricultural practices toward smart agriculture and the integration of cutting-edge technologies.

In conclusion, the potato leaf disease detection project has a broad scope that includes developments in environmental sustainability, agriculture, and technology. It is positioned as a significant addition to the developing field of precision agriculture, with wider implications for global food security and sustainable farming practices, and its potential impact extends beyond the immediate challenges of disease management.

**1.2.2 NEED FOR POTATO LEAF DISEASE DETECTION**

Potato leaf disease detection is essential because there are major issues with global agriculture and food security that need to be addressed. Millions of people rely on potatoes as a primary source of income and nutrition in many different countries. However, if illnesses are not quickly detected and treated, potato crops are vulnerable to a number of diseases that can result in significant yield losses. Such losses can have serious financial ramifications for both large- and small-scale farmers, as well as the overall economy.

This makes the creation of an efficient system for detecting potato leaf disease crucial for a number of reasons. First and foremost, prompt implementation of interventions, like targeted pesticide application or other disease management tactics, depends critically on early detection. This can significantly slow the spread of illness, cut down on crop losses, and increase overall productivity, all of which will improve local and global food security.

Furthermore, the principles of sustainable agriculture are in line with the effective detection of diseases on potato leaves. Conventional methods of managing diseases frequently entail the careless application of pesticides, which not only puts the environment at risk but also drives up production costs for farmers. The project intends to offer a more sustainable solution by incorporating cutting-edge technologies, such as machine learning and image analysis, enabling accurate and focused responses to disease outbreaks.

The worldwide problem of feeding a growing population emphasizes the necessity of detecting potato leaf disease. Enhancing agricultural productivity becomes critical as food demand rises. The project aims to improve the efficiency of potato cultivation, ensure a more consistent food supply, and address the growing concerns regarding global food security by creating a strong disease detection system.

Furthermore, the application of cutting-edge technology like as Convolutional Neural Networks (CNN) places the project in the broader context of digital agriculture and smart farming, in addition to addressing current agricultural challenges. Real-time disease monitoring and data-driven decision-making are promising developments that are in line with precision agriculture's revolutionary shift, where technology is essential to maximizing resource utilization and enhancing general farm management techniques.

In conclusion, the vital intersections of environmental sustainability, global food security, and economic stability give rise to the necessity of detecting potato leaf disease. The goals of the project are in line with the imperatives to protect an essential food source, improve agricultural resilience, and aid in the creation of creative solutions to the problems that face contemporary agriculture.

**1.3 SUSTAINABLE TECHNOLOGY**

The pursuit of sustainable technology in the identification of potato leaf disease is indicative of a significant movement towards more productive, ecologically responsible, and commercially feasible farming methods. Conventional approaches to illness detection and treatment frequently entail the careless application of pesticides, endangering human health and causing environmental damage. A strong solution in this regard is the incorporation of sustainable technologies, such as sophisticated image analysis and machine learning algorithms.

Sustainable technology in the context of potato leaf disease detection means applying cutting-edge methods to achieve several crucial objectives. First, by making disease management more precise, it tackles the ecological impact of agriculture. The system can precisely identify and classify diseases by utilizing technologies such as Convolutional Neural Networks (CNNs) for image analysis. This allows for targeted interventions and lessens the dependency on broad-spectrum pesticides. This reduces the environmental impact and supports more general agricultural sustainability objectives.

Resource efficiency is enhanced by the project's use of sustainable technology. The technology enables farmers to maximize the use of inputs like fertilizer and pesticides by facilitating early disease detection. This focused strategy lowers production costs while conserving resources, in line with sustainable agriculture's tenets of wise resource use for sustained productivity.

Furthermore, farmers' ability to weather economic hardship is strengthened by sustainable technology used to detect potato leaf disease. Improved yields and stable income are the results of reducing crop losses through early detection. Small-scale farmers' livelihoods depend on this economic sustainability, which also helps to reduce poverty and builds resilience in the face of adversity.

The project's use of sustainable technology also takes advantage of its scalability and adaptability potential. The created system, which is based on cutting-edge machine learning algorithms, can be modified for wider agricultural applications, tackling problems with disease detection in different crops and geographical areas. The project's impact is amplified by its scalability, which positions it as an adaptable instrument capable of revolutionizing disease management practices worldwide.

In addition, the addition of real-time monitoring features to the sustainable technology framework fits in with the larger movement toward smart farming and digital agriculture. Farmers are better equipped to make decisions when timely alerts and data-driven insights are provided, which leads to more effective and adaptable agricultural practices.

In conclusion, there has been a paradigm shift in agricultural practices with the pursuit of sustainable technology for the detection of potato leaf disease. The project supports a more resilient and sustainable agriculture in the future by placing an emphasis on economic resilience, resource efficiency, and precision. This is in line with international efforts to strike a balance between food production and environmental stewardship.

**1.4 MOTIVATION**

The discovery of potato leaf disease is driven by the pressing need to solve major issues pertaining to food security and global agriculture. Potatoes are a staple crop that sustains millions of people's livelihoods and are the main food source for many communities around the world. The susceptibility of potato crops to diverse diseases presents a noteworthy peril to both food production and economic equilibrium.

The financial toll that illnesses have on potato farming is one of the main driving forces. Ignorance of diseases can result in significant yield losses for farmers, both large and small. The financial consequences affect not just individual farmers but also the economies of entire regions and countries, underscoring the need for quick development of efficient disease control instruments. Moreover, the potential of technology to transform agricultural practices emphasizes the need for Potato Leaf Disease Detection. Visual inspection is a common component of traditional disease detection techniques, but it may not be precise or timely. The combination of cutting-edge technologies, like image analysis and machine learning, presents an opportunity to increase illness detection's precision and efficacy.

The need to feed the world's expanding population is a driving force behind this project. As the world's population grows, it is essential to maximize agricultural productivity. The project addresses issues related to global food security by ensuring a more dependable food supply through the development of a strong disease detection system.

The need to detect potato leaf disease is further fueled by the environmental factor. Pesticides are widely used in conventional disease management techniques, which endangers ecosystems and contributes to environmental deterioration. The goal of the project is to reduce the need for indiscriminate pesticide use, minimize environmental impact by enabling targeted interventions, and provide a sustainable alternative.

Moreover, the idea aligns with the broader movement of digital agriculture and smart farming, which is motivated by the desire to use technology to make decisions that are more effective and informed by data. Real-time disease monitoring and proactive management are possible, and this fits in with the revolutionary trend toward precision agriculture, where technology is essential to maximizing resource efficiency and enhancing general farm management techniques.

In the end, the convergence of concerns about global food security, technology, the environment, and economics is what drives the detection of potato leaf disease. Through the resolution of these complex issues, the project hopes to significantly improve the sustainability and resilience of potato farming, thereby advancing the greater objective of guaranteeing food security for an expanding world population.

**1.5 PROBLEM STATEMENT**

The diseases that impact potato crops and the ensuing effects on agricultural productivity are the main causes of the problem at hand. Potato leaf diseases have the potential to cause significant yield losses, which could have an impact on farmers' livelihoods and jeopardize food security. Due to their reliance on visual inspection, which can cause delayed or erroneous disease identification, the current methods for disease detection in potato crops frequently lack the necessary precision and timeliness.

A more sophisticated and successful technological solution is needed for the detection of potato leaf disease given the current circumstances. The need for a reliable system that can precisely identify and categorize diseases affecting potato leaves is highlighted by the shortcomings of current approaches as well as the consequences unchecked diseases can have on the economy and food security. This means applying cutting-edge technology, such as image analysis and machine learning, to enable early and accurate detection, facilitate prompt interventions, and reduce crop losses.

Therefore, the main focus of the problem statement is the requirement for an advanced system to detect potato leaf disease that takes into account the shortcomings of current approaches. The objective is to create a technologically advanced solution that improves disease control efficiency, supports sustainable agriculture by reducing the negative environmental effects of pesticide use, and is in line with the larger goals of ensuring food security in the face of an expanding global population.

**1.6 RESEARCH OBJECTIVES**

Developing an advanced image processing framework specifically for the detection of potato leaf disease is the first research goal. Developing algorithms that can extract pertinent features from leaf images, like color variations, patterns, and textures, is one way to do this. The goal is to create a solid image processing framework that will serve as the basis for precise and thorough analysis of potato leaves, allowing for the detection of minute disease indicators.

The second research goal is to improve the precision and effectiveness of disease classification by applying Convolutional Neural Networks (CNN), a potent Deep Learning architecture. To enable the network to discover complex patterns linked to different diseases, this entails training CNN models on a varied dataset of potato leaf images. The goal is to increase disease detection's level of discrimination and sensitivity by utilizing deep learning's capabilities.

An essential component of efficient agricultural management is real-time disease monitoring. Incorporating real-time monitoring capabilities into the detection system is the third research goal. This includes creating algorithms and procedures for ongoing potato crop monitoring so that quick reactions are possible to new disease threats. The objective is to build a responsive system that can send out alerts in a timely manner, improving the proactive management of plant health.

The thorough assessment of the created detection models is the subject of the fourth research goal. This means assessing the performance of the CNN models and image processing framework using measures such as accuracy, precision, recall, and F1 score.. The objective of the research is to ensure the robustness and effectiveness of the models in a variety of scenarios and disease conditions by iteratively refining and optimizing them based on the evaluation results.

Investigating transfer learning strategies is the fifth research goal, which aims to improve the detection system's applicability. This involves optimizing previously trained models using particular datasets pertaining to diseases of potato leaves. The goal is to improve the models' capacity for generalization by utilizing transfer learning, which will enable them to adjust to differences in disease presentation in various geographic locations and environmental circumstances.

The ultimate research goal is to show that the developed potato leaf disease detection system is both user-friendly and practically viable. This entails working with stakeholders like farmers and agricultural specialists to conduct field tests and usability evaluations. The goal is to guarantee that end users in actual agricultural settings can utilize the technology with practicality and accuracy, in addition to its effectiveness and accuracy.

By addressing these research objectives, the study aims to contribute to the advancement of Potato leaf disease detection technologies, providing a comprehensive and practical solution to the challenges faced in contemporary agriculture.

**1.7 POTATO LEAF DISEASE DETECTION USING TENSORFLOW AND DEEP LEARNING TECHNIQUES**

The detection of potato leaf disease using TensorFlow and Deep learning techniques, such as Convolutional Neural Networks (CNN), Spatial Convolutional Neural Networks (SCNN), and Long Short-Term Memory (LSTM), is a state-of-the-art approach to addressing the main problems in agricultural operations. The open-source machine learning library TensorFlow is being used to demonstrate the dedication to using a stable and scalable framework for creating sophisticated models that can precisely identify and categorize diseases impacting potato crops.

CNN integration is essential to the project because these neural networks are made especially for image recognition applications. From potato leaf images, the system can automatically extract complex patterns and features by utilizing CNNs' hierarchical feature learning capabilities. This greatly improves the detection accuracy of the model by allowing it to distinguish minute differences that may be indicative of various diseases. CNNs are perfect for applications like potato leaf disease detection because they are skilled at capturing spatial hierarchies within images.

