PRECISION AGRICULTURAL DISEASE MANAGEMENT USING DRONE

A PROJECT REPORT

submitted by

CB.EN.U4ELC20025	**KALPESH P**
CB.EN.U4ELC20031	**SUJAN SURYA K**
CB.EN.U4ELC20055	**RAM NARAYAN B**
CB.EN.U4ELC20080	**VASANT RAJ P A**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

MAY 2024

AMRITA VISHWA VIDYAPEETHAM

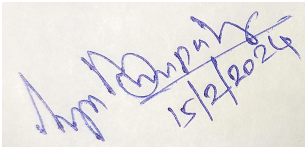
AMRITA SCHOOL OF ENGINEERING,COIMBATORE - 641 112

BONAFIDE CERTIFICATE

This is to certify that this project entitled **"PRECISION AGRICULTURAL DISEASE MANAGEMENT USING DRONE"** submitted by

CB.EN.U4ELC20025	**KALPESH P**
CB.EN.U4ELC20031	**SUJAN SURYA K**
CB.EN.U4ELC20055	**RAM NARAYAN B**
CB.EN.U4ELC20080	**VASANT RAJ P A**

in partial fulfillment of the requirements for the award of the **Degree of Bachelor of Technology** in **ELECTRICAL & ELECTRONICS ENGINEERING** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering.



Digital Signature of the Guide

Boopathy C P
Supervisor
Assistant Professor (Sl. Gr.)
Department of Electrical and Electronics Engineering
Amrita School of Engineering
Coimbatore- 641112

Balamurugan S
Chairperson
Professor
Department of Electrical and Electronics Engineering
Amrita School of Engineering
Coimbatore- 641112

This project report was evaluated by us on……

INTERNAL EXAMINER

EXTERNAL EXAMINER

Abstract

Precision agriculture is a contemporary approach that leverages technology to optimize crop yields while minimizing environmental impact. Our research project centers on developing a precision agriculture system utilizing drones equipped with advanced sensors and machine learning to efficiently identify crop diseases and precisely apply pesticides.

The system comprises three core elements: drones featuring camera and sensors, a machine learning model for disease detection, and a pesticide spraying mechanism. These components synergize to revolutionize disease management in agriculture.

By employing drones, we capture detailed aerial crop imagery, which the machine learning model processes for disease identification. Once identified, our system automatically initiates precise pesticide application, reducing resource waste and environmental harm.

This system's potential impact is considerable. It promises to curtail crop losses through early disease detection and targeted treatment. Furthermore, it enhances resource efficiency by minimizing pesticide use and fosters environmental conservation by reducing the dispersion of harmful chemicals.

This project envisions a future where agriculture is both more productive and environmentally responsible. By integrating drone technology, machine learning, and automated pesticide application, our precision agriculture system offers a promising and sustainable solution for managing crop diseases while promoting efficient and eco-friendly farming practices.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background of Precision Agriculture

The evolution of agriculture has undergone transformative phases since the Industrial Revolution. Initially characterized by the introduction of machinery such as tractors and implements, the agricultural landscape continued to evolve until the 1940s with the advent of synthetic chemicals. This marked a crucial juncture, setting the stage for increased global production but also raising concerns about the reliance on these chemicals.

Amidst these changes, *Precision Agriculture* emerged as a revolutionary management strategy. The overarching goal is to enhance resource use efficiency, increase productivity, improve crop quality, heighten profitability, and ensure the sustainability of agricultural production.

Technological advancements have played a pivotal role in propelling Precision Agriculture forward. The integration of sensors and actuators into agricultural machinery has not only increased crop yields but has also ushered in a reduction in resource inputs. This technological empowerment has provided crucial support to farmers, enabling them to supervise and implement agricultural practices with a heightened level of precision. It serves as a response to the challenges posed by the trajectory of agricultural development.

The contemporary agricultural landscape is experiencing a new *"Industrial revolution"* marked by the convergence of diverse technological components. This includes data management systems, variable application tools, satellite and Unmanned Aerial Vehicle (UAV) systems, Farm Management Information Systems (FMIS), localized irrigation, robotic implements, artificial intelligence, and autonomous mobile robotics. This amalgamation of technologies signifies a disruptive change, evident in the replacement of traditional machinery with advanced robotic systems, thereby heralding a new era in productivity and efficiency.

