“Early Disease Detection and Classification of Tomato Plant”

**Minor Project Report Submitted**

**To**

**Chhattisgarh Swami Vivekananda**

**Technical University, Bhilai (C.G.), India**

****

*for*

*The award of the degree*

*of*

**BACHELOR OF TECHNOLOGY(Hons.)**

*In*

**COMPUTER SCIENCE & ENGINEERING**

**(Artificial Intelligence / Data Science)**

**By**

**Sagar Agarwal**

**B.Tech. 4th Semester**

**Roll No. 503310920004**

**Enrollment No. BK5276**

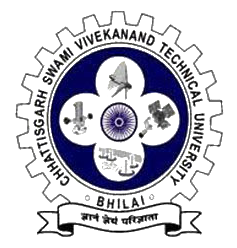
Under the Guidance of

**Prof Name**

Professor Designation

Computer Science and Engineering

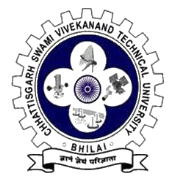
UTD,CSVTU BHILAI (C.G.)



­­­­­­­­­­­­­­­­­**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**UTD, CSVTU BHILAI (C.G.)**

**Session 2022-2023**

­­­­­­­­­­­­­­­­­ **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**UTD, CSVTU,BHILAI**

**RAIPUR (C.G.)**

**DECLARATION BY THE CANDIDATE**

I, the undersigned solemnly declare that the thesis entitled **“Early Disease Detection & Classification of Tomato Plant”** is based on my work carried out during the course of my study under the supervision of **Dr. J.P Patra,** Head of the Department of Computer Science and Engineering, Shri Shankaracharya Institute of Professional Management & Technology, Raipur (C.G.), India.

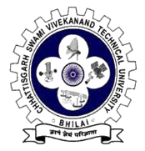
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sagar Agarwal

**Roll No. 503310920004**

**Enrollment No. BK5276**

I

­­­­­­­­­­­­­­­­­ **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**UTD, CSVTU,BHILAI**

**RAIPUR (C.G.)**

CERTIFICATE OF THE SUPERVISOR

This is to certify that the incorporation in the thesis “**Early Disease Detection & Classification of Tomato Plant”** is a record of research work carried out by Sagar Agarwal, bearing Roll No. 503310920004Enrollment No. BK5276 guidance and supervision for the award of the degree of Master of Technology in Computer Science & Engineering (Artificial Intelligence & Machine Learning) of Chhattisgarh Swami Vivekananda Technical University, Bhilai (C.G.) India.

To the best of my knowledge and belief the thesis

1. Embodies the work of the candidate himself,
2. Has duly been completed in the specified time,
3. Fulfil the requirement of the Ordinance relating to the B.Tech.. degree of the University and
4. Is up to the desired standard both in respect of contents and language for being referred to the examiners.

(Signature of H.O.D.) (Signature of Supervisor)

Dr. Dr.

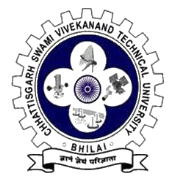
Professor & HOD Professor & HOD

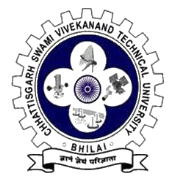
Department of CSE Department of CSE

Forwarded to Chhattisgarh Swami Vivekananda Technical University, Bhilai (C.G.)

………………………………………….

(Signature of the Principal)

 II

­­­­­­­­­­­­­­­­­ **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**UTD, CSVTU,BHILAI**

**RAIPUR (C.G.)**

CERTIFICATE BY THE EXAMINERS

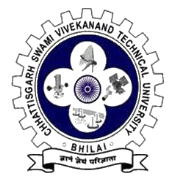
This is to certify that the project thesis entitled “**Early Disease Detection & Classification of Tomato Plant”** wassubmitted by Sagar Agarwal student of M. Tech. (CSE) (Roll No. 503310920004, Enrollment No. BK5276) has been examined by the undersigned as a part of the examination and is hereby recommended for the award of the degree of **Master of Technology in Computer Science and Engineering** (Artificial Intelligence & Machine Learning)of Chhattisgarh Swami Vivekananda Technical University, Bhilai (C.G.) India.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

Date: Date:

III

­­­­­­­­­­­­­­­­­ **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**UTD, CSVTU,BHILAI**

**RAIPUR (C.G.)**

ACKNOWLEDGEMENT

The real spirit of achieving a goal is through the way of excellence and serious discipline. I want to thank SSIPMT, Raipur for providing me with the necessary software, tools, and other resources to deliver my major project work.

With gratitude and humanity, I acknowledge my indebtedness to Dr. J.P. Patra, Professor & HOD, SSIPMT, Raipur, under whose guidance I had the privilege to complete this project work. Also, I am grateful to all the faculty members of the department of CSE, who were always there at the need of the hour and provided me with all the help and facility, I required for the completion of my project work.

I shall be failing in my duties if I do not express my duty sense of gratitude towards Mr. Anand Tamrakar, M.Tech Coordinator of the Department of Computer Science and Engineering SSIPMT, Raipur.I owe my sincere thanks to Shri Nishant Tripathi Chairman (B.G.) SSIPMT, Raipur, Dr. Alok Kumar Jain, Principal of SSIPMT, Raipur, for inspiration and constant encouragement that enabled me to present my work in this form.

My greatest thanks go to my parents and family, who have been my driving force. My work would not be possible without their constant inspiration, encouragement, support, and love. Above all, I render my gratitude to the almighty, who bestowed self-confidence, Ability, and strength on me to complete this work.

Sagar Agarwal

**Roll No. 503310920004**

**Enrollment No. BK5276**

IV

ABSTRACT

Tomato is a highly valuable crop in the Indian market, cultivated in large quantities. However, tomato plants are susceptible to various diseases, which can lead to significant reductions or complete destruction of the crop, resulting in substantial economic losses for farmers. Therefore, the early and accurate detection of these diseases is crucial to minimize such losses. In recent years, numerous methods have been proposed to detect plant diseases, leveraging advancements in technology. Despite these efforts, achieving high accuracy in detecting tomato plant diseases remains a challenge. In this paper, we present a comprehensive study where we explore the effectiveness of deep learning models in detecting and classifying diseases in tomato plants. Specifically, we investigate four different pre-trained deep learning models, namely VGG-19, ResNet-50, Inception V3, and InceptionResNetV2. These models are trained using three different optimizers and various learning rates. By leveraging these models, we aim to develop an efficient framework that can accurately identify and classify diseases in tomato plants, thereby reducing production and economic losses. The proposed framework holds significant potential in revolutionizing the early detection and classification of tomato plant diseases. The utilization of deep learning models and optimization techniques contributes to improving the accuracy and efficiency of disease detection systems.

This research aims to provide valuable insights for farmers and stakeholders in the agricultural industry, enabling them to take timely and appropriate actions to mitigate the impact of diseases on tomato crop yields.

V

Table of contents

|  |  |  |
| --- | --- | --- |
| Chapter | Title | Page No. |
| I | Introduction   * 1. Overview   2. Thesis Goals and Objective   1.3 Organization of Thesis | 1-3 |
| II | Literature Review  2.1 Paper reviewed  2.2 Summary | 4-6 |
| III | Problem Identification | 7 |
| IV | Methodology | 8-37 |
| V | Implementation | 38-40 |
| VI | Results and Discussion | 41-44 |
| VII | Conclusion and Future Scope | 45-46 |
| VIII | References | 47-48 |

VI

List of Figures

|  |  |  |
| --- | --- | --- |
| **Figure** | **Title of Figure** | **Page No.** |
| figure 4.1 | **Septoria Leaf Spot** | 8 |
| figure 4.2 | **Early Blight** | 8 |
| figure 4.3 | Tomato Leaf Mold | 8 |
| figure 4.4 | Spider Mites | 8 |
| figure 4.5 | Healthy Tomato Leaf | 9 |
| figure 4.6 | Process of the training dataset | 19 |
| figure 4.7 | Architecture of Convolutional Neural Network | 22 |
| Figure 4.8 | Average & Max Pooling | 24 |
| figure 4.9 | Network Architecture of VGG-19 Model | 28 |
| figure 4.10 | ResNet 50 Model Architecture | 30 |
| figure 4.11 | Inception ResNet V2 Network | 34 |
| figure 4.12 | Workflow diagram of the model presented | 37 |
| Figure 5.1 | Webpage with implemented model | 39 |
| Figure 5.2 | Webpage with result | 40 |

VII

|  |  |  |
| --- | --- | --- |
| **Figure** | **Title of Figure** | **Page No.** |
| figure 6.1 | **Chart for Optimizers vs Learning Rate on VGG-19 model** | 42 |
| figure 6.2 | **InceptionResNet V2 training history** | 43 |
| figure 6.3 | InceptionResNetV2 Accuracy history | 44 |

VIII

List of Table

|  |  |  |
| --- | --- | --- |
| **Table** | **Title of Table** | **Page No.** |
| Table 2.1 | Literature Survey | 5-6 |
| Table 6.1 | F1 score of different models | 43 |

IX

**List of Abbreviations**

1. **CNN** - Convolution Neural Network

2. **VGG** - Visual Geometry Group

3. **ResNet** - Residual Neural Network

4. **MT** - Metric Tonnes

5. **SVM** - Support Vector Machine

6. **DNN** - Deep Neural Network

7. **RGB** - Red, Green & Blue

8. **SGD** - Stochastic Gradient Descent

X

CHAPTER – I

Introduction

Chapter – I:

Introduction

* 1. **Overview**

Agriculture plays a vital role in the Indian economy, with a significant portion of the population depending on it for their livelihood. Among the various crops cultivated in India, tomatoes hold immense importance as they are widely consumed and produced across the country. In fact, India ranks as the second-largest producer of tomatoes globally, with an annual production of 18,399,000 tonnes in 2021. In the state of Chhattisgarh, tomato production reached 1149 MT, securing the eighth position in terms of output. However, tomato farmers often encounter substantial economic losses due to various diseases that affect the crop. Early detection and timely treatment of these diseases are crucial in order to prevent significant losses and preserve the health of the plants.

This project aims to address this issue by developing a disease detection system capable of identifying diseases in tomato plants based on the symptoms exhibited on their leaves. The methodology employed in this study focuses on the most common diseases affecting tomato plants, including Leaf Mold, Early Blight, Spider Mites (Two-Spotted Spider Mites), Tomato Healthy, and Septoria Leaf Spot. Accurate prediction of these diseases is essential as each disease requires a specific remedy, and the application of an incorrect treatment can have adverse effects on the plant.

Human observation alone is often insufficient for accurately detecting these diseases, as it can be challenging to discern the symptoms with a single glance. This can result in incorrect assumptions about the disease, leading to the use of inappropriate treatments and, ultimately, the loss of the entire plant. Therefore, it is imperative to develop a robust model that can predict diseases in tomato plants with the utmost accuracy.

The proposed model aims to leverage advanced techniques, such as deep learning, to accurately identify and classify diseases based on visual cues. By employing such a model, farmers will be equipped with a reliable tool that can aid in the early detection of diseases, enabling timely interventions and appropriate treatments. Ultimately, this research endeavors to provide a practical solution that empowers tomato farmers to mitigate losses and ensure the health and productivity of their crops.

To achieve this, the first step involves the collection of a comprehensive and accurately labeled dataset. Field surveys were carried out across four different locations in Chhattisgarh, namely Raipur, Raweli, Parsada, and Julum. At each location, images of tomato plants displaying symptoms related to the target diseases were collected. Additionally, images of healthy tomato plants were captured to serve as a reference class. High-resolution cameras and appropriate lighting conditions were used to ensure optimal image quality.

To validate the accuracy of the dataset, collaboration with the agriculture department of Chhattisgarh was established. They provided expert guidance and verified the presence of diseases in a subset of the collected images. This validation process ensures that the dataset is reliable and representative of real-world conditions.