The disease detection process's spatial analysis component is greatly improved with the use of SCNN. Because SCNN is designed to take use of the spatial connections between pixels in a picture, it can extract features with greater detail. This is especially helpful when it comes to potato leaf diseases, since the spatial layouts and patterns of the afflicted regions can serve as important indications. By offering a unique framework for spatially aware feature extraction, the SCNN improves the system's overall performance and strengthens the model's capacity to identify intricate illness patterns.

The illness detection system gains a time component with the addition of LSTM. While LSTM is well-suited for studying time-series data, CNNs and SCNNs perform better in spatial analysis. This is because LSTM is made to capture sequential dependencies. LSTM is very helpful in the context of plant diseases, where the evolution of symptoms over time is essential for precise diagnosis. The algorithm grows increasingly skilled at identifying disease trajectories and changing symptoms by taking into account the historical context of leaf health, producing more precise and nuanced forecasts.

The computational foundation of the project is TensorFlow, which offers an adaptable and scalable framework for deep learning model implementation. Because of its adaptability, CNNs, SCNNs, and LSTMs may all be seamlessly integrated into one framework. The distributed computing capabilities of TensorFlow make it possible to train complicated models on huge datasets efficiently, which is essential for creating reliable and accurate disease detection systems.

For the diagnosis of potato leaf disease, a state-of-the-art methodology integrating CNNs, SCNNs, and LSTMs in combination with TensorFlow constitutes a multifaceted deep learning approach. By addressing the problem's geographical, temporal, and sequential components, this all-encompassing method creates a more precise, flexible, and efficient system to support agricultural operations and guarantee the yield and health of potato crops.

**CHAPTER 2**

**LITERATURE STUDY**

**2.1 REVIEW PAPERS ON POTATO LEAF DISEASE DETECTION**

In paper [1], Crop diseases negatively impact crop productivity and yield quality, leading to negative effects on food security and financial losses [1]. In most rural parts of India, agriculture is the main source of income. In this study, a variety of academics and experts carried out a comprehensive evaluation of recent research papers on the detection and categorization of bacterial and fungal plant diseases. Subsequently, we compiled these research works according to crucial elements including the kind of crop employed, the deep learning/machine learning framework employed, the dataset utilized in the trials, performance metrics, the kinds of illnesses identified and categorized, and the maximum precision obtained by the model. As per the research, real-field plant leaf photos were utilized in 70% of studies employing machine learning-based algorithms for disease classification, whereas laboratory-conditioned plant leaf images were employed in 30% of the studies. In contrast, 25% of research using deep learning-based techniques employed real-field photos, 20% used open image datasets, and 55% used laboratory-conditioned images from the Plant Village dataset. The average accuracy of deep learning-based techniques is 98.8%, whereas machine learning-based techniques only get an average accuracy of 92.2%. With regard to methods based on deep learning, We also looked at the outcomes of pre-trained and training-from-scratch models that have been used in several research to classify plant leaf diseases. Pretrained models perform better with 99.64% classification accuracy as compared to training models from scratch, which produced an average accuracy of 98.64%.

In paper [2], Global agricultural productivity and acreage have decreased as a result of fast urbanization and industrialization. Because of this and the increasing demand from well-educated urban families for chemical-free organic vegetables, greenhouses are becoming more and more well-liked for their unique advantages, especially in areas with harsh weather. They create the ideal environment for profitable harvests as well as longer, more fruitful growth seasons. A comprehensive Internet of Things (IoT) Smart Greenhouse system that integrates automation, cloud storage, alerting, monitoring, and disease prediction into a single, readily deployable package is suggested and demonstrated in the current article. In order to ensure improved crop productivity and prompt correction in the event of anomalous conditions, it continuously monitors environmental variables such as temperature, humidity, and soil moisture. There is also an integrated automated irrigation management system. Lastly, the best deep learning model is used for illness diagnosis from leaf photo data. A city dweller may also build a greenhouse and monitor it from his house, enabling him to make necessary corrections, by utilising cloud storage to maximise memory and storage.

In paper [3], Diseases linked to rice leaves often jeopardize the long-term viability of rice production, affecting a significant number of farmers globally. Early detection and timely treatment of rice leaf infection are critical to maintaining the healthy growth of rice plants and ensuring a sufficient supply and food security for the rapidly expanding population. Thus, machine-driven disease diagnostic systems may be able to lessen the drawbacks of traditional leaf disease diagnosis methods, which are frequently costly, time-consuming, and imprecise. Computer-assisted methods for diagnosing rice leaf disease are becoming increasingly common these days. However, there are a number of limitations that compromise the system's effectiveness and use: strong image backgrounds, unclear edges of symptoms, variations in the weather captured in the image, a lack of real field rice leaf image data, variations in symptoms from the same infection, multiple infections causing similar symptoms, and a lack of an efficient accurate system. To solve the above mentioned problems, a faster region-based convolutional neural network (Rapid RCNN) was employed in the current study for the actual time identification of rice leaf diseases. The quicker RCNN approach uses an enhanced the RPN architecture that accurately addresses the object position to give candidate areas. The robustness of the quicker RCNN model is increased via training it using real-field rice leaf datasets that are both publicly and privately available. At 96.09%, 95.85%, and 98.17% accuracy, respectively, It was demonstrated that the recommended deep-learning-based approach was effective in automatically diagnosing brown spot, hispa, and rice blast—three different diseases that affect rice leaves. Furthermore, 98.25% of the time, the model correctly identified a healthy rice leaf.

In paper [4], One of Ethiopia's most important agricultural crops in terms of economics is cotton, however it faces a variety of challenges in the leaf region. These limitations are typically described as difficult-to-detect illnesses and pests. The goal of this work was to create a model that would improve the deep learning method known as CNN for the identification of pests and cotton leaf disease. The researchers have employed common pests and diseases that affect cotton leaves, like leaf miners, spider mites, and bacterial blight, to get goal. The CNN model generalization was enhanced using the k-fold cross-validation technique, which was used to seperate the database. Almost 2400 specimens (six hundred photos in each class) were obtained for training in this study. This produced model is implemented with Python 3.7.3. It is furnished with the TensorFlow-backed Keras deep learning package and Jupyter, which is utilised as the development environment. The accuracy of this approach in classifying pests and leaf diseases in cotton plants was 96.4%. This demonstrated the viability of its implementation in real-time settings and the possible requirement for IT-based solutions to supplement conventional or labor-intensive disease and pest diagnosis methods..

In paper [5], The task of feeding the world's rapidly expanding population is extremely difficult. Fungal, viral, and bacterial diseases of plants can reduce food production because of the disparity between supply and demand. Crop output can be increased by identifying such illnesses early on and using the right fertiliser or pesticide. Therefore, from the outset, ongoing crop monitoring is necessary for the early diagnosis of plant diseases. Some research projects have recently been suggested as corrective actions. These methods, however, require expensive equipment that small-scale farmers cannot afford. Therefore, a low-cost method for plant disease detection is required. The difficulties and possibilities in identifying plant diseases are exaggerated in this study. In accordance with this, this study suggests an ensemble deep learning-based method for diagnosing plant diseases by combining deep learning-based models such as VGG16, ResNet50, and AlexNet. By examining the photos of the plant leaves, it can accurately identify plant illnesses. To assess the robustness of the suggested method, a wide range of experiments were carried out utilising various plant leaf image datasets, including cherry, grape, maize, pepper, potato, strawberry, and cardamom. The suggested method achieved a maximum detection accuracy of 100% for binary datasets and 99.53% for multi-class datasets, according to experimental results.

In paper [6], Plants are a vital source of energy for all living things on the planet. However, plant diseases negatively impact crop life and impede the efficient utilisation of plant products. Many problems occur when farmers manually diagnose illnesses because they lack the necessary knowledge and doctors are not always available. Crop disease identification and classification by hand takes a lot of time as well. Within this framework, a model is put out to diagnose plant diseases and recommend treatments. Here, VGG16 and ResNet50 are used to invent a CNN model based on transfer learning. The set of data is employed includes 34824 training photos and 8767 testing images for a total of 38 output classifications, 26 of which are crop illnesses that were detected in 14 different crops. ResNet50 demonstrated 99.3 percent accuracy with a significant computation time decrease over VGG16, while the VGG16 model demonstrated 99.1 percent accuracy.

In paper [7], Without a question, agriculture is the foundation of the Indian economy and a vital source of income. Many illnesses have a major impact on plant output, but if they are identified early and effectively, they can improve health and spur economic growth. The conventional methods for identifying and categorising diseases need a great deal of time, labour, and ongoing farm observation. Diseases brought on by bacteria, viruses, and fungus can frequently be prevented by employing disease detection techniques. Crop protection is essential to the preservation of agricultural products. Machine learning techniques are frequently employed to identify the impacted leaf images. This paper discusses the different machine learning techniques that are used to identify if a plant is diseased or not. The process involved multiple stages, including acquiring the image, extracting features, classifying the ailment, and displaying the outcome. Additionally, an accurate analysis of the various methods for plant disease detection must be done in this research. The goal is to use image analysis to diagnose plant diseases. It also provides the name of the fertiliser that should be used once the sickness has been identified. There is also a description of the insects and pests that caused the epidemic.

In paper [8], Plant diseases provide significant challenges to agricultural productivity management. They can result in substantial financial loss and inefficiencies in crop productivity when combined with inadequate knowledge to accurately identify crops. The first section of this work focuses on developing a Deep neural network for the detection of illness in maize because of the crop's economic importance and the success of deep learning in other image processing applications. A public domain dataset containing labelled photos of diseased or healthy maize plant leaves is used Convolutional neural network construction and training. This paper's second section uses the trained convolutional neural network to create a real-time smartphone app that can detect maize crop diseases in the field as they're happening. A mobile, easily portable, and affordable method of identifying illness in maize is provided by the smartphone app. In order to take preventative corrective action before there is a considerable loss of yield, the method proposed in this research allows real-time identification of early symptoms of plant diseases using photographs of maize crops in the field. Keywords: smartphone, crop disease diagnosis in real time, artificial intelligence in agriculture.

In paper [9], The project's goal is to improve the current medical system by creating an automatic recognition model for the classification of therapeutic plants using Machine learning (ML) , internet of things (IoT) techniques. The oldest medical system in India, Ayurveda, is still in use today because it encourages the prescription use of herbs for a variety of illnesses. To just a few benefits, herbs are inexpensive, readily available, and hardly cause negative effects. Although conventional medicine is widely acknowledged as a superior option to synthetic medications, its use has decreased due to false information and a lack of evidence. Here, a creative technique is proposed that makes use of the Raspberry Pi Model's camera to identify images of Indian medicinal plants in real time and reveal each one's unique therapeutic properties. Four machine learning models are developed in this study, one of which is proposed to identify features of a captured medicinal leaf on the RPi user interface. With 1526 leaf pictures from 25 distinct medicinal plants in a custom leaf dataset, the suggested model incorporates two feature extraction techniques: Using the histogram of directed gradients and the scale-invariant feature transform (SIFT), an accuracy (top-1) of 97.93% was predicted. The bag of visual words is assessed using a support vector machine classifier subsequent to feature selection by k-means clustering on SIFT descriptors. With RPi integration, the suggested model exhibits 98% actual time highest-3 correctness. The design of the system has the advantage of being tailored for medicinal plants, having less expensive cameras, and even working effectively in isolated regions.