Precision Agriculture stands as a response to the evolving needs of modern agriculture. Its roots in understanding and harnessing the variability of crops, coupled with technological advancements, have positioned it as a key player in the current agricultural revolution. [1]

1.2 Significance of Disease Management in Agriculture

Diseases in plants, stemming from both biotic and abiotic factors, present a significant threat to agricultural systems, causing substantial losses. The importance of Integrated Disease Management (IDM) strategies, particularly for resource-poor farmers, involves active participation in developing specific techniques and solutions tailored to their farming systems. The success and sustainability of IDM rely on incorporating environmentally friendly control components accessible to farmers.

Comprehensive training and awareness initiatives, targeting farmers, are crucial for IDM's effectiveness. Farmers must be empowered with a practical understanding of the ecology, aetiology, and epidemiology of major crop diseases. This understanding is attained through extensive training using participatory approaches, ensuring farmers possess the necessary knowledge to effectively manage their fields. This knowledge is then translated into practical decision-making tools and control strategies, enabling farmers to actively engage in disease management within their agricultural systems. [2]

1.3 Challenges in Traditional Disease Forecasting Methods

Traditional disease forecasting in agriculture faces significant challenges, owing to the limitations of manual inspection and sampling methods. Farmers who rely on visual inspections may find it time-consuming, labor-intensive, and prone to human error, resulting in disease detection delays and the overlooking of latent infections. Furthermore, traditional prediction models based on historical data, weather patterns, and crop phenology may lack adaptability to changing conditions, resulting in inaccuracies, particularly when dealing with emerging or re-emerging diseases.

Traditional models face challenges due to the dynamic nature of pathogens, hosts, and the environment, and their reliance on historical patterns may leave them unprepared when confronted with novel threats. Integrating advanced technologies such as remote sensing, satellite imagery, sensor networks, and machine learning algorithms is critical for overcoming these constraints. These technologies offer real-time data, improved accuracy, and a more sophisticated approach to disease forecasting. Embracing innovation in agriculture is critical for proactive and resilient disease management in the face of changing challenges.

1.4 Evolution of Drone Technology in Agriculture

Amidst the ongoing transformative period in agriculture known as Farming 4.0, the integration of Information and Communication Technologies (ICT) has ushered in the fourth revolution. This evolution encompasses cutting-edge technologies such as Remote Sensing, the Internet of Things (IoT), Unmanned Aerial Vehicles (UAVs), Big Data Analytics (BDA), and Machine Learning (ML), all holding the promise to revolutionize age-old farming practices.

Within the realm of smart farming, the advent of IoT enables the meticulous monitoring of various agricultural parameters, thereby fostering enhanced crop yields and optimized resource inputs. Precision agriculture, made possible through the synergies of IoT and UAV technologies, introduces intelligent practices like precision irrigation and judicious use of fertilizers and pesticides. This not only minimizes the environmental footprint but also contributes to mitigating the adverse effects of climate change. The utilization of UAVs as sophisticated sensing and communication platforms stands out as a pioneering breakthrough in precision agriculture, offering invaluable support for monitoring and decision-making across diverse agricultural practices. [3].

As we delve into the realm of technological advancements in agriculture, the focus shifts towards the indispensable role that Drone technology plays in revolutionizing traditional farming practices.

1.5 Literature Review

Effective crop monitoring and pesticide spraying have witnessed transformative advancements with the integration of drone technology. Researchers have continually enhanced drone capabilities, focusing on sensor selection, smart farming applications, and innovative pest control methodologies.

Crop monitoring's evolution is marked by the integration of unmanned aerial vehicles (UAVs) equipped with diverse sensors [4]. Noteworthy studies by Hunt Jr. et al., Primicerio et al., and Hassan-Esfahani et al. contributed to this field, utilizing sensors like digital color-infrared and multispectral cameras [4]. Recent breakthroughs, exemplified by Su et al.'s automatic yellow rust disease monitoring system, leverage deep learning for improved segmentation and disease detection [4].

Traditional pesticide application methods face challenges in uniformity and coverage. Drone-mounted sprayers address these issues, offering enhanced coverage and chemical effectiveness [4].