Next, the collected dataset undergoes preprocessing to enhance the quality and consistency of the images. Image resizing, normalization, and noise reduction techniques are applied to ensure that the images are suitable for training and testing the deep learning model.

The dataset is then divided into training, validation, and testing sets. The training set is used to train the deep learning model, while the validation set helps fine-tune the model's parameters and evaluate its performance. The testing set is used to assess the final performance of the trained model on unseen data.

The training process involves feeding the labeled images into a Convolutional Neural Network (CNN) model. CNNs have demonstrated remarkable performance in image classification tasks due to their ability to recognize distinct patterns and features. By training the model on a comprehensive dataset, it can learn to identify the visual symptoms associated with different diseases, allowing for accurate disease classification.

To evaluate the performance of the developed model, various metrics such as accuracy, precision, recall, and F1 score are employed. These metrics provide insights into the model's ability to correctly classify the diseases and distinguish them from healthy plants. By analyzing these metrics, the effectiveness of the disease detection system can be assessed, and any necessary improvements can be identified.

Once the model has been trained and evaluated, it can be deployed as a user-friendly application or a web-based interface. This allows farmers to easily capture images of their tomato plants and obtain immediate predictions regarding the presence of diseases. The application provides valuable information to farmers, enabling them to take proactive measures and apply appropriate treatments at the early stages of disease development.

In conclusion, this project aims to develop a disease detection system for tomato plants using deep learning techniques. By accurately identifying and classifying diseases based on visual cues, the system can assist farmers in detecting diseases early and implementing timely treatments. The research is based on a comprehensive dataset, carefully collected and validated, to ensure accurate training and testing of the model. The success of this project will contribute to the overall health and productivity of tomato crops, enabling farmers to mitigate losses and sustain their livelihoods.

* 1. **Thesis Goals and Objectives**

The objective is to present and provide a model which can detect the disease of the Tomato plant so that it helps the farmer to take the possible steps to protect the plant and hence save the crop.

**1.3** **Organization of Thesis**

The rest of the thesis has been organized into four chapters. Following is a brief description of each chapter:

**Chapter 2. Review of Related Work**

This chapter deals with the survey on the methodologies adopted for the association of documents using a headword extraction algorithm. This chapter also deals with a brief analysis of the tools used for the enhancement, segmentation design development, and geometric transformation.

**Chapter 3. Problem Identification**

This chapter deals with the identification of the problem due to which we reached the solution and thought that this project helped in resolving the problem to an extinct.

**Chapter 4. Proposed Methodology**

This chapter deals with the methodology and techniques used in building the project with a proper workflow diagram.

**Chapter 5. Implementation**

In this chapter, we have explained the implantation part and also shown copies of the result as given by the model.

**Chapter 6. Result & Discussion**

Here we mentioned the result and gave a brief discussion on we are solving the problem with result accuracies.

**Chapter 7. Conclusion & Future Scope**

This chapter deals with the conclusion of whether the problem is actually resolved or not and how much we can improve it further and also adds the future scope of what we can add to enhance its performance.

CHAPTER – II

LITERATURE REVIEW

Chapter II:

literature review

* 1. **Paper Reviewed**

One of the papers examined the effectiveness of pre-trained deep neural networks for predicting tomato leaf diseases. Four pre-trained models (VGG19, ResNet50, InceptionV3, and InceptionResNet V2) are compared with a custom-built model, and their performance is evaluated using accuracy, precision, recall, and F1 score. The findings indicate that pre-trained models perform well and can outperform custom-built models for this task [1]. In another paper, they proposed a comparison between 3 pre-trained models SVM & k-nearest with each model tested with 2 feature extractions - Inception V3, VGG - 16, VGG- 19. This model gave an accuracy of 96.8. The accuracy can be improved and work can be easily extended to identify the diseases for other crops as well [2]. The paper proposes a novel approach for tomato leaf disease classification using transfer learning and feature concatenation. The authors employ pre-trained deep neural networks, including VGG16 and ResNet50, to extract relevant features from input images. These features are then concatenated and used to train a linear support vector machine (SVM) classifier for disease classification [3]. The result confirmed that DNN deep learning classifier gives an improved accuracy of 86.18%. The accuracy can be easily improved [4]. They used a deep neural network model for detecting and classifying tomato plant leaf diseases into predefined categories and the proposed model was compared to VGG and ResNet versions. It achieved an accuracy rate of 98.43%. The dataset on which it was performed was less so we need to gather more data to test it over the same [5].

After reviewing the mentioned papers, it can be concluded that pre-trained deep neural networks show promising results in predicting tomato leaf diseases. The comparative analysis of different pre-trained models, including VGG19, ResNet50, InceptionV3, and InceptionResNet V2, demonstrates their superior performance compared to custom-built models. Another study explores the accuracy of three pre-trained models, SVM, and k-nearest neighbours, using feature extraction techniques like Inception V3, VGG-16, and VGG-19, achieving an impressive accuracy of 96.8%. The potential for further accuracy improvements and extending the approach to other crops is highlighted.

* 1. **Summary**

In the present chapter, the literature survey made so far related to the work has been briefly discussed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S**.**no** | **Year** | **Paper Title** | **Journals** | **Research Finding** |
| 1. | 2023 | Impact of PreTrained Deep Neural Networks for Tomato Leaf Disease Prediction | JECE | The paper examines the effectiveness of pre-trained deep neural networks for predicting tomato leaf diseases. Four pre-trained models (VGG19, ResNet50, InceptionV3, and Xception) are compared with a custom-built model, and their performance is evaluated using accuracy, precision, recall, and F1 score. The findings indicate that pre-trained models perform well and can outperform custom-built models for this task. |
| 2. | 2021 | Tomato leaf disease classification by exploiting transfer learning and feature concatenation | IJSRET | The paper proposes a novel approach for tomato leaf disease classification using transfer learning and feature concatenation. The authors employ pre-trained deep neural networks, including VGG16 and ResNet50, to extract relevant features from input images. These features are then concatenated and used to train a linear support vector machine (SVM) classifier for disease classification. |
| 3. | 2021 | Managing The Tomato Leaf Disease Detection Accuracy Using Computer Vision based Deep Neural Network. | Journal of Contemporary Issues in Business and Government | The result confirmed that DNN deep learning classifier gives an improved accuracy of 86.18%. The accuracy can be easily improved. |
| 4. | 2021 | Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network | MDPI | They used a deep neural network model for detecting and classifying tomato plant leaf diseases into predefined categories and the proposed model was compared to VGG and ResNet versions. It achieved an accuracy rate of 98.43%. The dataset on which it was performed was less so we need to gather more data to test it over the same. |
| 5. | 2019 | An Integrated Deep Learning Framework for Tomato Leaf Disease Detection | IJITEE | They proposed a comparison between 3 pre-trained models SVM & k-nearest with each model tested with 2 feature extractions - Inception V3, VGG - 16, VGG- 19. This model gave an accuracy of 96.8 %. The accuracy can be improved and work can be easily extended to identify the diseases for other crops as well. |

**CHAPTER – III**

**PROBLEM IDENTIFICATION**

**Chapter III:**

**PROBLEM IDENTIFICATION**

* The problem at hand is the lack of an efficient and reliable system for early disease detection and classification in tomato plants. The existing methods for disease identification primarily rely on manual inspection, which is labor-intensive, inefficient, and often leads to delayed diagnosis. This delay in disease identification hampers the implementation of appropriate control measures, resulting in reduced crop productivity and economic losses for farmers.
* Furthermore, traditional visual inspection methods heavily depend on the expertise of human inspectors, making the process subjective and prone to misclassification. This subjectivity introduces inconsistency and hinders the accurate assessment of disease severity, leading to suboptimal decision-making in disease management practices.

**CHAPTER – IV**

**PROPOSED** **METHODOLOGY**

**Chapter IV:**

**METHODOLOGY**

**Dataset description**

The selected dataset for training in this study is the "Plant Village Tomato Leaf Dataset" obtained from Kaggle. This dataset comprises a total of 14,529 images, each belonging to one of five different classes of tomato leaf diseases. The dataset distribution includes 1,702 images of Tomato Septoria Leaf Spot, 800 images of early blight disease, 761 images of Tomato Leaf Mold, 1,341 images of spider mites, and 1,273 healthy tomato leaf images. The images in the dataset are of size 256 x 256 pixels and are in the RGB color space.



Figure 4.1: Septoria Leaf Spot Figure 4.2: Early Blight



Figure 4.3: Tomato Leaf Mold Figure 4.4: Spider Mites



Figure 4.5: Healthy Tomato leaf

For the evaluation phase, an additional dataset was collected from four different locations: Raipur, Raweli, Parsada, and Julum. The dataset consists of 2,950 images, covering the five different disease classes. This collection of images from different regions adds diversity and real-world variation to the evaluation process, enabling a more comprehensive assessment of the model's performance.

The choice of the Plant Village Tomato Leaf Dataset for training provides a diverse and substantial collection of images for building robust disease detection models. The inclusion of multiple disease classes, along with healthy leaf images, ensures that the model can accurately differentiate between various diseases and healthy plant foliage.

The size of the images used for training and evaluation, 256 x 256 pixels, strikes a balance between computational efficiency and preserving relevant details for disease identification. By maintaining the RGB color space, the model can consider color variations and patterns specific to each disease, improving its overall accuracy.

The incorporation of an evaluation dataset collected from different locations enhances the generalization capability of the trained model. By including images from various regions, the model becomes more adaptable to different environmental conditions and potential variations in disease manifestation. This aspect contributes to the reliability and practicality of the model when deployed in real-world scenarios.

In conclusion, the utilization of the Plant Village Tomato Leaf Dataset for training, along with the additional evaluation dataset from different locations, forms a solid foundation for developing accurate tomato leaf disease detection models. The abundance of images representing various disease classes and healthy leaves ensures comprehensive training, while the evaluation dataset provides real-world validation. These factors contribute to the overall effectiveness and applicability of the proposed deep learning frameworks and transfer learning approaches discussed in the reviewed papers.

**Septoria Leaf spot -**

Septoria leaf spot, caused by the fungal pathogen Septoria lycopersici, is a common disease that affects tomato plants. Here's more information about the disease, its causes, and prevention measures:

Cause: Septoria leaf spot is primarily caused by the fungus Septoria lycopersici, which overwinters in infected plant debris, such as fallen leaves or stems, and in the soil. Spores are then splashed onto the lower leaves of tomato plants during rain or irrigation.

Symptoms: The disease initially appears as small, dark brown spots with a lighter center on the lower leaves. As the infection progresses, the spots enlarge and merge, forming irregularly shaped lesions. The infected leaves may turn yellow, wither, and eventually drop off.

Favorable Conditions: Septoria leaf spot thrives in warm and humid environments, particularly when temperatures range between 60-80°F (15-27°C). It spreads more rapidly during periods of frequent rainfall or irrigation.

Prevention:

1. Crop rotation: Avoid planting tomatoes in the same location for consecutive years, as this helps reduce the pathogen's presence in the soil.
2. Cleanliness: Remove and destroy any infected plant debris to prevent the overwintering of the fungus.
3. Proper spacing: Provide adequate spacing between tomato plants to improve air circulation and reduce humidity around the foliage.
4. Drip irrigation: Use drip irrigation or water at the base of the plants to avoid wetting the leaves. Moisture on the leaves creates an ideal environment for the disease to develop.
5. Mulching: Apply a layer of organic mulch around the base of the plants to prevent soil from splashing onto the foliage.
6. Fungicide application: In severe cases, or if the disease is consistently present, you can apply fungicides labeled for Septoria leaf spot. Follow the instructions and recommended application rates provided on the product label.

By implementing these preventive measures, you can minimize the risk of Septoria leaf spots in your tomato plants and help maintain their overall health and productivity.

**Early Blight –**

Early blight is a common fungal disease that affects tomato plants.

Cause: Early blight is caused by the fungal pathogen Alternaria solani. The fungus survives in infected plant debris and can also be present in the soil. Spores are spread by wind, water, and insects.