In paper [10], Diseases and crop pests pose serious risks to the world's food security. In India, one of the most important crops is the mung bean (Vigna Radiata). In India, a sizable portion of the populace is entirely reliant on mung beans. Therefore, high production efficiency is needed for mung beans, which isn't achieved because of considerable damage from illnesses and pests. convolutional neural networks (CNN) have recently shown impressive performance in image categorization thanks to the development of Deep Learning techniques This holds great potential for the identification of diseases and pests through efficient picture classification. In this research, we have proposed a novel deep learning-based approach to identify pests and diseases in mung beans. We have used transfer learning to address the issue of a limited quantity of mung bean crop photos available for training, which has the potential to produce highly promising results for rapid and simple pest and disease detection. Using healthy and diseased leaves collected over the course of several seasons, the proposed model has successfully identified six distinct mung bean illnesses and four different types of pests .The average accuracy of the proposed smartphone-based deep learning model for mung bean pest and disease detection is 91.55%., according to the conducted experiments.

**CHAPTER 3**

**PROPOSED METHODOLOGY**

**3.1 Convolutional neural network (CNN)** :

**3.1.1 Introduction to Convolutional neural networks (CNN)**

One kind of deep learning technique that is frequently used for image recognition and classification applications is the convolutional neural network (CNN). Convolutional, pooling, and fully connected layers are some of the layers that make up CNNs. Each layer has a distinct purpose in the feature extraction and classification process.

The input layer, the first layer of a CNN, receives the image as a 2D matrix of pixel values. The input is then sent through a series of filters applied by the convolutional layer, each of which conducts a convolution operation to extract particular features from the image. After that, a rectified linear unit (ReLU) activation function is applied to the feature maps to add non-linearity and enhance the model's capacity to learn intricate features.

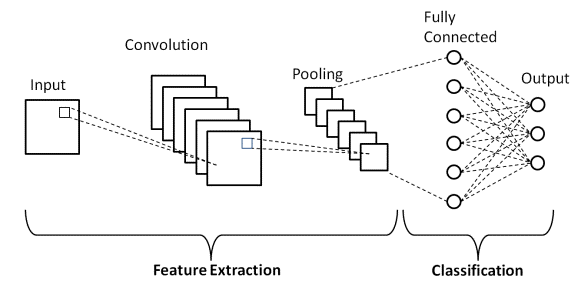
A pooling layer is usually added after the convolutional layer in order to downsample the feature maps and lower the input's dimensionality. Usually, max pooling—which chooses the highest value in each feature map region—is used to do this. After going through more convolutional and pooling layers, the pooled feature maps are flattened and sent to a fully connected layer, which completes the classification. Because CNNs employ many convolutional and pooling layers to learn

Fig 3.1. Architecture for CNN

hierarchical features from the input image, they are generally very good at image recognition and classification tasks.

**3.1.2 Layers in CNNs**

Convolutional neural networks, or CNNs, are made up of multiple layers, each of which serves a distinct purpose in the process of classifying and extracting features. The following are the most typical CNN layers:

1. Input Layer: A 2D matrix of pixel values is used by this layer to receive the input image.

2. Convolutional Layer: Using a series of filters, this layer convolutionally extracts particular features from the input image. Every filter gains the ability to identify particular patterns or features in the image, like corners or edges.

3. Activation Layer: This layer adds non-linearity to the model and enhances its capacity to learn intricate characteristics by using an activation function, such as the rectified linear unit (ReLU) function.

4. Pooling Layer: By downsampling the feature maps generated by the convolutional layer, this layer lowers the input's dimensionality and increases the computational efficiency of the model. Max pooling, which chooses the maximum value in each feature map region, is the most popular pooling strategy.

5. Fully Connected Layer: This layer performs the final classification using the output that has been flattened from the preceding layers. The model can identify intricate patterns and relationships between the features because every neuron in the layer is connected to every other neuron in the layer above it.

**3.1.3 Feature Extraction with Convolutional Layers**

From an input image, Convolutional Neural Networks (CNNs) are trained to identify and extract the most pertinent information. The CNN's convolutional layers handle the feature extraction procedure.

A collection of filters, sometimes referred to as kernels, make up each convolutional layer. These kernels convolve over the input image to create a set of feature maps. Every filter takes a particular feature, like an edge, corner, or blob, and extracts it from the input image.

Every filter performs a complete scan of the input image during the convolution process, computing a dot product between the filter weights and the corresponding image pixel values. One output value is produced by this dot product operation, and it is assigned to the appropriate spot in the feature. The output feature maps produced by the convolutional layers are a result of applying the filters to the input image. These feature maps contain important information about the spatial relationships between the features in the input image.

CNNs usually use many convolutional layers with an increasing number of filters to capture various features. As data moves through the layers, this enables the model to learn increasingly intricate and abstract properties.

In general, the CNN pipeline's feature extraction procedure with convolutional layers is essential because it helps the model identify and learn from the most pertinent elements in the input image.

**3.1.4 Optimization Techniques for CNNs**

Strong deep learning models known as convolutional neural networks (CNNs) have demonstrated remarkable performance in a range of computer vision applications, such as segmentation, object detection, and image categorization. However, when working with big datasets and intricate structures, training these models can be costly and time-consuming computationally. Numerous optimisation strategies have been developed to address these issues and raise CNN performance and efficiency levels.

1. Batch Normalisation: During training, the activations of the preceding layer are normalised for each batch using the batch normalisation technique. This aids in lessening the internal covariate shift, a phenomena that makes it more difficult to train a network by changing the distribution of inputs to a layer during training. Batch normalisation helps the network learn more effectively and converge more quickly by standardising the inputs.

2. Dropout: During training, a specific proportion of the neurons in a layer are randomly removed using the dropout regularisation technique. By limiting the network's reliance on any one neuron and pressuring it to acquire more resilient and generalizable properties, this helps avoid overfitting.

3. Data Augmentation: Using random transformations like flips, translations, and rotations on the input images, data augmentation creates an artificially larger training dataset. By exposing the network to additional changes in the training data, this helps to improve the network's generalisation performance.

4. Transfer Learning: Pretrained weights of an existing CNN model can be used to refine it on a new task or dataset using a technique called transfer learning. Starting with a pre-trained model that has previously learned generic features helps to reduce training time and data needs.

5. Gradient Descent Optimisation: Using this strategy, the network's weights are updated in the direction of the loss function's steepest descent. Numerous techniques, such as stochastic gradient descent (SGD), Adam, and Adagrad, have been developed to increase the stability and efficiency of gradient descent optimisation.

All things considered, these optimisation methods are essential for enhancing CNN performance, generalisation, and efficiency and are frequently employed in contemporary deep learning pipelines.

**3.1.5 Advancements in CNNs**

CNN developments have been instrumental in revolutionising a number of fields. The creation of deep CNNs, which have made it possible to create models that are more precise and complicated, is one of the main developments. These models have proven useful in a number of domains, including speech recognition, natural language processing, and image recognition.

The application of transfer learning, which includes taking pre-trained models and optimising them for certain tasks, is another noteworthy breakthrough. This method has made it possible to create models more quickly and effectively, particularly in situations when there aren't a lot of training data available. All things considered, these developments have increased the strength and adaptability of CNNs as instruments for a variety of uses.

In general, the CNN pipeline's feature extraction procedure with convolutional layers is essential because it helps the model identify and learn from the most pertinent elements in the input image.

All things considered, these optimisation methods are essential for enhancing CNN performance, generalisation, and efficiency and are frequently employed in contemporary deep learning pipelines.

**3.2. Long Short-Term Memory**

**3.2.1** **Introduction to Long Short-Term Memory (LSTM)**

Recurrent neural network (RNN) architecture with Long Short-Term Memory (LSTM) was created to overcome the difficulties associated with learning and remembering information over extended periods of time. It works especially well for sequential data tasks including time series analysis, speech recognition, and natural language processing.

1. Handling Sequential Memory In contrast to conventional neural networks, which have trouble storing information across long sequences, long short-term memory (LSTM) is highly effective at identifying and keeping patterns in sequential data. Through the use of gating mechanisms and specialised memory cells, the architecture enables the network to store and retrieve data selectively over extended periods of time.

2. Memory Cells and Gates: Long-term memory cells (LSTMs) are containers that can hold information for a long time. Information is controlled by gating mechanisms, which include the forget gate, output gate, and input gate, which control information entering, leaving, and inside memory cells. The network is prevented from either overemphasising or ignoring particular inputs by this restricted information flow.

3. Mitigation of Vanishing Gradient Problem: Long Short-Term Memory Networks (LSTMs) mitigate the vanishing gradient problem that conventional RNNs face, in which the training process reduces the influence of distant past data. The network may selectively update or reject data thanks to the gating mechanisms in the architecture, which lessen the influence of disappearing gradients and promote more efficient learning.

4. Applications: LSTMs are used in many different fields, including as sentiment analysis and language translation in natural language processing, as well as time series analysis for trend and pattern prediction.

To put it simply, long short-term memory networks are an effective way to learn and remember knowledge for tasks that require sequential data. They are useful tools in a variety of applications needing a nuanced understanding of context and patterns over time due to their capacity to selectively handle and store information over extended periods.

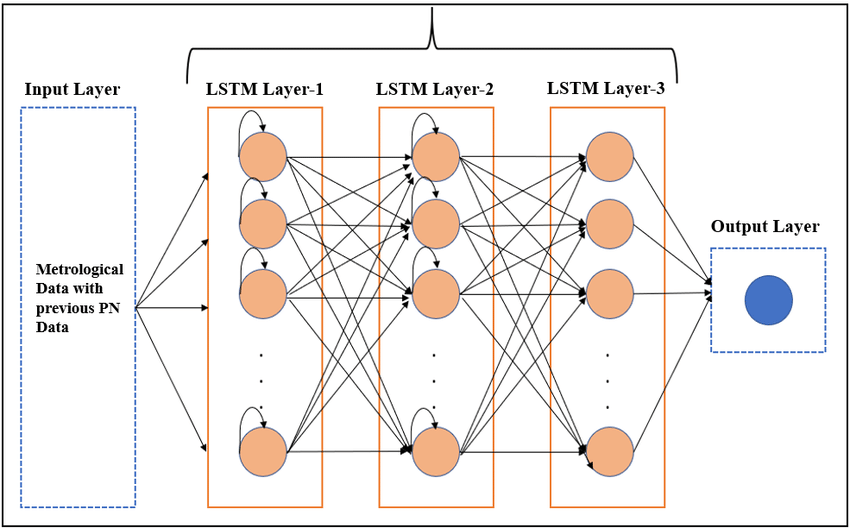


Fig 3.2. Architecture for LSTM

**3.2.2 Layers in Long Short-Term Memory (LSTM):**

The several layers that make up Long Short-Term Memory (LSTM) networks each have a distinct purpose in storing and handling sequential data. The following are the main levels of an LSTM architecture:

1. Input Layer: Sequential data, representing time series, natural language, or any other type of sequential input, is received by the input layer. After processing, each sequence element is put into the network for additional analysis.