Yamaha Motor Co. Ltd.'s early work laid the foundation for UAV-based spraying, leading to recent advancements by Martinez-Guanter et al. and Karan Kumar Shaw et al., incorporating adaptive control, genetic algorithms, and AI for variable spray systems [4].

Chen et al. harnessed edge intelligence and the Tiny-YOLOv3 algorithm for real-time pest detection, showcasing the successful integration of edge computing into smart farming practices [5]. The study, utilizing the APD-616X agricultural spray drone, demonstrated significant reductions in pesticide consumption, operating time, and workforce needs compared to traditional methods [5].

General trends in drone applications for agriculture are outlined by Hafeez et al., providing an encompassing review [4]. The study emphasizes the transition from semi-controlled to fully automated pesticide spraying systems, underscoring drones' potential in creating safer and more economical agricultural solutions [4].

Abouelmagd et al. contribute to precision agriculture with an optimized Capsule Neural Network (CapsNet) for tomato leaf disease classification [7]. The proposed CapsNet achieves remarkable accuracy, outperforming traditional Convolutional Neural Networks (CNNs) and laying the groundwork for employing unmanned aerial vehicles for efficient disease monitoring [7].

Stamford et al. introduced NDVIpi, a cost-effective dual camera system connected to a Raspberry Pi for NDVI imagery, demonstrating robust performance compared to commercial alternatives [8]. The study highlights the significance of red wavelengths in NDVI calculation and NDVIpi's sensitivity to chlorophyll content [8].

Soetedjo and Hendriarianti developed a low-cost infrared camera system for leaf detection and counting, demonstrating effectiveness in handling illumination changes and shadows [9]. The proposed method, evaluated against benchmark datasets, exhibits comparable performance and real-time execution feasibility for future applications in diverse natural environments [9].

Genze et al. propose DeBlurWeedSeg, a combined deblurring and segmentation model for weed and crop segmentation in motion-blurred images [10]. Trained on a dataset of blurred-sharp image pairs, DeBlurWeedSeg outperforms standard segmentation models, showcasing significant improvement in the Sørensen-Dice coefficient [10].

1.6 Objectives

For this research project on precision agriculture, the following objectives are defined:

1. To design and implement a precision agriculture system utilizing drones equipped with camera and machine learning techniques for the accurate identification of diseases in crops.
2. To develop a targeted pesticide application mechanism integrated with the drones for the precise treatment of identified crop diseases.

These objectives aim to create a comprehensive system that enhances disease management in agriculture by harnessing drone technology and cutting-edge machine learning techniques.

1.7 Block Diagram

In the realm of precision agriculture, a meticulously orchestrated workflow involving drone control and machine learning models is employed to bolster crop yields while minimizing input costs. The following steps outline our project's intricate process:

1. **Drone Control**: The drone is controlled to hover above the farm, capturing images of each plant through an onboard camera.
2. **Data Transmission to Base Station:** The captured images are then transmitted to a base station where a machine learning model, specialized in the identification of unhealthy plants, is located.
3. **Model Prediction**: The machine learning model predicts whether each plant is healthy or diseased, sending a response accordingly.
4. **Pesticide Application (If Required):** If a plant is identified as diseased, the motor for pesticide spraying is activated. Once the spraying is complete, the drone is moved to the next plant.
5. **Sequential Plant Monitoring**: The process repeats sequentially for each plant in the farm until all designated plants are covered.

6. **Return to Base Station**: Following the comprehensive monitoring process, the drone is manually brought back to the base station, serving as both a storage and charging facility.