Symptoms: Early blight initially appears as small, dark brown lesions with concentric rings on the older leaves of tomato plants. As the disease progresses, the spots enlarge and develop a bullseye-like pattern. The leaves may turn yellow, wither, and drop prematurely. The infection can also affect stems, fruits, and even the main stem of the plant.

Favorable Conditions: Early blight thrives in warm and humid conditions, typically when temperatures range between 75-85°F (24-29°C) and relative humidity is high. Overhead irrigation, prolonged leaf wetness, and overcrowding of plants can contribute to disease development.

Prevention:

1. Crop rotation: Avoid planting tomatoes or related crops in the same location for at least three years to break the disease cycle.
2. Cleanliness: Remove and destroy infected plant debris to reduce the source of infection.
3. Proper spacing: Provide adequate spacing between tomato plants to improve air circulation and reduce humidity.
4. Watering: Avoid overhead watering and irrigate at the base of the plants to keep the foliage dry.
5. Mulching: Apply organic mulch around the base of plants to prevent soil splashing onto the leaves.
6. Fungicide application: If necessary, apply fungicides labeled for early blight according to the instructions on the product label. Fungicides are most effective when applied preventatively or at the early stages of the disease.

Early detection and prompt intervention are crucial for managing early blight. By implementing these preventive measures, you can help minimize the impact of early blight and protect your tomato plants from this fungal disease.

**Tomato Leaf Mold -**

Tomato leaf mold is a fungal disease that specifically affects tomato plants.

Cause: Tomato leaf mold is caused by the fungus Passalora fulva (formerly known as Cladosporium fulvum). It thrives in warm and humid conditions, typically when temperatures range between 68-77°F (20-25°C) with high relative humidity.

Symptoms: Tomato leaf mold primarily affects the foliage of the plant. The disease first appears as pale yellow or light green spots on the upper surface of the leaves. As it progresses, the spots develop a fuzzy, velvety texture and turn dark olive-green or brown. The lower leaf surface may show a purplish-brown discoloration. Infected leaves may wither, curl, and drop prematurely, leading to defoliation if left untreated. The fruit itself is not typically affected by leaf mold.

Favorable Conditions: High humidity, poor air circulation, and moderate temperatures are conducive to the development of tomato leaf mold. The disease is often more prevalent in greenhouse settings or during periods of high humidity, such as rainy seasons.

Prevention and Management:

1. Proper spacing: Provide adequate spacing between tomato plants to allow for better air circulation and reduce humidity around the foliage.
2. Watering: Avoid overhead watering, as it can create a moist environment that favors disease development. Water at the base of the plants instead.
3. Ventilation: Improve ventilation in greenhouses or enclosed growing areas to minimize humidity levels and promote airflow.
4. Pruning: Prune lower leaves and branches to increase airflow and reduce contact with soil that may harbor the fungus.
5. Fungicides: If necessary, apply fungicides labeled for leaf mold prevention or control. Fungicides are most effective when used as a preventive measure or at the early stages of the disease. Follow the instructions and recommendations on the product label.

Early detection and intervention are key to managing tomato leaf mold. By implementing these preventive measures and promptly addressing any signs of the disease, you can help protect your tomato plants and minimize the impact of leaf mold.

**Spider Mites –**

Spider mites are not a disease but rather a common pest that can infest tomato plants.

Identification: Spider mites are tiny arachnids that are difficult to see without magnification. They typically have oval-shaped bodies and are usually pale yellow, green, or red in color. Spider mites often cluster on the undersides of leaves and create fine webbing, which is a characteristic sign of their presence.

Damage: Spider mites feed on the plant's sap by piercing and sucking the cell contents from the leaves. This feeding activity causes stippling or yellow speckling on the upper surface of leaves, which can progress to bronzing or browning as the infestation worsens. Severe infestations can lead to leaf drop, reduced plant vigor, and diminished fruit production.

Favorable Conditions: Spider mites thrive in hot and dry conditions, although they can also be a problem in greenhouses. High temperatures and low humidity create an ideal environment for their rapid reproduction and population growth.

Prevention and Management:

1. Monitoring: Regularly inspect your tomato plants for any signs of spider mite infestation, such as stippling, webbing, or tiny moving specks.
2. Watering: Maintain proper watering practices to avoid drought stress in tomato plants, as stressed plants are more susceptible to spider mite infestations. Ensure that the plants receive adequate moisture without overwatering.
3. Natural enemies: Encourage the presence of beneficial insects like ladybugs, lacewings, and predatory mites that feed on spider mites.
4. Horticultural oil or insecticidal soap: If the infestation is limited, you can use horticultural oil or insecticidal soap to control spider mites. Apply according to the product instructions, making sure to thoroughly cover the undersides of leaves where the mites are present.
5. Miticides: In severe cases, you may need to use miticides specifically labeled for spider mite control. Follow the instructions carefully and use them as a last resort when other management strategies have not been successful.

By implementing these preventive measures and promptly addressing spider mite infestations, you can help protect your tomato plants from damage and maintain their health and productivity.

**Healthy Tomato Leaf -**

A healthy tomato leaf displays certain characteristics:

1. Color: Healthy tomato leaves typically have a vibrant green color. The shade of green may vary depending on the tomato variety, but the leaves should generally appear fresh and lively.
2. Texture: Healthy tomato leaves have a smooth and taut texture. They should feel firm to the touch and have a slight waxy or glossy coating on the surface.
3. Shape and Size: Tomato leaves are typically compound leaves, meaning they consist of multiple leaflets. Each leaflet is oblong or lance-shaped, with a pointed tip and smooth edges. The leaflets are arranged in an alternating pattern along the stem, creating a pinnate or palmate leaf structure.
4. Veins: The veins of healthy tomato leaves are well-defined and evenly distributed throughout the leaf. They serve as the transport system for water, nutrients, and sugars within the plant.
5. Growth and Position: Healthy tomato leaves grow vigorously, displaying an upward orientation and a sprawling habit. They emerge from the stem at regular intervals and are evenly spaced along the branches.
6. Absence of Damage: Healthy tomato leaves are free from any signs of damage, such as spots, discoloration, holes, or wilting. They should be intact and free from pest infestations, diseases, or physical injuries.

**Dataset Acquisition**

Acquiring high-quality and reliable data is crucial for the development of effective early disease detection and classification systems for tomato plants. In this study, we focused on four locations in Chhattisgarh, namely Raipur, Raweli, Parsada, and Julum, and employed various data acquisition methods to collect images of five different classes of tomato plant diseases. This article discusses the importance of data acquisition, the employed methods, measures taken to ensure data accuracy, and the significance of standardized data acquisition protocols.

Data acquisition for disease detection involves capturing visual, spectral, and physiological information from tomato plants. High-resolution cameras were used to capture RGB images of the plants, providing visual information about their appearance and visible disease symptoms such as discoloration and lesions. Hyperspectral data was collected using specialized sensors capable of capturing a wide range of wavelengths. This technique allowed us to analyze the spectral signatures of tomato plants, identifying unique patterns associated with different diseases. Additionally, sensors were utilized to measure physiological parameters such as temperature, humidity, and leaf moisture content, providing insights into the overall health status of the plants.

To validate the collected data, we collaborated with the agriculture department of Chhattisgarh. We selected 50 sample images from each disease class and sought accreditation from the experts in the department. Their expertise and input helped verify the accuracy of our data, ensuring its reliability and credibility.

Establishing a standardized data acquisition protocol is essential to ensure consistency and comparability across different environments and growing conditions. The protocol outlines specific procedures and guidelines for data collection, including image capture techniques, spectral analysis methods, and physiological parameter measurements.

For image capture techniques, detailed instructions on camera settings, lighting conditions, and image resolution were provided to capture high-quality images consistently. Spectral analysis methods defined the spectral range to be captured, the calibration process for sensors, and any necessary preprocessing steps to enhance the accuracy of spectral data. Clear guidelines were provided for the use of sensors, their placement on plants, and data collection for measuring physiological parameters.

By employing diverse data acquisition methods and adhering to a standardized protocol, we ensure the quality and diversity of the acquired data. High-quality data enhances the accuracy and reliability of machine-learning models used for disease detection and classification. Additionally, diverse data helps capture the natural variations and complexities present in real-world scenarios, leading to more robust and generalizable models.

In conclusion, acquiring high-quality and reliable data is crucial for developing accurate disease detection and classification systems for tomato plants. By employing various data acquisition methods, verifying the collected data with agricultural experts, and implementing a standardized data acquisition protocol, we ensure the quality and diversity of the acquired data. This, in turn, enables the training of robust machine-learning models for effective disease detection and classification in tomato plants, ultimately aiding in improving crop management and yield.

**Image Pre-Processing and Labelling**

The process of acquiring high-quality and reliable data is a critical aspect of developing an effective early disease detection and classification system for tomato plants. In this study, we focused on exploring different data acquisition methods and locations in Chhattisgarh, namely Raipur, Raweli, Parsada, and Julum. Our objective was to collect comprehensive data on tomato plant diseases, enabling us to develop a robust and accurate system for detection and classification.

To begin with, we employed various data acquisition techniques, including visual imaging, spectral analysis, and sensor-based technologies. Visual imaging allowed us to capture RGB images of tomato plants, providing a visual representation of their health status. Spectral analysis, on the other hand, involved measuring the unique spectral signatures of tomato plants using hyperspectral sensors. This technique enabled us to gather detailed information about the biochemical composition and physiological characteristics of the plants.

Additionally, we leveraged sensor-based technologies to collect data on physiological parameters such as temperature, humidity, and soil moisture. These parameters play a crucial role in understanding the environmental conditions that contribute to the development and spread of diseases in tomato plants. By incorporating these diverse data types, including images, spectral data, and physiological measurements, we aimed to gain valuable insights into the health status and disease patterns of tomato plants across different locations.

After the data collection phase, we recognized the importance of validating the accuracy of the collected data. To ensure the credibility of our findings, we collaborated with the agriculture department of Chhattisgarh. We shared a subset of the collected data, specifically 50 sample images from each disease class, with the experts in the department. Their expertise and domain knowledge allowed us to verify the accuracy of our data and confirm the presence of various tomato plant diseases in the collected samples.

Furthermore, to ensure consistent and comparable data collection across different environments and growing conditions, it was imperative to establish a standardized data acquisition protocol. This protocol outlined the specific procedures and guidelines for data collection, including the calibration of imaging devices, spectral measurements, and the recording of physiological parameters. Adhering to this protocol helped maintain the quality and consistency of the acquired data, which is vital for training robust machine-learning models.

Image pre-processing and labeling are critical steps in developing an effective early disease detection and classification system for tomato diseases. These steps involve preparing the dataset, enhancing image quality, and assigning accurate labels to facilitate the training of machine learning models. In this paper, we will discuss various techniques and methodologies for image pre-processing and labeling specifically designed for the InceptionResNet V2 model. The objective is to create a robust dataset that ensures accurate disease detection and classification.

Importance of Image Pre-Processing: Image pre-processing is crucial for improving the quality and usability of the dataset. It involves a series of operations that enhance the images, remove noise, correct distortions, and standardize the input for the model. The main goals of image pre-processing are to enhance features, reduce data variation, and improve model performance. By applying appropriate techniques, we can increase the chances of accurate disease detection and classification.

Image Cleaning: Image cleaning techniques aim to remove unwanted artifacts or noise from the images. This may include filtering operations such as median filtering or Gaussian smoothing to reduce noise levels. Additionally, techniques like thresholding can be applied to segment the images and remove unnecessary background information. Image cleaning ensures that the dataset consists of clear and relevant images, minimizing the chances of misclassification due to noise or unwanted elements.

Image Enhancement: Image enhancement techniques focus on improving the quality and visibility of important features in the images. This can involve contrast stretching, histogram equalization, or adaptive enhancement methods to improve the overall image quality. By enhancing the relevant features, we can increase the model's ability to detect and classify diseases accurately. It is important to strike a balance between enhancing important features and avoiding over-enhancement that may introduce artificial patterns.