2. Memory Cell: The memory cell, which holds information across lengthy durations, is the key component of an LSTM. It is made up of a cell state that contains long-term data and is managed by different gates to control data flow.

3. Forget Gate: This mechanism chooses which data from the previous cell state ought to be retained and which ought to be erased. It generates a forgetting factor that modifies the cell state by combining information from the prior concealed state with the current input.

4. Input Gate: This component determines what fresh data needs to be kept in the memory cell. In order to provide a candidate value for the new cell state, a tanh activation is used in conjunction with a sigmoid activation to determine which values will be updated.

5. Cell State Update: The fresh information ascertained by the input gate and the information kept by the forget gate are combined to update the cell state.

6. Output Gate: This gate controls the information that is output as the final concealed state from the cell state. It uses a tanh activation to represent the cell state and a sigmoid activation to decide which values to output.

7. Hidden State: The hidden state is both the LSTM's output at a given time step and its input for subsequently occurring time steps.

Together, these layers address the difficulties associated with learning and remembering knowledge over extended periods of time. Particularly the gating methods allow LSTMs to selectively update, forget, and output data, which is why sequential data tasks are a good fit for them.

**3.2.3 Feature Extraction with Long Short-Term Memory (LSTM):**

Although their primary purpose is sequential data processing, Long Short-Term Memory (LSTM) networks can also be utilised for feature extraction in a variety of applications. Using LSTMs, feature extraction is achieved as follows:

1. Sequential Learning: Long Short-term memory (LSTM) models are able to capture sequential patterns and relationships in data, which makes them appropriate for tasks requiring the knowledge of context over time.

2. Time Series Data: LSTMs may learn patterns and trends over time by extracting features from past data points in time series analysis. Long short-term dependencies are captured by LSTMs thanks to their memory cells and gating processes, which also help them recognise subtle patterns.

3. Natural Language Processing (NLP): sentiment evaluation, language translation, and speech recognition are just a few of the NLP tasks that frequently employ LSTMs. LSTMs can extract linguistic features, comprehend context, and produce meaningful representations by processing sequential input.

4. Image Sequences: LSTMs can be used to extract characteristics from the temporal evolution of visual data by applying them to image sequences.This is helpful for applications where the temporal aspect of visual data is crucial, such as action detection and video analysis.

5. Financial Time Series: LSTMs can aid with feature extraction for activities like stock price prediction by identifying patterns in financial time series data. To find pertinent features, the network learns from past price changes and market trends.

6. Healthcare Data: Using time series data from patient monitoring systems, LSTMs can be used in the healthcare industry to extract features pertaining to abnormalities or trends in health. This is especially helpful for early medical condition diagnosis and predictive modelling.

7. Event Sequences: LSTMs are capable of processing event sequences and extracting features that show how a series of occurrences evolved over time. Applications such as anomaly detection and event prediction can benefit from this.

8. Sensor Data: LSTMs may extract characteristics from sequential sensor readings for applications using sensor data, such as Internet of Things devices. The network offers important insights by identifying patterns and trends in the sensor data.

Essentially, long short-term memory (LSTM) models are excellent at extracting complex correlations and patterns from sequential data, which makes them useful for a wide variety of feature extraction tasks. They are able to extract significant features for intricate jobs because of their capacity to retain information over long sequences.

**3.2.4 Optimization Techniques for LSTM**

In order to train and optimise machine learning models to perform better, optimisation techniques are essential. The following are a few typical optimisation methods:

1. Stochastic Gradient Descent (SGD): SGD is a basic optimisation approach that adjusts the model parameters in a way that minimises the error in order to minimise the loss function.

The term "stochastic" describes how the algorithm uses a fraction of the training data for each iteration in order to update the parameters.

2. Batch Gradient Descent: The model parameters are updated in accordance with the gradient of the loss function, which is computed using the complete training dataset.It provides a more accurate estimate of the gradient but can be computationally expensive for large datasets.

3. Mini-Batch Gradient Descent: By updating the parameters using a tiny random subset (mini-batch) of the training data, mini-batch gradient descent finds a middle ground between batch gradient descent and SGD. This technique combines batch gradient descent's stability with SGD's efficiency.

4. Learning Rate Scheduling: Convergence may depend on modifying the learning rate during training. In order to enable the model to converge more quickly, learning rate scheduling entails gradually lowering the learning rate.

5. Momentum: By applying a moving average of the gradient, momentum introduces a strategy that helps accelerate SGD in the appropriate direction and dampens oscillations. It facilitates the optimisation algorithm's passage through brief minima and speeds up convergence.

6. Adagrad: Adagrad uses historical gradients to modify the learning rates of each parameter separately. For infrequent parameters, it makes larger updates, and for frequent parameters, it makes smaller changes.

7. RMSprop: Root Mean Square Propagation, or RMSprop, is an adaptive learning rate optimisation approach that keeps track of each parameter's moving average of squared gradients. By normalising the gradient updates, it reduces its sensitivity to the parameter scale.

8. Adam: The popular optimisation algorithm Adam (Adaptive Moment Estimation) blends the concepts of momentum and RMSprop. It preserves both the decaying average of previous squared gradients (RMSprop) and the decaying average of previous gradients (momentum).

9. Batch Normalisation: This method helps to reduce problems such as vanishing or inflating gradients by normalising each layer's input. It increases training stability and speed, enabling faster learning rates.

10. Regularisation Techniques: By forbidding overly complex models, techniques like as L1 and L2 regularisation add penalty terms to the loss function, hence preventing overfitting. Another regularisation method called dropout involves randomly removing neurons from the training population in order to stop hidden units from co-adapting.

Effective model training, avoiding convergence problems, and improving generalisation performance all depend on these optimisation strategies. The particulars of the dataset and the problem will determine which optimisation approach is best.

**3.2.5 Advancements in LSTM**

Recent years have witnessed several developments in recurrent neural networks (RNNs), specifically in Long Short-Term Memory (LSTM) networks. Here are a few noteworthy advancements:

1. Attention Mechanisms: By including attention mechanisms into LSTM architectures, the model is better able to concentrate on pertinent segments of the input sequence, which improves performance in applications like natural language processing and machine translation.

2. Gated Recurrent Units (GRUs): GRUs are becoming more popular as an LSTM substitute. Even though they are less complex than LSTMs, they have demonstrated competitive performance in some applications, which has prompted continued research and analysis of the two architectures.

3. Bi-Directional Long Short-Term Memory LSTMs: These LSTMs process input sequences both forward and backward, storing contextual data from both the past and future. This has shown to be helpful for applications like speech recognition, where context from both directions is essential.

4. Advanced Architectures: To improve the model's ability to capture hierarchical representations in sequential data, researchers have suggested modifying the fundamental LSTM structure. Examples of these changes include depth-wise separable LSTMs.

5. Transfer Learning with LSTMs: Using methods taken from the field of transfer learning, LSTMs may now be fine-tuned for particular tasks and pre-trained on big datasets. This has proven especially helpful in situations where task-specific data is scarce.

6. Effective Training Methods: Developments in parallelization and optimisation methods have sped up LSTM training, increasing its scalability and suitability for large-scale applications.

7. Hybrid Models: By combining LSTMs with other neural network architectures, including transformer models or convolutional neural networks (CNNs), hybrid models that combine the best features of several architectures to achieve better performance have been created.

8. Memory-Efficient LSTMs: In order to overcome issues with resource-intensive calculations and increase their viability for deployment on edge devices, research has concentrated on creating LSTMs that are more memory-efficient.

9. Applications in Diverse fields: Long short-term memory banks (LSTMs) are finding success in an increasing number of fields, such as natural language processing (language modelling and sentiment analysis), healthcare (patient monitoring), and finance (time series analysis).

10. Explainability and Interpretability: Efforts have been made to improve LSTMs' interpretability, particularly in crucial areas like finance and healthcare, by making them more transparent and understandable.

**3.3 Spatial Convolutional Neural Network (SCNN)**

**3.3.1** **Introduction to Spatial Convolutional Neural Network (SCNN)**

A specific kind of convolutional neural network (CNN) made to process and analyse spatial data seen in images is called a spatial convolutional neural network (SCNN). In contrast to conventional CNNs, which concentrate on identifying patterns and features throughout the image, SCNNs pay particular attention to the spatial interactions among pixels.

1. Spatial Information Processing: SCNNs are designed to capture the spatial dependencies and relationships between pixels in an image, allowing them to comprehend the contextual information of nearby regions. This is one of the key features and functionalities of SCNNs.

2. Local Feature Extraction: To extract local features from various spatial regions of the input, the network uses convolutional layers with learnable filters. This makes it possible for the model to identify minute details and patterns inside particular regions.

3. Contextual Understanding: SCNNs have procedures that take into account each pixel's surrounds and spatial context. For tasks like semantic segmentation, where the spatial arrangement of features is critical, this contextual understanding is very helpful.

4. Segmentation Tasks: Typically, segmentation tasks include applying convolutional neural networks (CNNs) to classify and assign a class to each pixel in an image. SCNNs are excellent at defining object borders and creating fine-grained segmentation maps because they place a strong emphasis on spatial relationships.

5. Architecture Modifications: To improve the network's capacity to record long-range dependencies and interactions between pixels, SCNNs may incorporate specialised layers or spatial attention algorithms.

6. Applications: SCNNs are used in computer vision tasks like object recognition, scene parsing, and picture segmentation in domains like autonomous driving and medical image analysis, which call for precise spatial analysis

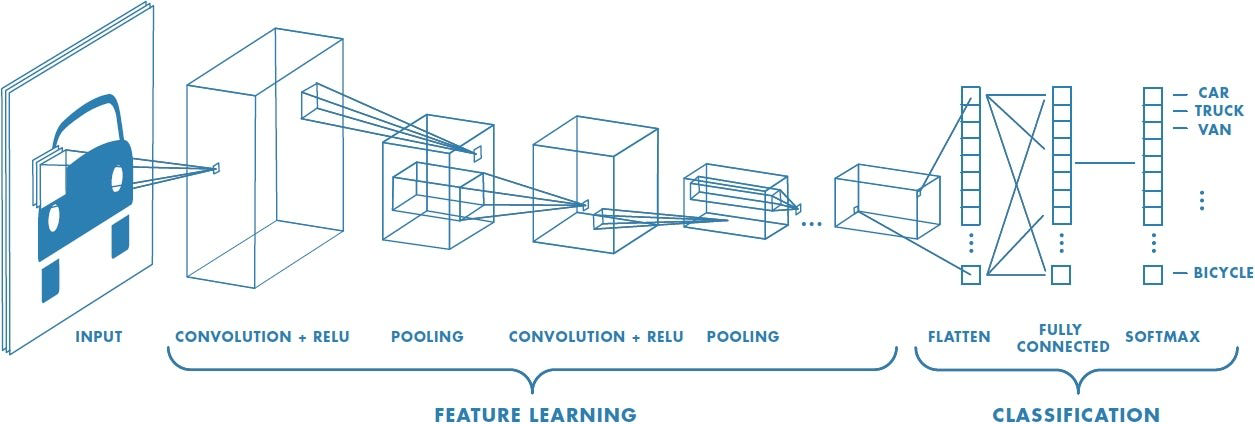


Fig 3.3. Architecture for SCNN

**3.3.2 Layers in SCNNs**

Depending on the design suggested in research articles or particular implementations, the precise layers in a SCNN can change. I can, however, give you a broad notion of the layers that are frequently present in structures made for segmentation and spatial analysis tasks:

1. Convolutional Layers: Essential for extracting spatial features. Convolutional filters are used by these layers to scan the input image and identify spatial patterns.