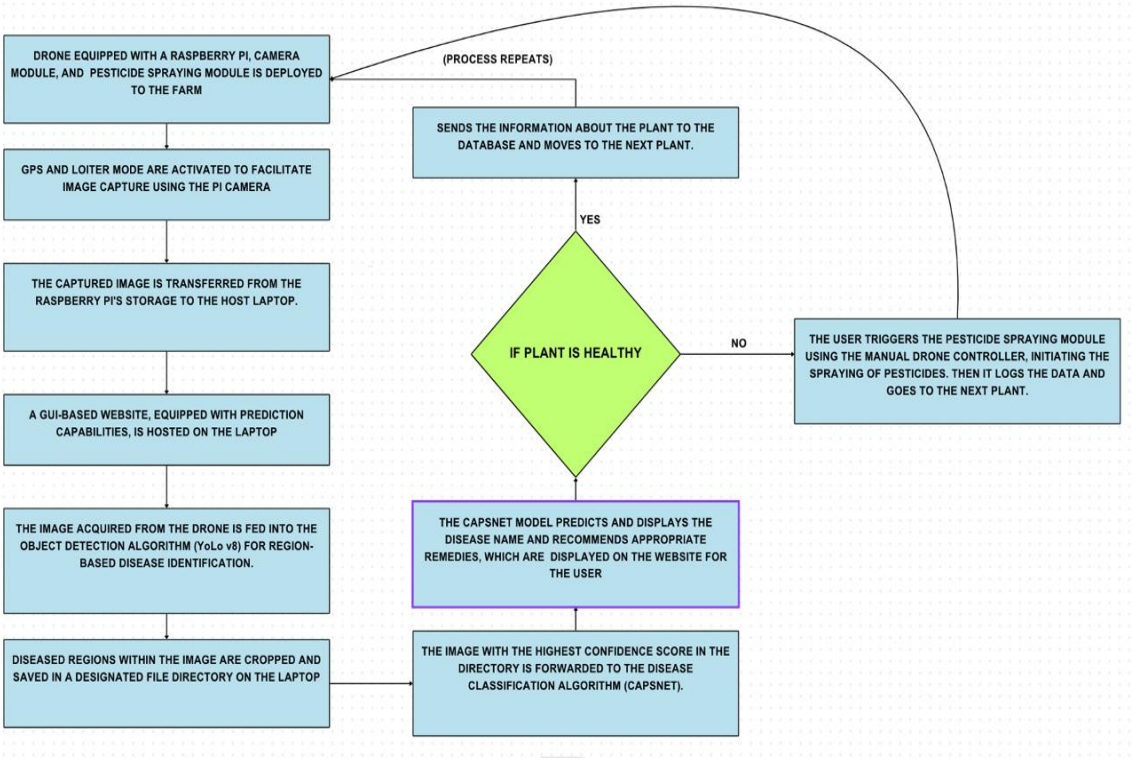


Figure 1.1: Block Diagram Representation

1.8 Report Outline

This report consists of this introductory chapter and other chapters arranged as below:

Chapter 2 presents about the Disease Prediction Architecture.

Chapter 3 describes the Dataset analysis used for Deep Learning.

Chapter 4 presents the Plant Disease Diagnosis Modelling.

Chapter 5 presents the Object Detection Methodology used.

Chapter 6 describes the various Image processing techniques used for addressing image quality.

Chapter 7 presents the Results obtained after training the CapsNet and YoLo v8 models.

Chapter 8 presents the details of Drone implementation along with the communication setup to facilitate disease diagnosis and treatment.

Chapter 9 presents about hosting the predictive model using a GUI based Website.

Chapter 10 presents the Discussion and Conclusion of this project

Chapter 2

DISEASE PREDICTION ARCHITECTURE

2.1 Environment and Deep Learning Implementation Framework

2.1.1 Google Colab

The research leverages Google Colab, a cloud-based Jupyter notebook environment, for seamless execution and collaboration.