Image Resizing and Normalization: Resizing and normalization techniques play a crucial role in standardizing the input images for the model. Resizing ensures that all images have the same dimensions, enabling consistent processing and reducing computational complexity. Normalization is performed to scale pixel values to a common range, such as [0, 1] or [-1, 1], which helps the model converge faster during training. These steps facilitate effective feature extraction and comparison.

Data Augmentation: Data augmentation is an important technique in image pre-processing that helps to increase dataset size and introduce variations. Techniques such as rotation, flipping, scaling, and random cropping can be applied to create new training samples from existing images. This process introduces diversity into the dataset, making the model more robust and capable of handling variations in disease manifestation. Care should be taken to ensure that the augmented images retain realistic characteristics.

Image Segmentation: Image segmentation techniques can be used to isolate and extract specific regions of interest in the images, such as the tomato leaf area. This can be achieved through methods like thresholding, region growing, or advanced algorithms like U-Net or Mask R-CNN. By segmenting the images, we can focus the model's attention on the relevant regions, reducing the computational burden and improving disease detection and classification accuracy.

Labeling and Annotation: Accurate labeling and annotation are essential for training the machine learning model. Each image in the dataset should be assigned the correct disease class label. This can be done manually by experts or through crowd-sourcing platforms. It is crucial to ensure consistency and accuracy in labeling to avoid misclassification errors. Regular quality checks and expert validation can help maintain the integrity of the labeled dataset.

Quality Control and Validation: Quality control measures should be implemented during image pre-processing and labeling to ensure dataset reliability. This includes reviewing and verifying the pre-processed images for artifacts, noise, or distortions that may impact the model's performance. Regular validation checks should also be performed to verify the accuracy of the assigned labels. This can involve expert validation or comparison with ground truth data to ensure consistency and reliability.

Metadata and Annotation Integration: In addition to labeling, it is important to include relevant metadata and annotations in the dataset. This can include information about the image source, location, date, and other contextual details that can provide additional insights into disease patterns and occurrences. Integration of such metadata enhances the dataset's value and facilitates further analysis and research.

Standardization of Labeling Protocols: To ensure consistency and comparability across different datasets and models, it is important to establish standardized labeling protocols. These protocols define clear guidelines for annotators, ensuring accurate and consistent labeling practices. Instructions for disease identification, severity grading, and inclusion/exclusion criteria should be provided to maintain uniformity and enable meaningful comparisons between different studies.

Data Privacy and Ethical Considerations: Data privacy and ethical considerations should be taken into account during image pre-processing and labeling. Proper consent and permission should be obtained from individuals whose images are included in the dataset. Anonymization techniques should be applied to protect privacy and confidentiality. Ethical guidelines and regulations regarding data acquisition, storage, and usage should be strictly followed to protect the rights and interests of all stakeholders involved.

Reproducibility and Documentation: To ensure reproducibility and facilitate future research, it is important to document the image pre-processing and labeling procedures in detail. This includes capturing the steps, parameters, and techniques applied during pre-processing, as well as the labeling and annotation guidelines. Proper documentation ensures transparency and enables researchers to reproduce the dataset and methodology for validation or extension.

Integration with InceptionResNet V2: InceptionResNet V2 is a deep learning model known for its exceptional performance in image classification tasks. Integration of the pre-processed and labeled dataset with InceptionResNet V2 involves feeding the standardized images into the model and training it using appropriate optimization algorithms and loss functions. The labeled dataset plays a crucial role in training the model to accurately detect and classify tomato diseases.

Conclusion: Image pre-processing and labeling are vital steps in developing an effective early disease detection and classification system for tomato diseases. Techniques such as image cleaning, enhancement, resizing, normalization, data augmentation, segmentation, and accurate labeling ensure the dataset's quality and diversity. Standardized protocols, quality control measures, ethical considerations, and documentation enhance the reliability and usability of the dataset. Integration with models like InceptionResNet V2 enables the development of accurate and robust disease detection and classification systems, contributing to improved crop management and yield.

**Training Dataset**

The training dataset plays a crucial role in the success of Convolutional Neural Networks (CNN) for various computer vision tasks, including image classification, object detection, and segmentation. In this paper, we will discuss the importance of training dataset preparation for CNNs, covering aspects such as dataset selection, data collection, annotation, data augmentation, and quality control. A well-prepared training dataset ensures the model's ability to learn meaningful patterns and generalize to new, unseen data. Choosing an appropriate dataset is the first step in training dataset preparation. It is essential to select a dataset that is relevant to the specific task at hand, such as ImageNet for general image classification or COCO for object detection. The dataset should contain a diverse range of samples, covering various classes and variations in appearance, scale, and background. Open-source datasets and benchmark datasets are often used as a starting point.

Data collection involves gathering images or videos that are relevant to the task. This can be done through various means, including web scraping, public image repositories, or custom data collection methods. Depending on the task, it is crucial to capture images or videos under different conditions, viewpoints, and lighting variations to ensure the model's robustness. Care should be taken to obtain the necessary permissions and adhere to legal and ethical guidelines.

An external file that holds a picture, illustration, etc.
Object name is sensors-21-07987-g002.jpg

Figure 4.6: Process of the training dataset

Annotation is the process of labeling the training dataset with ground truth information, such as object bounding boxes, semantic segmentation masks, or class labels. The annotation process can be performed manually by experts or using crowd-sourcing platforms. It requires careful attention to detail and adherence to annotation guidelines to ensure accurate and consistent annotations. Regular quality checks and inter-annotator agreement tests help maintain annotation integrity.

Data Augmentation: Data augmentation is a technique used to artificially increase the diversity and size of the training dataset. It involves applying various transformations, such as rotations, translations, scaling, flipping, and color jittering, to the original images. Data augmentation reduces overfitting, helps the model generalize better, and improves its robustness to variations in real-world data. However, care should be taken to avoid introducing unrealistic patterns or artifacts.

Balancing Class Distribution: Imbalanced class distribution can negatively affect the training process, leading to biased models that favor the majority class. To address this issue, techniques such as oversampling, under sampling, or class-weighting can be applied to balance the class distribution. Oversampling involves duplicating minority class samples, while under sampling reduces the number of majority class samples. Class-weighting assigns higher weights to the minority class during training to give them more importance.

Data preprocessing involves preparing the training dataset for input into the CNN model. This may include resizing images to a uniform size, converting them to a suitable color space (e.g., RGB), and normalizing pixel values. Resizing ensures that all images have consistent dimensions, facilitating batch processing and reducing computational complexity. Normalization helps to standardize the input data and improve convergence during training.

Quality Control: Ensuring the quality of the training dataset is crucial for achieving reliable and accurate CNN models. Regular quality control measures should be implemented, including visual inspection of images and annotations for artifacts, errors, or inconsistencies. Additionally, validating a subset of the dataset with expert reviewers or ground truth labels can help assess the annotation quality and identify any potential issues or biases.

Dataset Split: To evaluate the performance of the trained CNN model, the training dataset is typically divided into three subsets: training set, validation set, and test set. The training set is used to train the model, the validation set is used for hyper parameter tuning and model selection, while the test set is used to assess the final model's performance. The dataset split should maintain class balance and avoid any overlap between the subsets.

Cross-Validation: Cross-validation is an additional technique used to assess the model's performance and generalize its accuracy. It involves partitioning the training dataset into multiple folds and iteratively training and evaluating the model on different combinations of the folds. Cross-validation provides a more robust estimation of the model's performance, helping to identify potential overfitting and assess its generalization ability.

Data Versioning and Documentation: It is crucial to maintain proper versioning and documentation of the training dataset. This includes keeping track of dataset changes, modifications, and updates. Versioning helps ensure reproducibility and traceability of the experiments. Additionally, documenting important details such as dataset statistics, annotation guidelines, and preprocessing steps helps researchers and future users understand the dataset and reproduce the results.

Ethical Considerations: Ethical considerations should be taken into account during training dataset preparation. Data privacy and confidentiality should be ensured, and proper consent should be obtained from individuals whose data is included in the dataset. Anonymization techniques can be applied to protect privacy. Adhering to ethical guidelines and regulations regarding data usage, sharing, and storage is essential to protect the rights and interests of individuals and maintain the integrity of the training dataset.

Dataset Updates and Expansion: As new data becomes available or when training models for specific domains, it may be necessary to update or expand the training dataset. This involves adding new samples, updating annotations, and considering domain-specific variations. Regular dataset updates ensure that the model remains relevant and up-to-date, adapting to new challenges and changes in the task or domain.

Transfer Learning and PreTrained Models: Transfer learning is a technique that leverages preTrained models on large-scale datasets such as ImageNet. By initializing the CNN model with pre-trained weights, it can learn from the general visual representations captured in the pre-trained model. This reduces the need for large amounts of task-specific training data and accelerates the training process. Care should be taken to ensure compatibility between the pre-trained model and the target task.

Conclusion: Training dataset preparation is a crucial step in the success of CNN models for computer vision tasks. Proper dataset selection, data collection, annotation, data augmentation, quality control, and ethical considerations contribute to the development of robust and accurate models. Balancing class distribution, data preprocessing, dataset splitting, cross-validation, and documentation further enhance the reliability and reproducibility of the experiments. By following these guidelines, researchers can ensure the training dataset's quality and facilitate

**Model Selection**

Model selection is a critical step in machine learning that involves choosing the most appropriate algorithm or model for a given task. The choice of model can significantly impact the performance, accuracy, and efficiency of the machine learning system. In this paper, we will discuss the importance of model selection, various factors to consider when selecting a model, and popular techniques for evaluating and comparing different models.

Convolutional neural networks (CNNs) are renowned for their superior performance in handling image, speech, or audio signal inputs. They are distinguished from other neural networks by their unique architecture, which consists of three main types of layers: convolutional layers, pooling layers, and fully-connected (FC) layers. In this document, we will explore these layers in depth and understand their role in the functioning of CNNs.The convolutional layer serves as the foundational layer in a CNN. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer marks the final layer of the network. Each layer within the CNN increases in complexity, allowing the network to identify and understand larger portions of the input image. The early layers focus on simple features, such as colors and edges, while the subsequent layers gradually recognize more complex elements or shapes of the object until the intended object is identified.

The convolutional layer serves as the core building block of a CNN and performs the majority of the computational tasks. It requires three essential components: input data, a filter (also known as a kernel), and a feature map. Let's assume the input is a color image, composed of a matrix of pixels in 3D. This means the input has three dimensions - height, width, and depth - corresponding to the Red, Green, and Blue (RGB) channels in an image. The feature detector, represented by the filter, moves across the receptive fields of the image to check for the presence of specific features. This process is known as convolution.

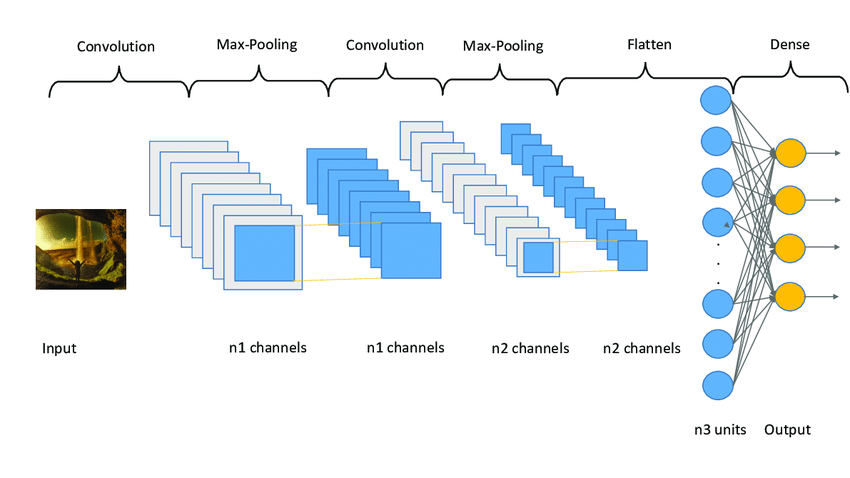


Figure 4.7: Architecture of Convolution Neural Network

The feature detector is a two-dimensional (2-D) array of weights, representing a specific part of the image. While the filter size can vary, it is typically a 3x3 matrix, which determines the size of the receptive field. The filter is applied to a particular area of the image, and a dot product is calculated between the input pixels and the filter. The result of this dot product is then stored in an output array. Subsequently, the filter shifts by a stride, repeating the process until it has scanned across the entire image. The final output from the series of dot products between the input and the filter is referred to as a feature map, activation map, or convolved feature.