2. Spatial Convolutional Layers: These layers are made especially for the study of space. To underline the significance of spatial relationships and contextual information, they could incorporate extra processes.

3. Pooling Layers: To lower computational complexity and spatial dimensions. Max pooling and average pooling are two popular pooling techniques.

4. Spatial Attention Mechanism: In order to concentrate on significant spatial regions, certain SCNN designs may include attention methods. The model's ability to represent long-range interdependence is improved by these approaches.

5. Fully Connected Layers: Usually present in the network's later phases for tasks involving segmentation or classification. They link each neuron in the current layer to every neuron in the layer above.

6. Normalisation Layers: To speed up and stabilise the training process, batch normalisation or layer normalisation layers may be used.

7. Activation Layers: To give the model non-linearity, use the Rectified Linear Unit (ReLU) or other activation functions.

8. Upsampling Layers: These are crucial for jobs like picture segmentation, in which spatial data must be recovered from downsampled images. Techniques like transposed convolution and bilinear interpolation are frequently used.

9. Output Layer: The output layer might change based on the particular activity. A softmax layer can be used to generate probability maps for every class in segmentation tasks.

10. Skip Connections: To help lessen the effects of the vanishing gradient issue, skip connections, also known as residual connections, can be used to let information move over several levels.

11. Dropout Layer: This layer randomly removes neurons from training in order to prevent overfitting through regularisation.

**3.3.3 Feature Extraction with SCNN**

Capturing pertinent spatial information and patterns from input data—typically images—is the process of using a Spatial Convolutional Neural Network (SCNN) for feature extraction. Here is a detailed explanation of how SCNN is used for feature extraction:

1. Input Data: Typically represented as a picture, the input data is a spatial dataset. The SCNN seeks to analyse the spatial information included in each pixel of the image.

2. Spatial Convolutional Layers: To process the input, SCNNs use spatial convolutional layers. These layers extract spatial characteristics from the input image by scanning it with learnable filters. Every filter isolates and highlights particular areas of the picture to reveal connections and patterns.

3. Optional Spatial Attention Mechanism: A spatial attention mechanism is included in certain SCNN systems. By improving the network's concentration on significant spatial regions, this method enables the model to give particular areas priority during the convolutional process.

4. Pooling Layers: Pooling layers are frequently used to downsample the spatial dimensions of the data after convolution. In order to reduce computational complexity and preserve important properties, max pooling or average pooling is used.

5. Normalisation and Activation Layers: To stabilise the training process, either layer normalisation or batch normalisation can be used. By adding non-linearity to the model, activation functions like ReLU enable the model to capture intricate spatial representations.

6. Skip Connections (Optional): In order to improve information flow across layers, certain SCNN systems use skip connections. This makes it possible for the network to record both local and global spatial data and helps to avoid the vanishing gradient problem.

7. Output, or Intermediate Features: A collection of intermediate features that symbolise the retrieved spatial information is the SCNN layers' output. Patterns, textures, and spatial relationships found in the input data are encoded by these features.

8. Additional Processing (Optional): The extracted characteristics may go through additional processing, depending on the nature of the assignment. Layers for up sampling can be used for tasks like image segmentation in order to recover spatial information that was lost during down sampling.

**3.3.4 Optimization Techniques for SCNNs**

The goal of optimisation strategies for Spatial Convolutional Neural Networks (SCNNs) is to enhance the network's generalisation performance, convergence speed, and training efficiency. Here are a few popular methods of SCNN optimisation:

1. Stochastic Gradient Descent (SGD): Based on the gradient of the loss function with regard to the network's parameters, SGD is a basic optimisation technique that updates the parameters. To improve computational efficiency, mini-batch SGD is frequently used to compute the gradient on a fraction of the training data.

2. Learning Rate Scheduling: It may be advantageous to modify the learning rate while receiving instruction. Improved convergence and avoiding overshooting the ideal parameter values can be achieved with the aid of strategies like learning rate annealing, in which the learning rate gradually drops.

3. Momentum: This method reduces oscillations and aids in accelerating SGD in the appropriate direction. The optimizer can overcome minor gradients and proceed in the right direction by introducing a moving average of previous gradients.

4. Methods of Adaptive Learning Rate: Based on each parameter's historical gradients, adaptive techniques like Adam, RMSprop, and Adagrad modify the learning rate for that parameter. During training, these techniques can adjust to the changing weight of various parameters.

5. Batch Normalisation: By normalising a layer's inputs, batch normalisation lowers internal covariate shift and speeds up training. It facilitates and expedites the optimisation process.

6. Dropout: This regularisation method involves randomly removing neurons from the training set to stop hidden units from co-adapting. This enhances the network's generalisation and helps avoid overfitting.

7. Gradient Clipping: A predetermined threshold is applied to gradients during backpropagation. If the gradient is larger than the threshold, it is reduced in size. This prevents slopes from blowing up and stabilises training.

8. Early Stopping: To reduce overfitting and enhance generalisation, monitor the model's performance on a validation set during training and stop the model when the performance begins to deteriorate.

9. Transfer Learning: When labelled data is scarce, using previously trained models on related tasks or domains can speed up training and enhance performance.

10. Data Augmentation: By adding transformations like flips, rotations, and scaling to the training set, you can improve generalisation by broadening its diversity.

**3.3.5 Advancements in SCNNs**

A new age of spatial data processing has been brought about by the major breakthroughs made in Spatial Convolutional Neural Networks (SCNNs) in recent years. Scholars and professionals have concentrated on subtle enhancements that go beyond conventional capacities. By incorporating attention processes, SCNNs are now able to highlight important spatial regions and improve their capacity to identify fine details in large, complicated datasets. Combining additional modules or other neural network types with SCNNs to create hybrid designs has become popular, promoting synergy for improved performance. Refinements in transfer learning strategies tailored for SCNNs have unlocked more efficient leveraging of pre-trained models and representations, amplifying their efficacy.

Resources-constrained environments can now use SCNNs because real-time processing is made possible by advances in computer efficiency. The addition of temporal information has improved SCNNs, making them more applicable by enabling them to understand temporal dynamics in addition to spatial complexities. Through the use of adversarial training approaches, SCNNs' resilience to adversarial attacks has been strengthened, leading to more resilient performance. By using self-supervised learning techniques, SCNNs are now able to extract valuable information from unlabeled geographical data, reducing the need for large-scale labelled datasets. A significant focus is on improving the interpretability of SCNNs by continued research into methods that clarify the decision-making procedures, promoting increased openness and confidence.

Domain adaptation strategies have progressed, enabling SCNNs to generalize effectively across varying spatial domains. This confluence of advancements signifies the maturation of SCNNs, positioning them as formidable tools for spatial analysis across a spectrum of real-world applications.

**CHAPTER 4**

**MODULE/ EMPIRICAL STUDY**

**4.1 Pre-processing techniques**

Pre-processing is an essential step in any machine-learning project. In the context of the plant leaf classification project, pre-processing involves techniques such as data splitting, caching, and shuffling to ensure speed and accuracy. Data splitting involves dividing the dataset into separate sets for training, testing, and validation. This ensures that the model does not overfit the training data and performs well on unseen data.

Caching is used to store the pre-processed data in memory or disk for faster retrieval during training. This reduces the overall time required for data loading and pre-processing, thereby increasing the efficiency of the model. Shuffling is another pre-processing technique that randomizes the order of the training examples to prevent the model from learning the order of the data.

All these pre-processing techniques help to ensure that the model trains on high-quality, unbiased data and performs well on new, unseen data. By optimizing the data inputs, pre-processing helps to improve the overall accuracy and efficiency of the model.

**4.2 Segmentation**

After pre-processing the dataset, the next step is to segment it into training, testing, and validation sets. Segmentation is important as it allows us to evaluate the performance of our model on data that it has never seen before, which is crucial to avoid overfitting.

Typically, the dataset is split into a training set (used to train the model), a validation set (used to tune hyperparameters and prevent overfitting), and a testing set (used to evaluate the final performance of the model). The split ratio can vary depending on the size of the dataset, but a common split is 70% for training, 15% for validation, and 15% for testing. It is important to ensure that the data is split randomly and that each split contains a representative sample of the entire dataset. This can be achieved using techniques such as stratified sampling, which ensures that each class is represented proportionally in each split.

Overall, segmentation is a crucial step in the development of a machine learning model, as it allows us to evaluate its performance and make necessary adjustments to improve its accuracy.

**4.3 Feature extraction**

After segmenting the dataset, the next step is feature extraction. In this stage, we extract relevant information or features from the images that can be used by the machine learning model to differentiate between healthy and diseased plant leaves. Color and texture features are extracted from the leaf images.

Color-based features can be extracted using color histograms or color moments, while texture-based features can be extracted using techniques such as Local Binary Patterns (LBP) or Gray-Level Co-occurrence Matrix (GLCM).

These features can provide a rich representation of the plant leaf images, enabling the machine-learning model to make accurate predictions. The choice of feature extraction technique depends on the nature of the dataset and the problem being solved.

Feature extraction is a critical step in the machine learning pipeline as it determines the quality of features that the model uses to make predictions. A good feature extraction method should be able to capture the essential characteristics of the plant leaves while eliminating irrelevant information.

**4.4 Feature selection**

After extracting the features from the images, it is important to select the most important and relevant features that contribute most towards the classification of the images as healthy or diseased. This step is called Feature selection. There are various methods to perform feature selection such as filter methods, wrapper methods, and embedded methods. In this project, filter methods are used for feature selection.

Filter methods work by ranking the features based on their relevance to the target variable. The features are scored using statistical tests such as chi-square, ANOVA, and correlation. The features with the highest scores are then selected for further analysis. The advantage of using filter methods is that they are computationally efficient and can handle a large number of features.

In this project, filter methods are used to select the most important features from the extracted color and texture features. The features are ranked based on their correlation with the target variable (healthy or diseased). The top-ranked features are then selected for training the deep learning model. This step helps in reducing the dimensionality of the dataset and improving the accuracy of the model.