2.1.2 Deep Learning Frameworks

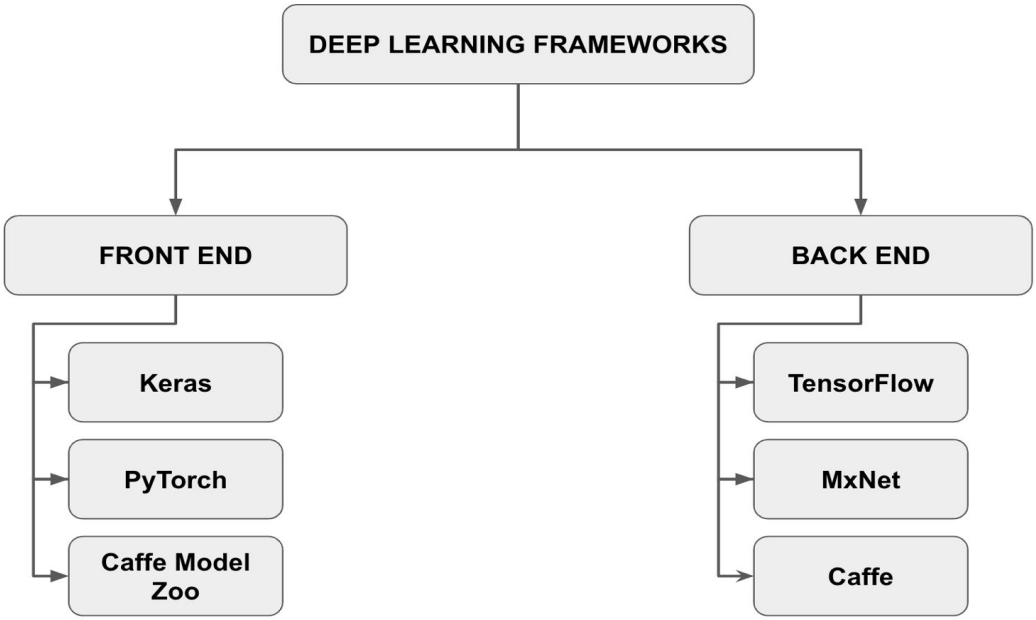


Figure 2.1: Deep Learning Frameworks

1. **Front End Framework - Keras**: Keras acts as a high-level frontend API that runs on top of TensorFlow (or other backend engines like Theano or Microsoft Cognitive Toolkit). It provides a user-friendly interface for designing and defining neural network architectures. Keras simplifies the process of building, training, and evaluating models, offering a more intuitive and user-friendly experience.

2. **Back End Framework - TensorFlow**: TensorFlow serves as the backend deep learning framework. It provides the computation and optimization capabilities needed for training and executing neural network models. TensorFlow handles low-level operations, computation graph construction, and optimization.

2.1.3 Libraries

1. **Numpy:**

   - **Role**: Numerical Operations and Array Manipulations
   - **Functionalities**: Numpy is employed for efficient numerical operations and array manipulations. It provides a foundation for handling numerical data in the form of arrays, enabling streamlined mathematical operations.

2. **Matplotlib and Seaborn:**

   - **Role**: Data Visualization
   - **Functionalities**: Matplotlib and Seaborn are utilized for creating visualizations, including training history plots and confusion matrices. These libraries offer versatile tools for generating insightful visual representations of data.

3. **Scikit-Learn:**

   - **Role**: Machine Learning Metrics
   - **Functionalities**: Scikit-Learn is integrated for metrics computation, including the confusion matrix and classification report. It provides a comprehensive set of tools for machine learning tasks, facilitating model evaluation and metrics computation.

4. **Keras ImageDataGenerator:**

   - **Role**: Real-time Data Augmentation
   - **Functionalities** Keras ImageDataGenerator is utilized for real-time data augmentation during the training of neural network models. It enables the generation of augmented training samples, contributing to improved model generalization and performance.

2.1.4 Callbacks for Training

1. **EarlyStopping:**

   - **Role:** Halt Training on Plateau
   - **Functionalities:** EarlyStopping is a crucial callback that halts training once a monitored metric ceases to improve. It prevents overfitting by terminating training when there is no further improvement in the specified metric.

2. **Reduce Learning Rate On Plateau:**

   - **Role:** Adaptive Learning Rate Adjustment
   - **Functionalities:** 'ReduceLROnPlateau' is an adaptive callback that adjusts the learning rate during training plateaus. By reducing the learning rate, it aims to fine-tune the model for improved convergence and performance.

2.1.5 Other Utilities

1. **PIL (Python Imaging Library):**

   - **Role:** Image Loading and Preprocessing
   - **Functionalities:** PIL is integral for image loading and preprocessing steps in the workflow. It facilitates the handling of image data, including loading images and performing preprocessing operations.

2. **I/O Module:**

   - **Role:** Input and Output Stream Handling
   - **Functionalities:** The io module facilitates the handling of input and output streams within the code. It provides essential utilities for managing data streams and file operations.

2.2 Capsule Network (CapsNet)

2.2.1 Introduction:

A Capsule Neural Network is intended to mimic the operation of biological neural networks, with a particular emphasis on improving recognition and segmentation skills. These networks, classified as a subtype of Artificial Neural Networks, introduce a novel feature known as capsules, which are effectively nested layers beneath the central capsule layer.