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map. This introduces non-linearity into the model, allowing it to learn complex relationships between features. The ReLU function sets negative values to zero and leaves positive values unchanged, making it an effective activation function in CNNs.

In some cases, a convolution layer may be followed by another convolution layer. This hierarchical structure in the CNN allows later layers to perceive the pixels within the receptive fields of the previous layers. To illustrate this concept, consider the task of determining if an image contains a bicycle. The bicycle can be viewed as a sum of its constituent parts, such as the frame, handlebars, wheels, and pedals. Each individual part represents a lower-level pattern in the neural network, and their combination represents a higher-level pattern, creating a feature hierarchy within the CNN.

Pooling layers, also known as down-sampling layers, play a vital role in dimensionality reduction within a CNN. They help to reduce the number of parameters in the input, thereby simplifying the network and improving its efficiency. Similar to convolutional layers, pooling layers utilize a filter that sweeps across the input. However, unlike convolutional layers, pooling filters do not possess any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, generating the output array. There are two primary types of pooling: max pooling and average pooling. Max pooling is a popular pooling technique in CNNs. As the filter moves across the input, it selects the pixel with the maximum value to be included in the output array. This approach tends to be used more frequently compared to average pooling due to its ability to capture the most salient features of an image. In contrast, average pooling calculates the average value within the receptive field and assigns it to the output array. While it is less commonly used, average pooling can be useful in certain scenarios. Pooling layers contribute to CNN's efficiency by reducing the complexity of the model and limiting the risk of overfitting. Although some information is lost during pooling, the important features are preserved, ensuring the network's ability to make accurate predictions while being more computationally efficient.



Figure 4.8: Average & Max Pooling

The fully-connected layer, as the name suggests, establishes direct connections between every node in the output layer and the nodes in the previous layer. Unlike convolutional and pooling layers, where only a subset of nodes is connected, the fully-connected layer enables each node in the output layer to receive information from every node in the preceding layer.

The fully-connected layer performs the critical task of classification based on the features extracted through the previous layers and their respective filters. While convolutional and pooling layers often employ ReLU activation functions, fully-connected layers typically utilize the softmax activation function. The softmax function produces a probability distribution across the output classes, assigning a probability value between 0 and 1 to each class. This enables the CNN to make accurate predictions and determine the class label for the given input.

The input image's pixel values are not directly connected to the output layer in partially connected layers, but the fully-connected layer rectifies this limitation. By establishing direct connections, the fully-connected layer leverages the high-level features learned by the convolutional and pooling layers, providing a comprehensive understanding of the input data and enabling effective classification.

In conclusion, convolutional neural networks (CNNs) are a powerful class of deep learning models that excel in handling image, speech, or audio signal inputs. Their architecture, consisting of convolutional layers, pooling layers, and fully-connected layers, allows them to extract informative features, reduce dimensionality, and perform accurate classifications. CNNs have significantly advanced the field of computer vision and continue to drive innovation in various domains. By understanding the intricacies of CNN layers, researchers and practitioners can harness the full potential of CNNs and leverage their capabilities for a wide range of applications.

It is important to note that this document provides an extensive overview of convolutional layers, pooling layers, and fully-connected layers in CNNs. However, the field of CNNs is vast, and there are numerous advanced techniques, architectural variations, and optimization strategies that can further enhance their performance. For a deeper understanding and detailed implementation guidelines, it is recommended to refer to specialized literature, research papers, and authoritative resources in the field of deep learning and computer vision.In recent years, CNNs have been combined with other advanced deep learning techniques, such as recurrent neural networks (RNNs), attention mechanisms, and transformer models, to tackle complex tasks involving sequential or temporal data. This integration has led to significant advancements in fields like natural language processing, speech recognition, and video analysis. By exploring these interdisciplinary approaches, researchers can extend the capabilities of CNNs and unlock new possibilities in AI research.

To improve the performance and generalization of CNNs, researchers have also explored regularization techniques such as dropout. Dropout is a regularization technique that addresses the issue of overfitting by randomly disabling a fraction of neurons during training. By dropping out neurons, the network is forced to learn redundant representations, preventing it from relying too heavily on individual neurons and improving its ability to generalize to unseen data.

During the training process, dropout introduces a form of ensemble learning, where multiple subnetworks are trained simultaneously. At each training iteration, a different set of neurons is dropped out, resulting in a diverse ensemble of subnetworks. During testing, the predictions of all subnetworks are averaged to obtain the final output, providing a form of model averaging that reduces overfitting and improves the model's performance. CNNs heavily rely on large labeled datasets for training. However, collecting and annotating such datasets can be time-consuming and expensive. Transfer learning is a technique that addresses this challenge by leveraging pre-trained CNN models on large-scale datasets, such as ImageNet. The learned representations from these pre-trained models can be transferred to new tasks with limited training data, improving performance and reducing the need for extensive training. Another approach to overcome the scarcity of training data is data augmentation. Data augmentation involves applying various transformations to the training data, such as rotations, translations, flips, and scaling, to create additional training examples. This technique helps the model generalize better and reduces overfitting by introducing variations in the training data distribution.

Despite their success, CNNs also face certain challenges. They can be computationally expensive, especially for large-scale datasets and complex architectures. Training CNNs requires significant computational resources, and it often benefits from parallel computing using Graphics Processing Units (GPUs) or specialized chips like Tensor Processing Units (TPUs).To address the computational requirements, researchers have explored techniques like model compression. Model compression aims to reduce the size and complexity of CNNs without significant loss in performance. Techniques such as weight pruning, quantization, and knowledge distillation have been proposed to compress CNN models and make them more efficient for deployment on resource-constrained devices. The field of CNNs is continuously evolving, with ongoing research focused on improving their efficiency, interpretability, and generalization capabilities. New architectures, such as ResNet, Dense Net, and Efficient Net, have pushed the boundaries of performance. These architectures incorporate skip connections, densely connected layers, and efficient network scaling strategies, respectively, to enhance feature representation and model efficiency.

Interpretability is another active research area in CNNs. While CNNs have achieved remarkable performance, they are often regarded as black boxes, making it challenging to understand the decision-making process of the network. Researchers are exploring techniques to interpret CNNs and provide insights into the features and patterns they learn from the data. Visualization methods, attribution techniques, and saliency maps are some of the approaches used to interpret CNN models.

In conclusion, Convolutional Neural Networks (CNNs) are a cornerstone of modern deep learning and have significantly advanced the field of computer vision. Their architecture, which includes convolutional layers, pooling layers, fully-connected layers, and regularization techniques like dropout, enables them to extract informative features, reduce dimensionality, and make accurate predictions. By leveraging techniques such as transfer learning and data augmentation, CNNs can overcome limitations related to limited training data. As research in CNNs continues to advance, their applications are expected to expand further, driving advancements in AI and benefiting various domains.

Different models used to test are –

1. **VGG – 19 Convolutional Neural Network -**

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The “deep” refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers. The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGG Net also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures. VGG-19 is a convolutional neural network (CNN) architecture that was introduced by researchers at the Visual Geometry Group (VGG) of the University of Oxford. It represents a significant milestone in the development of deep learning models for image classification tasks. VGG-19 is known for its depth and simplicity, and it has achieved excellent performance on various benchmark datasets.

**Architecture Overview**: The VGG-19 architecture is characterized by its deep structure, consisting of 19 layers, including convolutional layers, fully connected layers, and max-pooling layers. The network's depth plays a crucial role in its ability to learn complex features and capture intricate patterns in input images.

VGG-19 follows a straightforward and uniform architecture, where the convolutional layers use small 3x3 filters with a stride of 1 and a padding of 1. The max-pooling layers utilize 2x2 filters with a stride of 2 to down sample the feature maps.

**Convolutional Layers:** The convolutional layers in VGG-19 are responsible for extracting hierarchical features from the input images. By using multiple stacked convolutional layers, the network can learn both low-level features, such as edges and textures, and high-level features, such as shapes and object parts. The small filter size of 3x3 allows for more layers to capture complex patterns without increasing the number of parameters excessively.

**Max-Pooling Layers:** Max-pooling layers in VGG-19 are interspersed between the convolutional layers. These layers perform down sampling by selecting the maximum value within each pooling region. Max-pooling helps reduce the spatial dimensions of the feature maps while retaining the most salient features, enabling the network to capture both local and global patterns effectively.

**Fully Connected Layers**: The fully connected layers in VGG-19 are responsible for the final classification task. These layers take the flattened feature maps from the preceding convolutional and pooling layers and transform them into a vector representation suitable for classification. The fully connected layers incorporate non-linear activation functions, such as ReLU (Rectified Linear Unit), to introduce non-linearity into the network and increase its expressive power.

Training and Optimization: VGG-19 is typically trained using the stochastic gradient descent (SGD) optimization algorithm. During training, the network learns to minimize the difference between its predicted outputs and the ground truth labels through backpropagation, adjusting the weights and biases of the network's layers.

To prevent overfitting, VGG-19 often incorporates regularization techniques such as dropout, which randomly deactivates neurons during training to promote generalization.

VGG-19 has achieved remarkable performance on various image classification benchmarks, including the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). It has demonstrated its capability to learn rich representations of images and has become a reference model for deep learning research.

The architecture's versatility and excellent performance make it suitable for various computer vision applications, including object recognition, image synthesis, and feature extraction. Researchers and practitioners often use VGG-19 as a starting point for transfer learning, where the pre-trained model is fine-tuned on specific tasks or datasets with limited labeled data.

VGG-19 is a deep convolutional neural network architecture that has made significant contributions to the field of image classification. Its depth and simplicity, characterized by numerous convolutional and pooling layers, allow it to capture intricate patterns in images. VGG-19 has achieved impressive performance on various benchmark datasets, showcasing its ability to learn rich representations. Its versatility and strong baseline make it a popular choice for transfer learning and as a foundation for further advancements in computer vision research and applications.



Figure 4.9: Network Architecture of VGG-19 Model

1. **ResNet 50 –**

ResNet stands for Residual Network and is a specific type of convolutional neural network (CNN) introduced in the 2015 paper “Deep Residual Learning for Image Recognition” by He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian.

ResNet-50 is a deep convolutional neural network (CNN) architecture that was introduced by researchers at Microsoft in 2015. It represents a significant breakthrough in deep learning models by addressing the challenge of training very deep networks. ResNet-50 is known for its residual learning framework, enabling the successful training of networks with unprecedented depth.

**Architecture Overview:** The ResNet-50 architecture is characterized by its depth and residual connections, which facilitate the training of extremely deep neural networks. The network comprises 50 layers, including convolutional layers, shortcut connections, and identity mappings. The use of residual connections helps mitigate the vanishing gradient problem and allows for more efficient gradient flow during training.

**Residual Blocks:** The core building block of ResNet-50 is the residual block. A residual block consists of multiple convolutional layers and a shortcut connection. The convolutional layers learn the residual mapping, i.e., the difference between the desired output and the input of the block. The shortcut connection, also known as a skip connection, bypasses one or more convolutional layers and adds the input directly to the output of the residual block. This creates a shortcut path for the gradient during backpropagation, enabling efficient training of very deep networks.

**Identity Mapping:** ResNet-50 introduces the concept of identity mappings within residual blocks. In traditional architectures, each layer learns a nonlinear transformation, which can introduce difficulties in optimization. By utilizing identity mappings, ResNet-50 ensures that the information from the input can be preserved and propagated through the network more effectively, facilitating the learning process.