**4.5 Evaluation**

Evaluation is an essential part of any machine learning project, as it helps to measure the performance of the model. In this project, we evaluated our CNN model using the ROC curve and accuracy metric. The ROC curve is a plot that helps to visualize the performance of the classification model at different classification thresholds. It represents the trade-off between true positive rate (sensitivity) and false positive rate (1-specificity) for various threshold values. The ROC curve is a useful tool for selecting the best threshold for a particular classification problem.

Accuracy is another widely used metric to evaluate the performance of classification models. It represents the percentage of correctly classified samples in the test dataset. In this project, we used accuracy as a metric to evaluate the overall performance of our CNN model. However, accuracy alone may not be sufficient in some cases, especially when the dataset is imbalanced. In such cases, precision, recall, and F1 scores may also be used to measure the performance of the model.

To evaluate the performance of our model, we split the dataset into training, validation, and testing sets. We trained the model on the training set, tuned the hyperparameters on the validation set, and evaluated the model's performance on the testing set. We plotted the ROC curve and calculated the accuracy of the model on the test set. By analyzing the ROC curve and accuracy, we determined whether the model was overfitting or underfitting and made necessary adjustments to improve its performance.

**4.6 Final Prediction**

The final step in the project is to predict whether an input image of a plant leaf is healthy or diseased based on the trained model. To make a prediction, we pass the preprocessed and segmented image through the trained CNN model. The model will then output a probability score indicating the likelihood that the input image is healthy or diseased.

To make a final prediction, we set a threshold probability score. If the output probability score is greater than the threshold, we classify the input image as diseased, and if it is less than the threshold, we classify it as healthy. The threshold value can be set based on the desired trade-off between false positives and false negatives.

Once the final prediction is made, it can be displayed to the user along with a confidence score indicating the level of confidence that the model has in the prediction. This can help the user make informed decisions about the health of their plants and take appropriate actions to prevent or treat any diseases.

**CHAPTER 5**

**CONCLUSION**

Our study shows that deep convolutional neural networks are remarkably effective at correctly identifying and classifying potato leaf diseases. With an F1-score of 96.71%, the generated model demonstrated remarkable accuracy in identifying two common plant diseases. With the use of this cutting-edge method, plant disease detection precision might be greatly increased, resulting in quicker and more effective treatment plans. This technology has the potential to completely transform the agricultural industry by reducing the negative effects of plant diseases on crop yields and raising total agricultural production with more research and development.

**CHAPTER 6**

**FUTURE ENHANCEMENT**

Expansion of the dataset: Although the dataset utilised in this study is already somewhat substantial, many plant diseases are still missing. More images of sick plant leaves can be collected and added to the dataset in the future to improve the model's accuracy and resilience.

Transfer learning: Another intriguing topic for further study is the application of transfer learning. This strategy involves fine-tuning a pre-trained CNN that has been trained on a broad, diversified dataset using a smaller, more focused dataset, such as the plant leaf disease dataset used in this work. As a result, the model might perform better and need less training time.

Real-time illness detection: At the moment, the study's algorithm is intended to categorise photos of sick plant leaves. But it would be intriguing to investigate the prospect of creating a real-time system in the future that can identify illnesses as they arise in the field. In order to enable more prompt and focused treatment, this may entail employing cameras or other sensors to monitor plants and notify farmers when the illness is discovered.

Multi-class classification: The current model is able to distinguish between two distinct diseases, namely potato early blight and potato late blight, and healthy leaves. To enable the model to be applied in more varied agricultural environments, it would be beneficial to expand its classification capabilities to encompass a broader spectrum of illnesses in the future. This would necessitate expanding and diversifying the dataset in addition to altering the CNN's design to accommodate multi-class categorization.

**REFERENCES**

[1] Singh, S. K., & Gupta, S. K. (2021). Deep learning-based plant disease detection: A comprehensive review. In Intelligent Computing Techniques for Smart Energy Systems (pp. 307-321). Springer, S

[2] Sharma, R., & Kanwar, P. (2021). An overview of plant disease detection techniques using deep learning. Journal of Ambient Intelligence and Humanized Computing, 12(7), 6773-6786.

[3] Raza, G., & Raza, A. (2021). Deep learning-based plant disease detection using transfer learning: A review. International Journal of Computer Science and Network Security, 21(4), 138-147.

[4] Patel, S. K., & Patel, S. R. (2021). A review on plant disease detection and classification using deep learning approach. In Proceedings of International Conference on Emerging Trends in Computer Science and Information Technology (pp. 373-383). Springer.

[5] Nair, M., Gopi, R., & Kumar, S. (2021). Plant disease detection using deep learning techniques: A review. In Advances in Intelligent Systems and Computing (Vol. 1331, pp. 134-143). Springer.

[6] Munda, S., Singh, P. K., & Mishra, S. (2021). Deep learning-based plant disease detection: A review. Journal of Plant Diseases and Protection, 128(2), 169- 187.

[7] Koirala, A., & Panday, S. (2021). A review of deep learning for plant disease detection. SN Computer Science, 2(1), 1-17.

[8] Khan, A. M., Shafique, S., & Awan, I. A. (2021). Plant disease classification using deep learning: A survey. Computers and Electronics in Agriculture, 186, 106074.

[9] Kaur, R., & Verma, M. (2021). A survey of plant disease detection using deep learning techniques. International Journal of Engineering and Advanced Technology, 10(4), 467-475.

[10] Acharya, D., Kumar, P., & Gautam, A. K. (2021). Identification of plant 61 diseases: A review on deep learning approach. International Journal of Intelligent Systems and Applications, 13(8), 1-11.

[11] Ferdous, S., Al Mamun, S., Arafat, Y., & Islam, M. R. (2020). Leaf disease detection of different crops: A deep learning approach. International Journal of Computer Applications, 179(44), 16-23.

[12] Ghosal, S., Kumar, A., & Bandyopadhyay, S. (2019). An overview of deep learning-based approaches for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 162, 219-232.

[13] Singh, D., & Jindal, M. (2019). Leaf disease detection and classification using deep learning: A review. Journal of Ambient Intelligence and Humanized Computing, 10(6), 2061-2084.

[14] Li, H., Guo, X., Liu, Q., & Zhang, Z. (2019). A novel deep learning method for plant disease detection based on Gabor wavelet and convolutional neural network. Frontiers in Plant Science, 10, 118.

[15] Wang, S., Zhang, X., & Zhang, J. (2019). A review of image classification algorithms for plant diseases. Plant Methods, 15(1), 1-14.

[16] Xie, F., Liu, Y., & Zou, X. (2019). A survey on deep learning-based plant disease recognition. Neurocomputing, 330, 208-221.

[17] Osman, M., Rahman, M. M., & Zhang, Y. (2018). Early plant disease detection using image processing and machine learning techniques. Journal of Advanced Research in Dynamical and Control Systems, 10(8), 219-226.

[18] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.

[19] You, Q., Yan, S., Zhang, H., & Tang, X. (2015). Robust deep learning for improved classification of traffic signs. IEEE Transactions on Neural Networks and Learning Systems, 27(8), 1738-1751.

[20] Sánchez, C., & Poggio, T. (2011). Deep learning: advances and challenges. A not-so-short review. arXiv preprint arXiv:1803.01164.

**APPENDIX**

**APPENDIX 1**

import numpy as np

import matplotlib.pyplot as plt

import cv2

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Flatten, Reshape, Conv2D, MaxPooling2D, BatchNormalization

from tensorflow.keras.preprocessing.image import img\_to\_array, load\_img

from tensorflow.keras.optimizers import Adam

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import seaborn as sns

# Function to preprocess image

def preprocess\_img(img\_path, target\_size=(128, 128)):

my\_image = load\_img(img\_path, target\_size=target\_size)

my\_image\_array = img\_to\_array(my\_image) / 255.0

return my\_image\_array

# Function for prediction and visualization

def prediction\_cnn(model, img\_path, class\_names):

my\_image\_array = preprocess\_img(img\_path)

my\_image\_array = np.expand\_dims(my\_image\_array, axis=0)

# Make predictions

out = np.round(model.predict(my\_image\_array)[0], 2)

# Create subplots with adjusted spacing

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6), gridspec\_kw={'width\_ratios': [3, 1]})

# Plot image

img\_cv2 = cv2.imread(img\_path)

img\_rgb = cv2.cvtColor(img\_cv2, cv2.COLOR\_BGR2RGB)

ax1.imshow(img\_rgb)

ax1.axis('off') # Hide axis for the image

# Plot bar chart

bar\_positions = np.arange(len(class\_names))

ax2.barh(bar\_positions, out, color='lightgray', edgecolor='red', linewidth=1, height=0.5)

for i, v in enumerate(out):

ax2.text(v + 0.01, i, f"{100 \* v:.2f}%", color='black', va='center', fontweight='bold')

ax2.set\_xticks([])

ax2.set\_yticks(bar\_positions)

ax2.set\_yticklabels(class\_names, fontweight='bold', fontsize=14)

# Adjust spacing between subplots

fig.tight\_layout(pad=4.0)

plt.show()

# CNN Model for Image Classification

def create\_cnn\_model(input\_shape, num\_classes):

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=input\_shape),

BatchNormalization(),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

BatchNormalization(),

MaxPooling2D((2, 2)),

Conv2D(128, (3, 3), activation='relu'),

BatchNormalization(),

MaxPooling2D((2, 2)),

Flatten(),

Dense(256, activation='relu'),

BatchNormalization(),

Dense(num\_classes, activation='softmax')

])

model.compile(loss='categorical\_crossentropy', optimizer=Adam(lr=0.0001), metrics=['accuracy'])

return model

# Function to evaluate model accuracy

def evaluate\_model\_accuracy(model, test\_dataset):

y\_true = []

y\_pred = []

for image\_batch, label\_batch in test\_dataset:

y\_true.extend(label\_batch.numpy())

preds = model.predict(image\_batch)

y\_pred.extend(np.argmax(preds, axis=-1))

acc = accuracy\_score(y\_true, y\_pred)

return acc

# Assuming you have a test dataset with labeled images

def evaluate\_cnn\_model(model, test\_dataset, class\_names):

y\_true = []

y\_pred = []

for image\_batch, label\_batch in test\_dataset:

y\_true.extend(label\_batch.numpy())

preds = model.predict(image\_batch)

y\_pred.extend(np.argmax(preds, axis=-1))

# Evaluate accuracy

acc = accuracy\_score(y\_true, y\_pred)

print(f"Model Accuracy: {acc:.2f}")

# Confusion matrix

cm = confusion\_matrix(y\_true, y\_pred, normalize='true')

sns.heatmap(cm, annot=True, cmap='viridis', cbar=None)

plt.title("Confusion matrix", fontweight='bold')

plt.ylabel("True", fontsize=14)

plt.xlabel("Predicted", fontsize=14)

plt.show()

# Classification report

print(classification\_report(y\_true, y\_pred, target\_names=class\_names))

# Assuming you have a test dataset

test\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

'test\_data',

image\_size=(128, 128),

batch\_size=32

)