The term 'capsule' refers to the existence of these nested structures, which play a crucial part in setting the parameters of object properties. Using face recognition as an example, **network capsules are responsible not only for detecting the existence of certain facial traits, but also for understanding how these aspects are organised in relation to one another.** This goes beyond simple binary detection to a more subtle study of *feature spatial arrangement*, guaranteeing that the algorithm successfully detects a face by taking into account the precise order of its constituent elements.

A significant question that may emerge is how these capsule networks determine the right feature order. The answer lies in the network's exposure to a large number of different images during the training phase. The capsule network becomes competent in effectively organising and recognising features through the analysis of hundreds or even thousands of photos, allowing it to generalise well and produce correct identifications, like in the case of facial recognition. This feature emphasises the importance of the learning process in capsule networks and their adaptability to a variety of input conditions

2.2.2 Understanding Capsule Network Operation:

Let's delve into the mechanics of Capsule Networks. **Capsules initiate by conducting matrix multiplications to scrutinize spatial correlations between low-level and high-level features**. Subsequently, capsules employ **dynamic routing** to select a parent, a technique elaborated upon later in this report. Once a parent is determined, *the vectors' sum, constrained between 0 and 1, is computed while preserving their orientation*.

This methodology diverges significantly from conventional neural networks. *While tradi-*

tional networks rely on neurons, capsule networks utilize capsules to encapsulate crucial visual details. Capsules generate vectors, enabling them to discern the orientation of a face or specific feature, a capability neurons lack. Even as the feature's orientation changes, the vector's magnitude remains constant, adjusting only its direction accordingly.

Capsule networks exhibit proficiency on smaller datasets and offer more interpretable representations. They *meticulously encode image particulars like texture, position, and pose*. Nonetheless, their efficacy on larger datasets remains a limitation. Figure 2.2 illustrates the architectural layout of a Capsule Network [16], showcasing its intricate design and functionality.

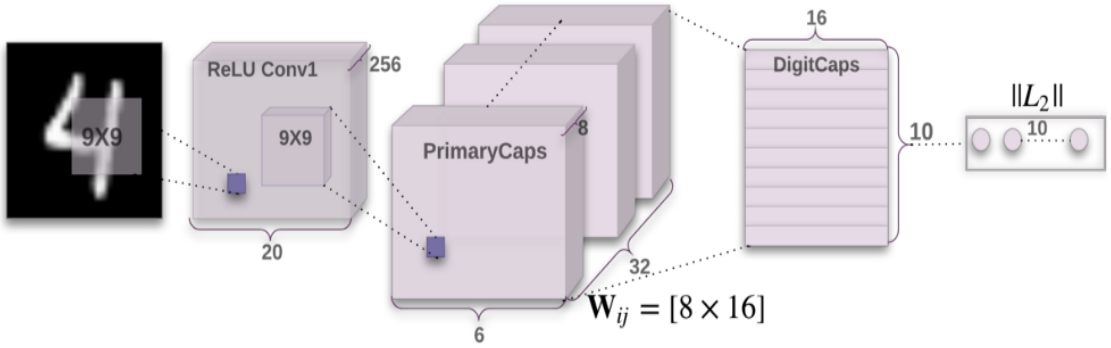


Figure 2.2: Capsule Network Architecture

2.2.3 Features:

Capsule Network offers numerous notable features:

- Partial relationships are captured by Capsule Networks, which inherently depict hierarchical aspects. This provides **advantages over CNNs in terms of handling intricate structures** more skillfully.
- Because of **dynamic routing**, Capsule Networks are excellent at identifying objects from a variety of perspectives. When handling different orientations, this performs better than CNNs.
- Capsule Networks use part-whole links to **enable good generalisation with less training examples**. Compared to CNNs, this lessens the reliance on data, increasing efficiency.

2.2.4 Drawbacks:

- Capsule networks can be **computationally taxing**, especially when dynamic routing is involved. This might affect training speed and efficiency.
- Compared to typical neural networks, capsule networks have not been widely used; this could be because of their **complexity and resource requirements.**
- On large datasets, Capsule Networks might find it **difficult to outperform regular networks**, which would restrict their usefulness in some situations.