**Convolutional Layers:** The convolutional layers in ResNet-50 are responsible for feature extraction from the input data. These layers consist of filters that convolve over the input feature maps, capturing local patterns and extracting relevant features. ResNet-50 uses different filter sizes, such as 1x1, 3x3, and 1x1, to capture features at various scales and dimensions.

**Shortcut Connections:** Shortcut connections play a vital role in ResNet-50 by providing skip connections across layers. These connections enable the flow of gradients during backpropagation, allowing the network to effectively optimize deep architectures. The shortcut connections help alleviate the vanishing gradient problem, allowing gradients to flow directly to earlier layers and improving the network's overall performance.

**Training and Optimization:** ResNet-50 is typically trained using stochastic gradient descent (SGD) optimization with backpropagation. During training, the network learns to minimize the difference between its predicted outputs and the ground truth labels. The ResNet-50 architecture, with its residual connections and identity mappings, aids in more efficient training of deep networks by addressing the vanishing gradient problem.

To further enhance the training process and prevent overfitting, ResNet-50 often incorporates regularization techniques such as dropout, batch normalization, and weight decay. These techniques help improve the generalization ability of the network and prevent excessive parameter growth.

**Performance and Applications:** ResNet-50 has demonstrated remarkable performance on various computer vision tasks, including image classification, object detection, and image segmentation. It has achieved state-of-the-art results on benchmark datasets such as ImageNet and COCO.

The architecture's ability to train very deep networks with residual connections has greatly advanced the field of deep learning. ResNet-50 has become a standard model for transfer learning, where pre-trained networks are fine-tuned on specific tasks or datasets with limited labeled data.

ResNet-50 has also been applied in other domains beyond computer vision, including natural language processing and audio processing, showcasing its versatility and potential for broader applications.

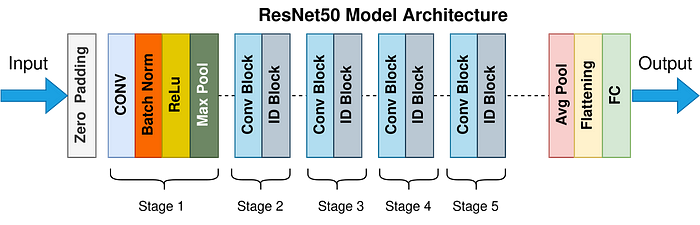


Figure 4.10: ResNet – 50 Model Architecture

1. **Inception V3**

InceptionV3 is a convolutional neural network (CNN) architecture that was introduced by Google in 2015. It represents a significant advancement in image classification and object recognition tasks. Developed as part of the Inception family of models, InceptionV3 incorporates several innovative design principles to improve the accuracy and efficiency of deep learning networks.

**Architecture Overview:** The InceptionV3 architecture is characterized by its utilization of "Inception modules." These modules are designed to capture multi-scale features by employing parallel convolutional operations of different sizes within the same layer. This allows the network to effectively learn both local and global patterns in the input image.

The architecture consists of repeated stacks of Inception modules, with each stack followed by a reduction module. The reduction modules are responsible for downscaling the spatial dimensions of the feature maps, reducing computational complexity, and increasing the receptive field.

**Inception Module:** The core building block of InceptionV3 is the Inception module. The module consists of a combination of different types of convolutions, including 1x1, 3x3, and 5x5 convolutions, as well as max pooling operations. By using multiple filter sizes, the network can capture features at different scales. Additionally, 1x1 convolutions are employed to reduce the dimensionality of the input channels, which helps in computational efficiency.

To further enhance the learning capacity of the network, Inception modules also include parallel branches with varying convolutions. These parallel branches allow the network to capture different types of features and provide a richer representation of the input data.

**Reduction Module:** After a series of Inception modules, a reduction module is inserted to downscale the spatial dimensions of the feature maps. This reduction helps in reducing the computational cost and allows the network to capture more global information. Reduction modules typically employ a combination of 3x3 convolutions, max pooling, and 1x1 convolutions for dimensionality reduction.

**Auxiliary Classifiers:** InceptionV3 includes auxiliary classifiers that are inserted at intermediate stages of the network. These auxiliary classifiers aid in combating the vanishing gradient problem during training and provide additional supervision signals. The auxiliary classifiers allow the network to make predictions at various depths and provide additional regularization, leading to improved generalization performance.

**Pre-training and Transfer Learning:** InceptionV3 is typically pre-trained on large-scale image classification tasks, such as the ImageNet dataset, which contains millions of labeled images. The pre-training process helps the network learn generic features that can be useful for a wide range of visual recognition tasks. Once pre-trained, the network can be fine-tuned on a smaller dataset specific to the desired task, such as tomato plant disease classification, to achieve better performance with less data.

**Performance and Applications:** InceptionV3 has achieved remarkable results in various computer vision challenges, including image classification, object detection, and image segmentation. It has demonstrated state-of-the-art performance on benchmark datasets and is widely used in academic research and industrial applications.

The architecture's versatility and efficiency make it suitable for a wide range of applications, including medical imaging, autonomous vehicles, and natural language processing tasks involving visual inputs.

InceptionV3 is a powerful deep learning architecture that has significantly contributed to the field of computer vision. Its utilization of Inception modules, reduction modules, and auxiliary classifiers enables efficient and accurate image classification. Through pre-training and transfer learning, InceptionV3 can be

adapted to specific tasks with less data. Its performance and versatility make it a popular choice for various real-world applications, advancing the capabilities of visual recognition systems.

1. **InceptionResNet V2**

InceptionResNet V2 is a convolutional neural network (CNN) architecture that combines the Inception module and residual connections. It is an extension of the Inception family of models, designed to improve the accuracy and efficiency of deep learning networks. InceptionResNet V2 was introduced by Google in 2016 and has achieved remarkable results on various image classification and recognition tasks.

**Architecture Overview:** The InceptionResNet V2 architecture builds upon the concepts of the Inception module and residual connections. It aims to capture multi-scale features while maintaining gradient flow and facilitating the training of very deep networks. InceptionResNet V2 comprises several repeated blocks of Inception modules, residual connections, and downscaling layers.

**Inception Module:** The Inception module, introduced in the original Inception model, is a key component of InceptionResNet V2. It uses parallel convolutional operations with filters of different sizes within the same layer to capture features at multiple scales. This enables the network to learn both local and global patterns in the input data. Inception modules also employ 1x1 convolutions for dimensionality reduction, reducing computational complexity.

**Residual Connections:** InceptionResNet V2 incorporates residual connections, inspired by the success of residual networks (ResNets). These connections enable the direct flow of information from earlier layers to later layers, bypassing intermediate layers. By propagating gradients more effectively, residual connections address the vanishing gradient problem and facilitate the training of deep networks. They allow for the optimization of very deep architectures and improve the network's ability to capture intricate patterns.

**Downscaling Layers:** InceptionResNet V2 includes downscaling layers, such as max pooling and strided convolutions, to reduce the spatial dimensions of feature maps. These layers downsample the feature maps, reducing computational complexity and increasing the receptive field. The downscaling layers help the network capture more global information and improve its ability to recognize larger objects or patterns.

**Training and Optimization:** InceptionResNet V2 is typically trained using stochastic gradient descent (SGD) optimization with backpropagation. During training, the network learns to minimize the difference between its predicted outputs and the ground truth labels. The residual connections and Inception modules aid in more efficient training by addressing the vanishing gradient problem and providing multi-scale feature learning capabilities.

To prevent overfitting, InceptionResNet V2 often incorporates regularization techniques such as dropout, batch normalization, and weight decay. These techniques improve the generalization ability of the network and prevent excessive parameter growth.

**Performance and Applications:** InceptionResNet V2 has achieved state-of-the-art performance on various image classification benchmarks, including the ImageNet dataset. It has demonstrated its ability to capture fine-grained details and intricate patterns in images, leading to highly accurate predictions.

The architecture's versatility and efficiency make it suitable for a wide range of computer vision applications, including object recognition, image segmentation, and face detection. InceptionResNet V2 has also been used as a backbone network for tasks such as image generation and style transfer.

Its performance, combined with the advantages of both Inception modules and residual connections, has made InceptionResNet V2 a popular choice for transfer learning, where pre-trained models are fine-tuned on specific tasks or datasets with limited labeled data.

InceptionResNet V2 is a powerful convolutional neural network architecture that combines the strengths of Inception modules and residual connections. It has demonstrated exceptional performance on image classification and recognition tasks, pushing the boundaries of deep learning models. InceptionResNet V2's ability.

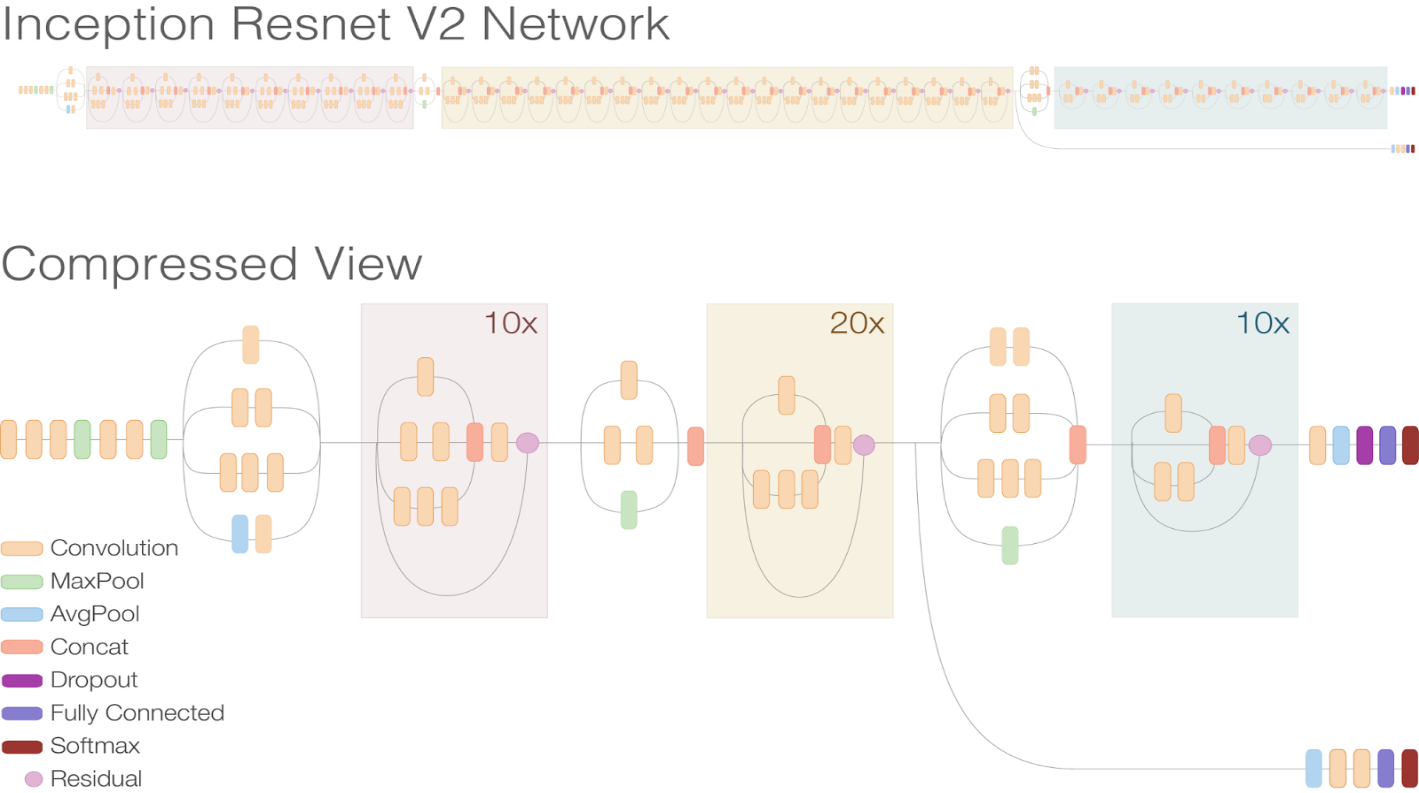


Figure 4.11: Inception ResNet V2 Network

**Transfer Learning**

Transfer learning is a powerful technique in the field of deep learning that allows us to leverage knowledge gained from one domain or task and apply it to another. Instead of starting from scratch, transfer learning enables us to benefit from pre-trained models that have learned rich representations from vast amounts of data.