# Assuming class names are ['Early\_blight', 'Healthy', 'Late\_blight']

class\_names = test\_ds.class\_names

num\_classes = len(class\_names)

# Create and train the CNN model

input\_shape = (128, 128, 3)

cnn\_model = create\_cnn\_model(input\_shape, num\_classes)

cnn\_model.summary() # Display model summary

# Train the model (replace 'train\_ds' with your training dataset)

history = cnn\_model.fit(train\_ds, epochs=20, validation\_data=val\_ds)

# Evaluate the CNN model

evaluate\_cnn\_model(cnn\_model, test\_ds, class\_names)

# Save the CNN model

cnn\_model.save("cnn\_model.h5")

# Call the prediction function with the appropriate image paths

img\_path1 = "path/to/image1.jpg"

img\_path2 = "path/to/image2.jpg"

img\_path3 = "path/to/image3.jpg"

prediction\_cnn(cnn\_model, img\_path1, class\_names)

prediction\_cnn(cnn\_model, img\_path2, class\_names)

prediction\_cnn(cnn\_model, img\_path3, class\_names)

import numpy as np

import matplotlib.pyplot as plt

import cv2

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Flatten, Reshape

from tensorflow.keras.preprocessing.image import img\_to\_array, load\_img

# Define the image dimensions

img\_height, img\_width = 128, 128

# Define the class names

categories = ['Early\_blight', 'Healthy', 'Late\_blight']

# Define LSTM model

lstm\_model = Sequential()

lstm\_model.add(Reshape((img\_height, img\_width \* 3), input\_shape=(img\_height, img\_width, 3)))

lstm\_model.add(LSTM(100))

lstm\_model.add(Dense(len(categories), activation='softmax'))

lstm\_model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Function to preprocess an image

def preprocess\_image(img\_path):

image = load\_img(img\_path, target\_size=(img\_height, img\_width))

image\_array = img\_to\_array(image)

return image\_array

# Function for prediction and visualization

def predict\_and\_visualize\_lstm(img\_path):

image\_array = preprocess\_image(img\_path)

image\_array = np.expand\_dims(image\_array, axis=0)

# Make predictions

predictions = np.round(lstm\_model.predict(image\_array)[0], 2)

# Create subplots with adjusted spacing

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6), gridspec\_kw={'width\_ratios': [3, 1]})

# Plot image with LSD visualization

image\_cv2 = cv2.imread(img\_path)

gray = cv2.cvtColor(image\_cv2, cv2.COLOR\_BGR2GRAY)

lsd = cv2.createLineSegmentDetector(0)

lines = lsd.detect(gray)[0]

drawn\_image = lsd.drawSegments(image\_cv2, lines)

ax1.imshow(cv2.cvtColor(drawn\_image, cv2.COLOR\_BGR2RGB))

ax1.axis('off') # Hide axis for the image

# Plot bar chart

bar\_positions = np.arange(len(categories))

ax2.barh(bar\_positions, predictions, color='lightgray', edgecolor='red', linewidth=1, height=0.5)

for i, v in enumerate(predictions):

ax2.text(v + 0.01, i, f"{100 \* v:.2f}%", color='black', va='center', fontweight='bold')

ax2.set\_xticks([])

ax2.set\_yticks(bar\_positions)

ax2.set\_yticklabels(categories, fontweight='bold', fontsize=14)

# Adjust spacing between subplots

fig.tight\_layout(pad=4.0)

fig.savefig('predicted\_image.png', bbox\_inches='tight')

return plt.show()

# Example usage with appropriate image paths

image\_path1 = r"path\to\your\image1.jpg"

image\_path2 = r"path\to\your\image2.jpg"

image\_path3 = r"path\to\your\image3.jpg"

predict\_and\_visualize\_lstm(image\_path1)

predict\_and\_visualize\_lstm(image\_path2)

predict\_and\_visualize\_lstm(image\_path3)

# Assuming you have a test dataset with labeled images

test\_image\_paths = [image\_path1, image\_path2, image\_path3]

test\_labels = [1, 0, 2] # Assuming the labels are 0, 1, 2 for ['Early\_blight', 'Healthy', 'Late\_blight']

# Initialize an empty list to store the predictions

predictions = []

# Iterate through the test images and make predictions

for img\_path in test\_image\_paths:

image\_array = preprocess\_image(img\_path)

image\_array = np.expand\_dims(image\_array, axis=0)

predicted\_output = np.round(lstm\_model.predict(image\_array)[0], 2)

predicted\_label = np.argmax(predicted\_output)

predictions.append(predicted\_label)

# Compare the predicted labels with the true labels and compute the accuracy

correct\_predictions = sum([1 for i in range(len(test\_labels)) if test\_labels[i] == predictions[i]])

accuracy = correct\_predictions / len(test\_labels)

print(f"Accuracy of the LSTM model: {accuracy:.2f}")

import numpy as np

import matplotlib.pyplot as plt

import cv2

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization

from tensorflow.keras.preprocessing.image import img\_to\_array, load\_img

from tensorflow.keras.optimizers import Adam

# Define the image dimensions

img\_height, img\_width = 128, 128

# Define the class names

categories = ['Early\_blight', 'Healthy', 'Late\_blight']

# Define SCNN model

scnn\_model = Sequential()

scnn\_model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_height, img\_width, 3)))

scnn\_model.add(BatchNormalization())

scnn\_model.add(MaxPooling2D((2, 2)))

scnn\_model.add(Conv2D(64, (3, 3), activation='relu'))

scnn\_model.add(BatchNormalization())

scnn\_model.add(MaxPooling2D((2, 2)))

scnn\_model.add(Conv2D(128, (3, 3), activation='relu'))

scnn\_model.add(BatchNormalization())

scnn\_model.add(MaxPooling2D((2, 2)))

scnn\_model.add(Flatten())

scnn\_model.add(Dense(256, activation='relu'))

scnn\_model.add(BatchNormalization())

scnn\_model.add(Dense(len(categories), activation='softmax'))

# Compile the model

scnn\_model.compile(loss='categorical\_crossentropy', optimizer=Adam(lr=0.0001), metrics=['accuracy'])

# Function to preprocess image

def preprocess\_image(img\_path):

my\_image = load\_img(img\_path, target\_size=(img\_height, img\_width))

my\_image\_array = img\_to\_array(my\_image) / 255.0 # Normalize pixel values to [0, 1]

return my\_image\_array

# Function for prediction and visualization

def predict\_and\_visualize\_scnn(img\_path):

my\_image\_array = preprocess\_image(img\_path)

my\_image\_array = np.expand\_dims(my\_image\_array, axis=0)

# Make predictions

predictions = np.round(scnn\_model.predict(my\_image\_array)[0], 2)

# Create subplots with adjusted spacing

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6), gridspec\_kw={'width\_ratios': [3, 1]})

# Plot image

img\_cv2 = cv2.imread(img\_path)

img\_rgb = cv2.cvtColor(img\_cv2, cv2.COLOR\_BGR2RGB)

ax1.imshow(img\_rgb)

ax1.axis('off') # Hide axis for the image

# Plot bar chart

bar\_positions = np.arange(len(categories))

ax2.barh(bar\_positions, predictions, color='lightgray', edgecolor='red', linewidth=1, height=0.5)

for i, v in enumerate(predictions):

ax2.text(v + 0.01, i, f"{100 \* v:.2f}%", color='black', va='center', fontweight='bold')

ax2.set\_xticks([])

ax2.set\_yticks(bar\_positions)

ax2.set\_yticklabels(categories, fontweight='bold', fontsize=14)

# Adjust spacing between subplots

fig.tight\_layout(pad=4.0)

plt.show()

# Example usage with appropriate image paths

img1 = r"C:\Users\gadda\data\Healthy\04481ca2-f94c-457e-b785-1ac05800b7ec\_\_\_RS\_HL 1930.JPG"

img2 = r"C:\Users\gadda\data\Early\_blight\0182e991-97f0-4805-a1f7-6e1b4306d518\_\_\_RS\_Early.B 7015.JPG"

img3 = r"C:\Users\gadda\data\Late\_blight\01270f5c-a44b-4da7-9398-289088c197ab\_\_\_RS\_LB 2517.JPG"

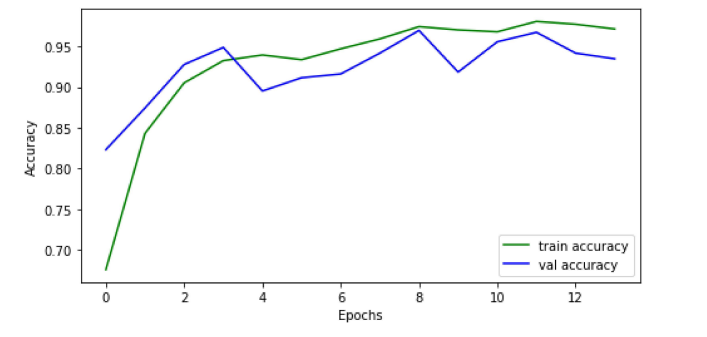
# Make predictions for the example images

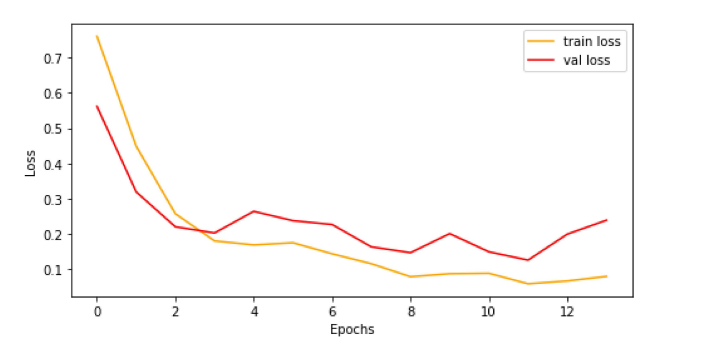
predict\_and\_visualize\_scnn(img1)

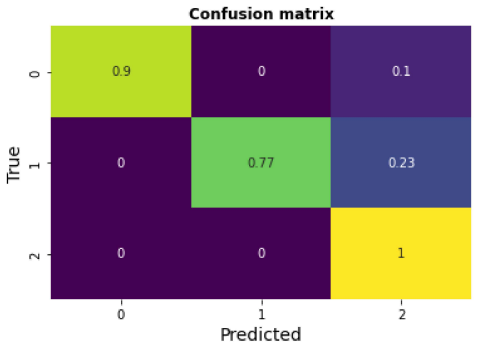
predict\_and\_visualize\_scnn(img2)

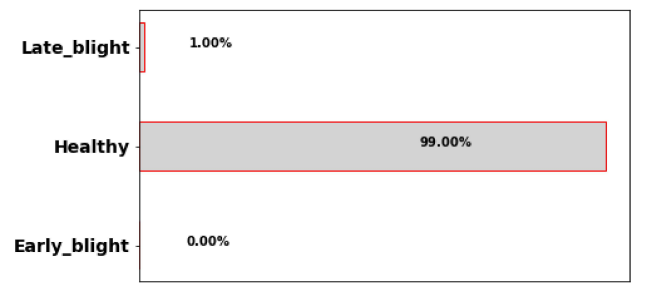
predict\_and\_visualize\_scnn(img3)