2.2.5 Applications:

Capsule Networks are used in a variety of applications, including:

- **Plant Disease Diagnosis**: In agriculture, capsule networks are used to diagnose plant diseases. These networks help to detect and treat illnesses early by analysing photos of leaves and plant structures, boosting crop health and productivity.
- **Social Media Analysis**: Capsule Networks find use in analysing social media data, providing insights and patterns in content, and assisting in sentiment analysis and trend identification.
- **Product Recognition and Fraud Detection**: Capsule Networks are beneficial in a variety of sectors, including product recognition and fraud detection, ensuring reliable identification and validation.

Capsule Networks (CapsNets) excel in a variety of applications as mentioned above. CapsNets are useful for numerous applications demanding exact feature interpretation due to their unique hierarchical form.

2.3 Roboflow 3.0 Object Detection: YoLo v8

2.3.1 Introduction:

Roboflow 3.0 signifies a significant advancement in computer vision model development and deployment, offering a comprehensive suite of tools to enhance user experience, optimize model performance, and streamline workflow efficiency. The platform simplifies data preprocessing tasks, facilitates model training across various architectures, and provides robust tools for model evaluation and optimization. With seamless deployment options to diverse targets, Roboflow ensures widespread accessibility of models across different environments.

Introducing YOLOv8, the latest iteration of the renowned real-time object detection algorithm, *You Only Look Once (YOLO)*. YOLOv8 enhances accuracy, speed, and efficiency through cutting-edge improvements in network architecture and optimization strategies. Leveraging a deep convolutional neural network (CNN) as its backbone architecture, YOLOv8 ensures efficient feature extraction for precise object detection. Training involves meticulous optimization of model parameters, resulting in minimized detection errors. With optimizations in network design and hardware acceleration techniques, YOLOv8 maintains real-time performance while finding applications across domains like autonomous driving, surveillance, robotics, and industrial automation.

2.3.2 Working of YoLo v8:

YOLOv8 (You Only Look Once version 8) revolutionizes real-time object detection by combining speed and accuracy. Operating on a grid-based approach, YOLOv8 directly predicts bounding boxes and class probabilities from grid cells, ensuring swift detection. Its backbone architecture, typically a deep convolutional neural network (CNN), efficiently extracts features essential for precise object detection. Training involves optimizing parameters using techniques like backpropagation on large annotated datasets, aiming to minimize detection errors and improve metrics like precision and recall.

YOLOv8 prioritizes speed and efficiency through network design optimizations, inference algorithm enhancements, and hardware acceleration. By leveraging GPUs and specialized NPUs, YOLOv8 achieves rapid inference without compromising accuracy. This versatility finds application in diverse domains like autonomous driving, surveillance and robotics.

2.3.3 Understanding Key Components of YoLo v8:

- **Backbone Architecture:** YOLOv8 utilizes a deep convolutional neural network (CNN) as its backbone architecture, efficiently extracting features from input images essential for accurate object detection.
- **Grid-based Object Detection:** YOLOv8 operates by dividing the input image into a grid and directly predicting bounding boxes and class probabilities for objects within each grid cell. This grid-based approach enables swift detection of objects in real-time applications.
- **Training Methodology:** Training YOLOv8 involves optimizing model parameters using techniques like backpropagation and gradient descent on annotated datasets. This ensures the model learns to accurately predict object locations and classifications, minimizing detection errors.

The key components of YoLo v8 explained above can be seen through the illustration given in Fig. 2.3