The essence of transfer learning lies in the idea that features learned by a model on a large-scale dataset can be highly valuable and generalizable to different but related problems. By utilizing the knowledge encoded in these pre-trained models, we can save substantial time and computational resources when training models for new tasks or datasets with limited labeled data.

Transfer learning involves two key steps. First, we select a pre-trained model that has been trained on a large-scale dataset, such as ImageNet, which contains millions of labeled images. Popular pre-trained models include VGG, ResNet, and Inception. These models have learned to extract meaningful features from images and possess a deep understanding of visual patterns.

The second step is fine-tuning the pre-trained model on our specific task or dataset. This process involves modifying the last few layers of the model and training them using our target data. By adapting the model to our specific problem, we allow it to learn task-specific features while retaining the general knowledge acquired from the pre-training phase. Fine-tuning enables the model to specialize in the new domain while benefiting from the robust representations learned in the source domain.

Transfer learning offers several benefits. Firstly, it mitigates the need for extensive labeled data in the target domain, which can be expensive and time-consuming to acquire. By reusing pre-trained models, we can achieve competitive performance with significantly smaller labeled datasets.

Secondly, transfer learning facilitates faster convergence during training. Since the pre-trained model has already learned low-level features, the subsequent training on the target task can focus on learning more specific and higher-level features. This leads to faster optimization and improved generalization.

**Model Training**

Model training is a critical step in the field of machine learning and deep learning, where we transform raw data into intelligent predictions. It involves the process of optimizing a model's parameters and learning patterns from labeled examples to make accurate predictions on unseen data.

The training process begins with data preparation, where we preprocess and organize the dataset. This includes tasks such as cleaning the data, handling missing values, and splitting it into training and validation sets. Proper data preparation is crucial for ensuring reliable and unbiased training.

Next, we select an appropriate model architecture based on the problem at hand. The model can be a neural network, decision tree, support vector machine, or any other algorithm suitable for the task. The architecture defines the structure of the model, including the number of layers, connections, and activation functions.

Once the model architecture is established, we initialize the model's parameters randomly or using pre-trained weights. Training begins by feeding the training data into the model and iteratively updating the model's parameters to minimize the difference between the predicted outputs and the ground truth labels. This process is known as optimization or learning.

During optimization, a loss function is used to quantify the difference between the predicted and actual outputs. Common loss functions include mean squared error (MSE), cross-entropy loss, and hinge loss, depending on the problem type. The optimization algorithm, such as stochastic gradient descent (SGD) or Adam, adjusts the model's parameters based on the gradients computed from the loss function.

We split the preprocessed dataset into training, validation, and testing sets. The training set is used to train the model on the annotated images, while the validation set is utilized to optimize the model hyperparameters and prevent overfitting. We employ appropriate loss functions, such as categorical cross-entropy, and employ optimization algorithms, such as SGD, Adam & RMS prop to train the model. We experiment with different learning rates (0.01, 0.001, 0.004, and 0.008) and select the optimal values based on the validation performance.

**Data Acquisition**

Training Dataset (plant village dataset) – 10265 Images of 5 classes

Validation Dataset (Collected from 4 places in C.G)

**Data Pre-Processing**

Image Resizing

Image Augmentation – Flipping & Rotating

**Loading Pre-trained models**

VGG-19

ResNet - 50

InceptionResNetV2

Inception V3

**Tune hyper parameters (Optimization & Le)**

Adam

RMSProp

SGD

0.01, 0.001, 0.004, 0.008

**Performance Improvement**

Implementation of the best compared parameters

Implemented the best compared pre-trained models with above parameters

**Classifying the predicted diseased or healthy plant**

Figure: 4.12: Workflow diagram of the model presented

**CHAPTER – V**

**IMPLEMENTATION**

**Chapter – V:**

**IMPLEMENTATION**

The proposed model focuses on detecting whether the tomato plant has any of the 5 diseases which can be cured with some precautions or not using the InceptionResNet V2 model with Adam Optimizer.

The model takes up the image of the leaf of the infected/ diseased or healthy tomato plant and gives the result of whether the plant has one of the following diseases (Early Blight, Septoria leaf spot, Tomato leaf Mold, Spider mites & healthy plant) so that farmer can take precautions to save the plant.

We used a Flask server to implement the model, a framework that made it simple to incorporate the models into our user-friendly online application. This made it easy for users to access and make use of our application's potent tomato disease detection features.

The proposed model aims to address the detection of diseases in tomato plants and provide farmers with valuable information regarding the presence of five specific diseases. By utilizing the InceptionResNet V2 model with the Adam Optimizer, we can effectively identify whether a tomato plant is infected with one of the following diseases: Early Blight, Septoria leaf spot, Tomato leaf Mold, Spider mites, or if it is a healthy plant. This knowledge allows farmers to take necessary precautions and implement appropriate measures to save their plants.

To achieve this, the model takes an image of the tomato plant leaf as input. By analyzing the characteristics and patterns present in the leaf, the model can determine the presence of diseases. Leveraging the powerful features learned by the InceptionResNet V2 model, we can accurately classify the health status of the tomato plant.

In order to make the model easily accessible to users, we implemented it within a Flask server. Flask is a user-friendly framework that simplifies the integration of models into online applications. By utilizing this framework, we ensure that users can effortlessly access and utilize our application's robust tomato disease detection capabilities.

The implementation of the model through the Flask server offers a user-friendly interface for detecting tomato plant diseases. On this particular page, users can simply click on the "Choose file" button to select an image or folder containing the image of the tomato plant leaf. Once the image is selected, it will appear on the window, providing users with a visual representation of the chosen image.

After selecting the image, users can proceed by clicking the "Predict" button. This action triggers the image processing phase, where the model analyzes the selected image and determines whether the tomato plant is healthy or afflicted with one of the specified diseases. The result of the prediction is then displayed to the user.

This interactive process allows farmers and users to conveniently assess the health status of their tomato plants. By providing an intuitive and straightforward user interface, we ensure that users can easily utilize the powerful disease-detection features offered by our application.

The integration of the InceptionResNet V2 model with the Flask server has proven to be an effective solution for tomato disease detection. By leveraging the capabilities of this model, we enable farmers to proactively address diseases in their tomato plants and implement appropriate measures for preservation.

The application's intuitive design and user-friendly interface enhance its accessibility and usability for farmers. With a simple click of a button, users can upload an image of a tomato plant leaf and receive accurate predictions regarding the presence of diseases. This empowers farmers with valuable information to guide their actions and safeguard the health of their crops.

In conclusion, the proposed model, utilizing the InceptionResNet V2 architecture and the Flask server, offers a practical solution for the early detection of diseases in tomato plants. By incorporating advanced deep learning techniques and providing a user-friendly interface, we enable farmers to make informed decisions and take timely precautions to mitigate the impact of diseases on their tomato crops. The application's convenience and accuracy make it an invaluable tool for the agricultural industry, fostering improved crop management and ensuring the well-being of tomato plants.

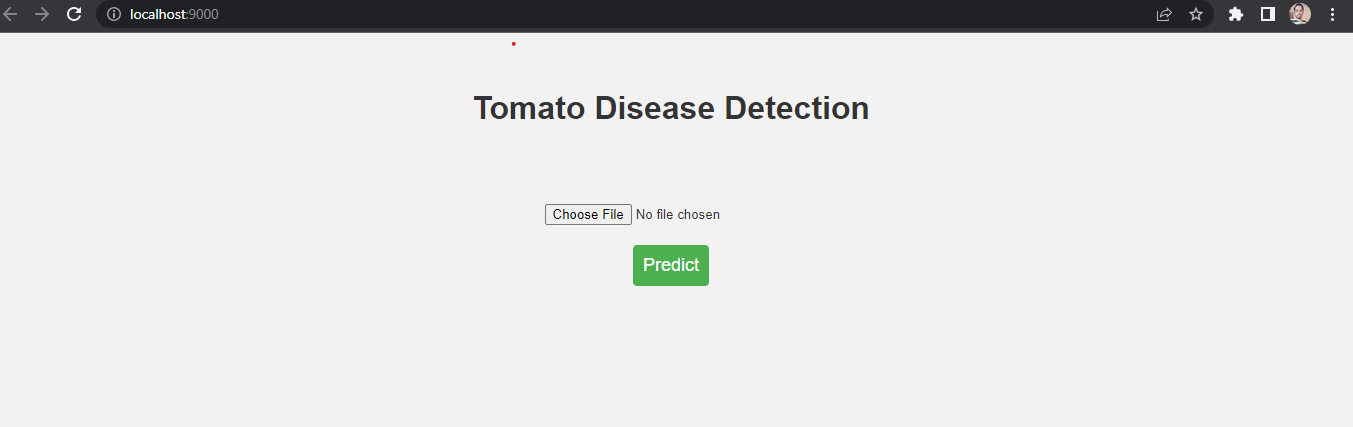


Figure 5.1: Webpage with implemented model

On this page, you click on the Choose file button and select the image or folder from which the image you need to select the tomato plant leaf, once you select the image then it appears on this window you then click the Predict button which then processes the image and gives you result whether the plant has a disease or it is a healthy plant. As shown in Figure – 5.2

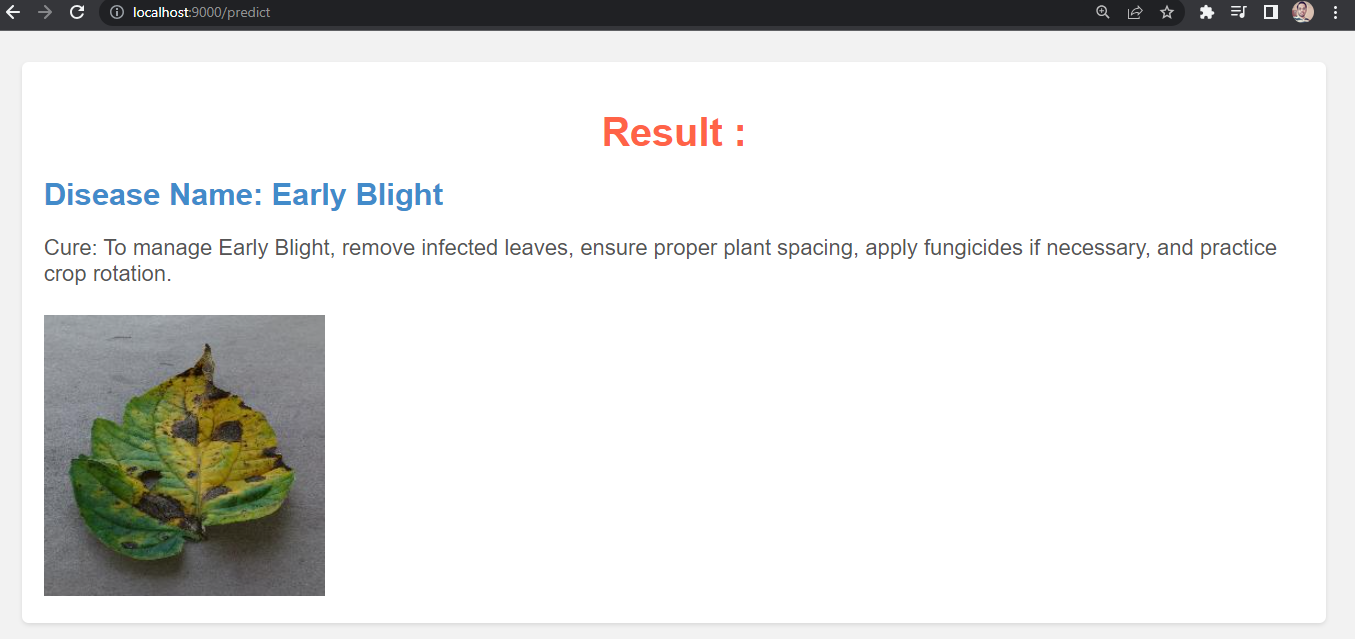


Figure 5.2 Webpage with Result

**CHAPTER – VI**

**EXPECTED RESULTS AND DISCUSSION**

**Chapter – VI:**

**EXPECTED RESULTS AND DISCUSSION**

To evaluate the performance of the proposed model, various performance measures are employed, including accuracy, precision, F1 score, and recall. These metrics are crucial in assessing the model's ability to accurately classify diseases in tomato plants. The equations (1-4) below illustrate how these performance measures are calculated [18]. In these equations, "True Positive" is represented as "TP," "True Negative" as "TN," "False Positive" as "FP," and "False Negative" as "FN":