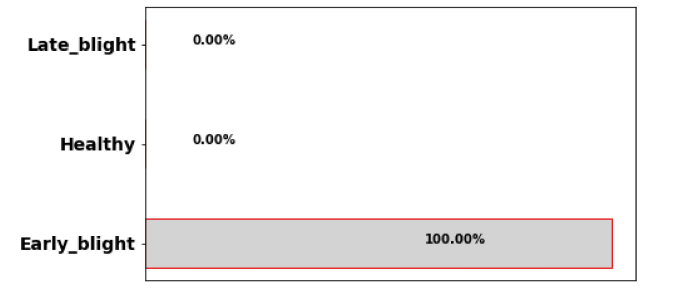
**APPENDIX 2**

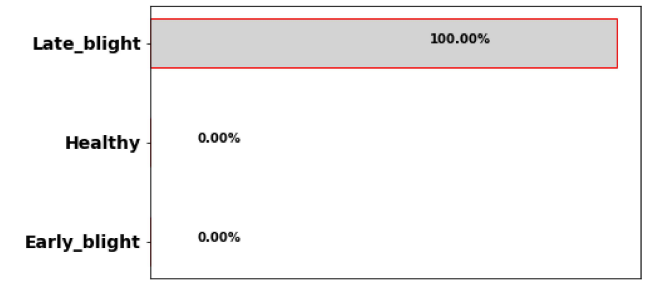
****

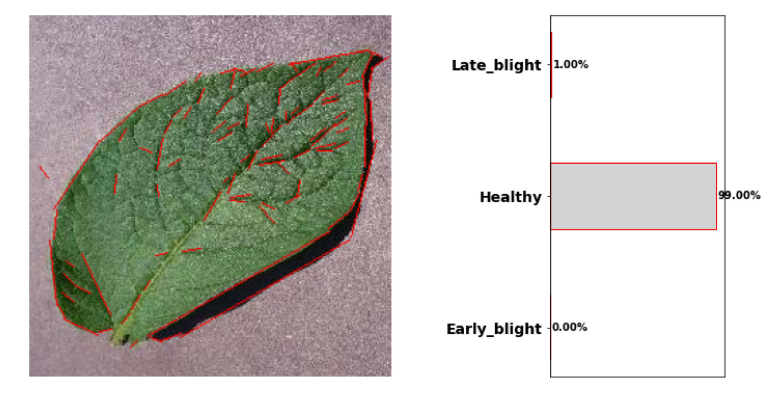


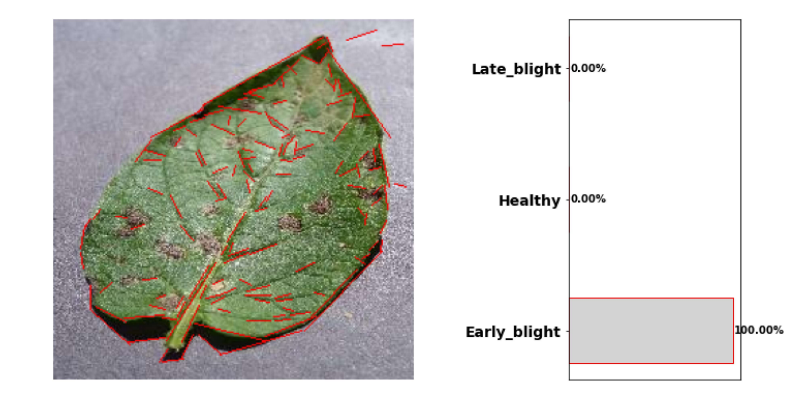


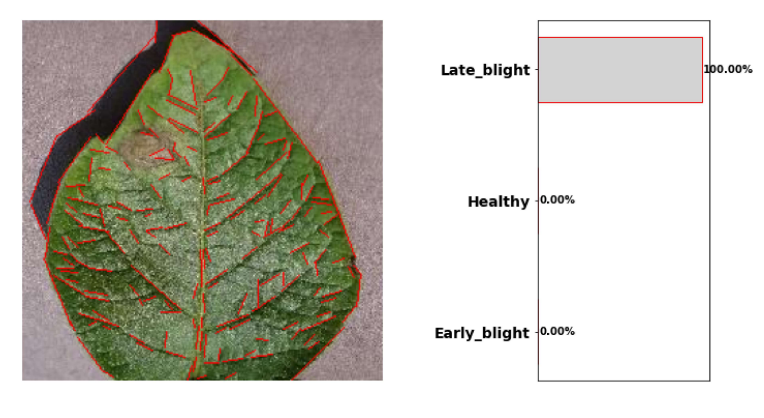


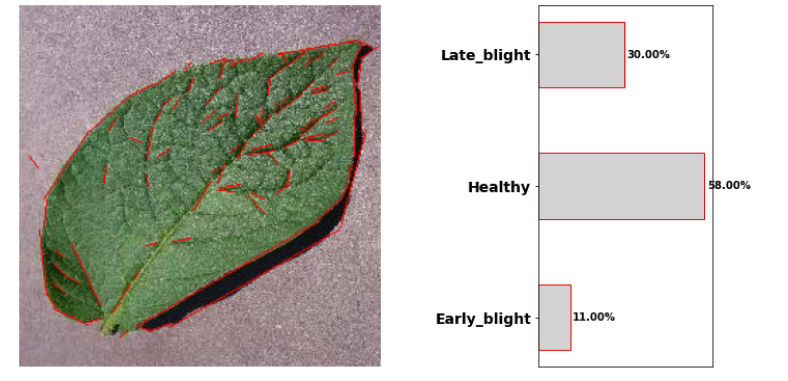
****

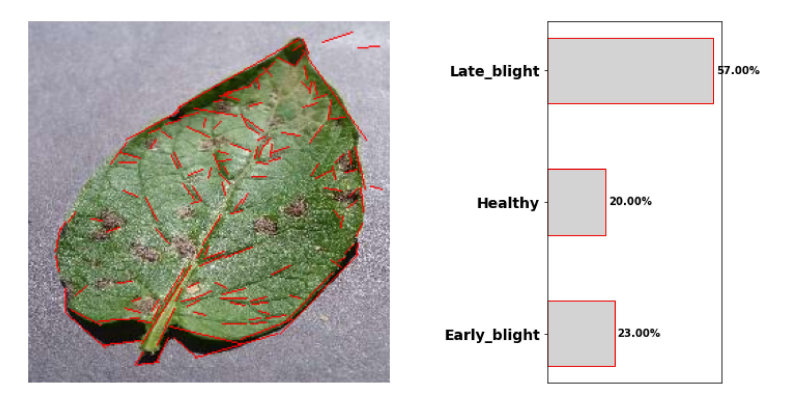
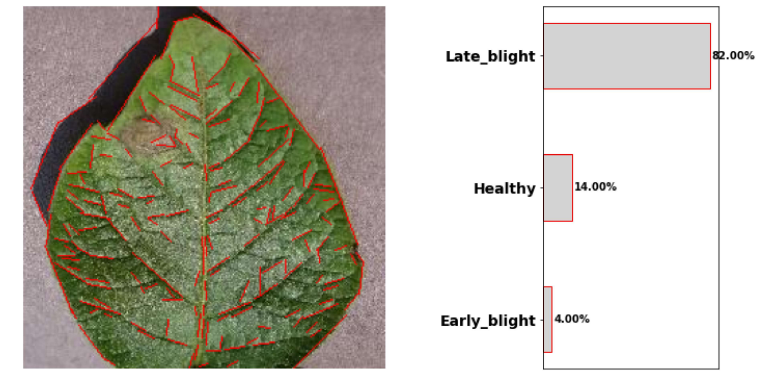
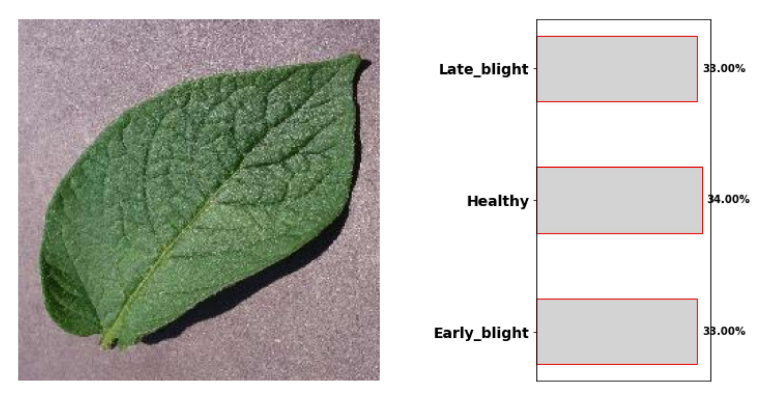
****

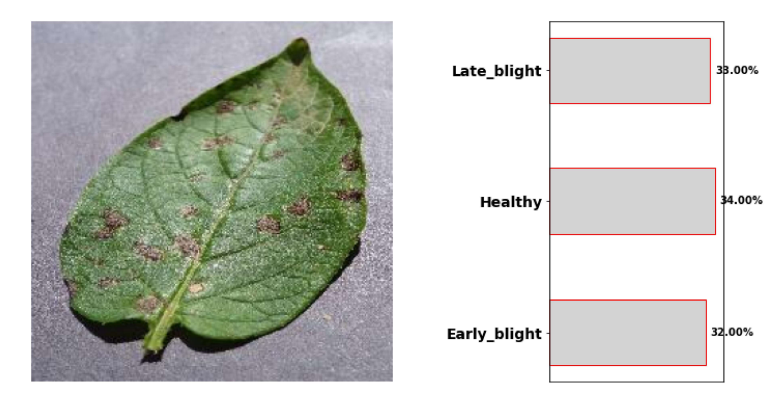


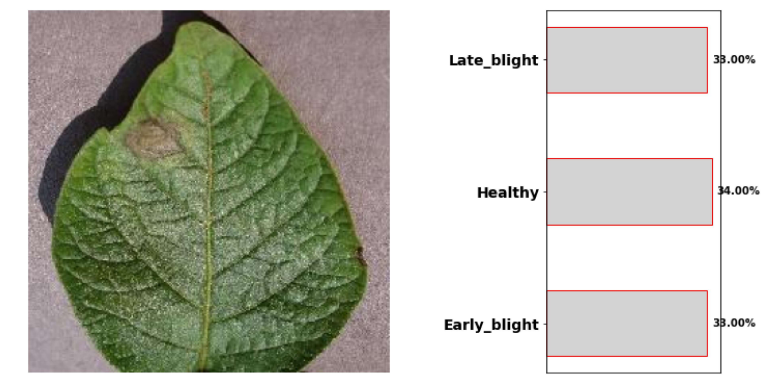






****



****

**PAPER PUBLICATION STATUS**

The publication process not yet started