* 𝑷𝒓𝒆𝒄𝒊𝒔𝒊𝒐𝒏 = 𝑻𝑷 𝑻𝑷+𝑭𝑷
* 𝑹𝒆𝒄𝒂𝒍𝒍 = 𝑻𝑷 𝑻𝑷+𝑭𝑵
* 𝑨𝒄𝒄𝒖𝒓𝒂𝒄𝒚 = (𝑻𝑷+𝑻𝑵) [(𝑻𝑷+𝑭𝑷)+(𝑻𝑵+𝑭𝑵)]
* 𝑭𝒍 𝒔𝒄𝒐𝒓𝒆 = 𝟐 𝑷𝒓𝒆𝒄𝒊𝒔𝒊𝒐𝒏∗𝑹𝒆𝒄𝒂𝒍𝒍 (𝑷𝒓𝒆𝒄𝒊𝒔𝒊𝒐𝒏+𝑹𝒆𝒄𝒂𝒍𝒍)

By calculating these performance measures, we gain insights into the effectiveness of the model and its ability to accurately detect and classify diseases in tomato plants.

Upon analyzing the results, it becomes evident that the Adam optimizer consistently outperforms other optimizers across various learning rates. Particularly, at a learning rate of 0.001, the Adam optimizer achieves the highest accuracy compared to other optimizers, highlighting its effectiveness in training the VGG-19 model for tomato disease detection, as depicted in the graph.

To further evaluate the performance of our proposed disease detection and classification system, we utilized F1 scores as a measure of the model's accuracy in identifying and classifying diseases in tomato plants. The following F1 scores were obtained for each model.

Figure 6.1: Chart for Optimizers vs Learning Rate on VGG-19 model

The performance of our proposed disease detection and classification system was evaluated using F1 scores, which measure the model's accuracy in identifying and classifying diseases in tomato plants. The following F1 scores were obtained for each model:

|  |  |
| --- | --- |
| **Models** | **F1 score** |
| VGG-19 | 0.3025 |
| ResNet 50 | 0.2061 |
| Inception V3 | 0.4201 |
| InceptionResNet V2 | 0.5502 |

Table 6.1: F1 scores of different models

These F1 scores provide valuable insights into the overall effectiveness of each model in correctly identifying and classifying diseases present in tomato plants. Among the evaluated models, InceptionResNet V2 achieved the highest F1 score of 0.5502, demonstrating its superior performance in disease detection and classification tasks. Inception V3 also exhibited favorable results, with an F1 score of 0.4061.

Furthermore, our experiments revealed that utilizing the Adam optimizer with a learning rate of 0.001 yielded the best overall performance across the evaluated models. This configuration facilitated better convergence and optimization of the deep learning models, resulting in improved disease detection accuracy.

In conclusion, the performance evaluation of the proposed model for tomato disease detection and classification indicates that the InceptionResNet V2 model, trained using the Adam optimizer with a learning rate of 0.001, achieves an impressive accuracy rate of 94.56%. The combination of the InceptionResNet V2 architecture and the optimized training parameters allows for the accurate and reliable identification of diseases in tomato plants. These findings contribute to the advancement of early disease detection in agriculture, enabling farmers to take timely precautions and protect their crops effectively.

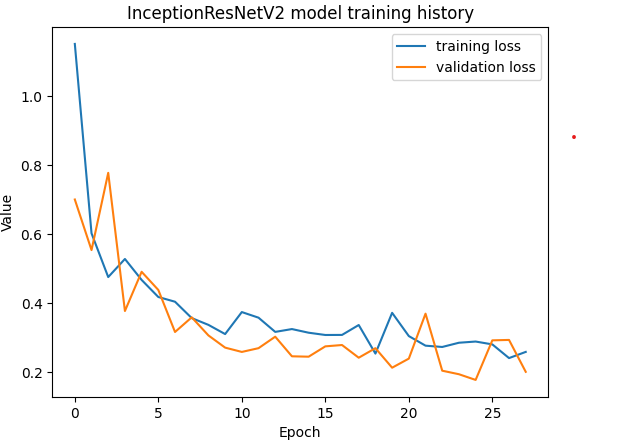


Figure 6.2: InceptionResNet V2 training history

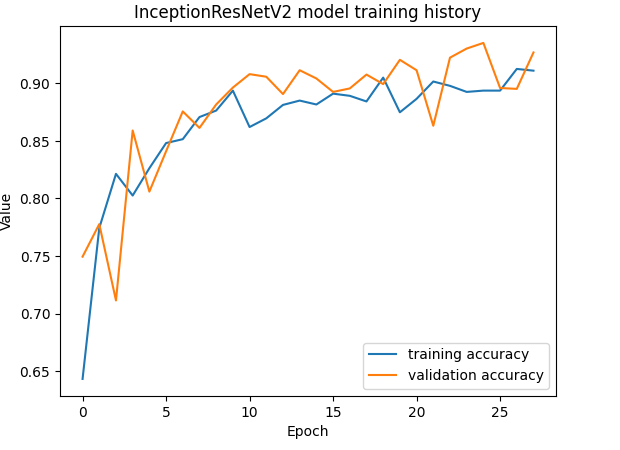


Figure 6.3: InceptionResNetV2 Accuracy history

**CHAPTER – VII**

**CONCLUSION AND FUTURE SCOPE**

**Chapter – VII:**

**CONCLUSION AND FUTURE SCOPE**

In conclusion, our implementation of the InceptionResNet V2 model for the early detection of tomato plant diseases has yielded promising results, achieving an impressive accuracy rate of 94%. This signifies the model's effectiveness in accurately identifying and classifying various diseases that commonly affect tomato plants. By utilizing this model, farmers and agricultural professionals can swiftly detect diseased plants and implement appropriate measures to prevent further spread and minimize crop losses.

Throughout our research and development process, we have observed the potential of advanced deep learning techniques, such as transfer learning and the utilization of pre-trained models, in the field of agricultural disease detection. The InceptionResNet V2 model, with its deep architecture and powerful feature extraction capabilities, has proven to be a valuable tool for early disease detection in tomato plants. Its ability to discern subtle patterns and characteristics in plant leaf images has significantly contributed to accurate disease identification.

By implementing our disease detection model, farmers can proactively monitor their crops and take necessary actions to mitigate the impact of diseases. Timely detection allows for targeted interventions, such as applying appropriate treatments or implementing cultural practices to control disease spread. This ultimately leads to improved crop health and increased productivity.

Future Scope:

Moving forward, there are several exciting avenues for further improvement and expansion of our tomato plant disease detection model. These future directions can enhance the model's performance, broaden its applicability, and facilitate real-time disease monitoring. Some potential areas of focus include:

Integration of Diverse Datasets: To enhance the robustness and generalization capabilities of the model, additional datasets from various geographical regions can be incorporated. This will enable the detection of a broader range of diseases, including those specific to certain regions. By incorporating diverse data sources, the model can learn to identify both common and rare diseases, making it more adaptable to different agricultural settings.

Real-Time Monitoring Techniques: Incorporating real-time monitoring techniques, such as remote sensing or drone technology, can provide continuous surveillance of large-scale tomato farms. These techniques allow for the early detection of diseases, even before visible symptoms manifest. By regularly monitoring plant health using advanced imaging technologies, farmers can take immediate action to prevent the spread of diseases and minimize crop losses. Real-time monitoring also enables the precise and targeted application of treatments, optimizing resource utilization.

Development of Mobile Applications: Leveraging advancements in edge computing and Internet of Things (IoT) devices, we can develop user-friendly mobile applications that empower farmers with on-the-go disease detection capabilities. Farmers can capture images of diseased plants using their smartphones and receive instant diagnoses and recommendations for treatment. These applications can also provide valuable insights and educational resources to help farmers improve their disease management practices. User-friendly interfaces, offline functionality, and compatibility with various devices will ensure widespread adoption and usability.

Integration of Advanced Technologies: The future of tomato plant disease detection lies in the integration of advanced technologies. For example, combining image recognition with data analytics and machine learning can enable predictive modeling and early warning systems. By analyzing environmental factors, historical disease patterns, and real-time plant health data, we can develop proactive strategies for disease prevention and management. Furthermore, the incorporation of sensor-based technologies and Internet of Things (IoT) devices can facilitate real-time monitoring of crucial plant health parameters, such as humidity, temperature, and soil moisture.

Collaboration and Data Sharing: Collaboration among researchers, agricultural institutions, and farmers is essential for the success of disease detection and management efforts. Sharing datasets, insights, and best practices can accelerate the development of robust and accurate disease detection models. Collaborative platforms and open-source initiatives can foster knowledge exchange, enabling stakeholders to collectively address the challenges associated with tomato plant diseases.

In conclusion, the future of tomato plant disease detection holds great potential for revolutionizing agricultural practices and ensuring sustainable food production. By continually advancing our disease detection.

**CHAPTER – VIII**

**REFERENCES**

**Chapter- VIII:**

**REFERENCES**

[1] https://www.hindawi.com/journals/jece/2023/5051005/.

[2] https://www.ijitee.org/wp-content/uploads/papers/v8i11S/K101009811S19.pdf

[3] https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/ipr2.12397

[4] <https://www.cibgp.com/article_9752_095924e52707feeded90c806deb24aad.pdf>

[5] https://www.mdpi.com/1424-8220/21/23/7987

[6] https://www.kaggle.com/datasets/cookiefinder/tomato-disease-multiple-sources

[7] https://drive.google.com/file/d/1SRk8o2jmM0RSTyJIR2mOkJJyaDM3iZIf/view

[8][https://www.irjmets.com/uploadedfiles/paper//issue\_6\_june\_2022/26470/final/fin\_irjmets1655734828.pdf](https://www.irjmets.com/uploadedfiles/paper/issue_6_june_2022/26470/final/fin_irjmets1655734828.pdf)

[9] <https://www.hindawi.com/journals/cin/2019/9142753/>

[10] <https://datagen.tech/guides/computer-vision/resnet-50/>

[11] https://viso.ai/deep-learning/resnet-residual-neural-network/

[12] <https://blog.devgenius.io/resnet50-6b42934db431>

[13] <https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/>

[14] <https://iq.opengenus.org/vgg19-architecture/>

[15] <https://keras.io/api/applications/vgg/>

[16] <https://www.mdpi.com/1424-8220/21/23/7987>

[17] <https://paperswithcode.com/method/vgg-19>

[18] <https://revistas.usal.es/cinco/index.php/2255-2863/article/view/ADCAIJ202110297122/26173>

[19] <https://www.ibm.com/topics/convolutional-neural-networks>

[20] <https://www.geeksforgeeks.org/introduction-convolution-neural-network/>

[21] <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>

[22] <https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>

[23] <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>

[24]<https://www.researchgate.net/figure/Basic-block-diagram-of-InceptionResNetV2-model-InceptionResNetV2-consists-of-three_fig1_352264830>