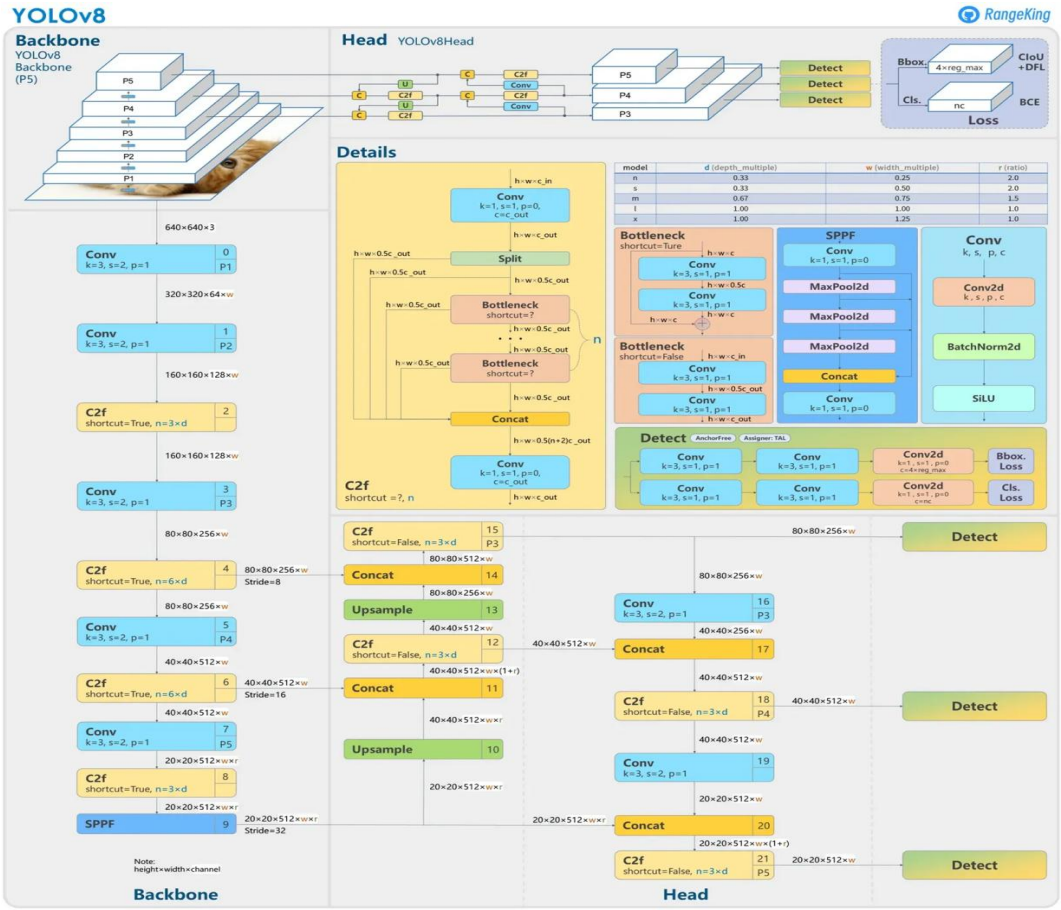


Figure 2.3: Architecture of the YoLo v8 Network

2.3.4 Steps Involved in YOLOv8:

1. **Input Processing**: An image is fed into the YOLOv8 network for object detection.
2. **Feature Extraction**: YOLOv8 utilizes a deep convolutional neural network (CNN) to extract features from the input image.
3. **Grid-based Detection**: YOLOv8 divides the image into a grid and predicts bounding boxes and class probabilities for objects within each grid cell.
4. **Optimization**: The model parameters are optimized through techniques like backpropagation and gradient descent using annotated datasets.
5. **Classification**: YOLOv8 predicts the class of each detected object.
6. **Localization**: Bounding box coordinates are refined by YOLOv8 for precise object localization.
7. **Output**: YOLOv8 generates a set of bounding boxes, each with a class label and a confidence score, indicating the probability of containing an object of the predicted class.

YOLOv8 excels in both speed and accuracy, making it suitable for real-time object detection applications. It achieves state-of-the-art performance across various benchmarks, demonstrating its effectiveness in diverse scenarios.

2.3.5 Features:

YOLOv8 boasts a range of remarkable features:

- **High Speed and Accuracy**: YOLOv8 achieves real-time object detection with exceptional precision, ensuring reliable results.
- **Single-stage Detection**: Simultaneously predicts object classes and precise locations in a single pass, providing comprehensive information.
- **Multi-scale Object Detection**: Capable of detecting objects of varying sizes by extracting features at multiple scales.
- **Robust to Data Imbalance**: YOLOv8 employs robust loss functions to handle imbalanced datasets, enhancing performance on challenging examples.

- **Versatility Across Environments**: YOLOv8 adapts to different environmental conditions, accommodating diverse object sizes and structures.
- **Efficiency in Training and Deployment**: Suitable for real-world applications, YOLOv8 demonstrates efficiency in both training and deployment phases.
- **Confidence Scoring**: Provides confidence scores for predictions, indicating the level of certainty in object detections.
- **Adaptability to Varied Imaging Conditions**: YOLOv8 operates effectively across various imaging circumstances, ensuring robust performance in different scenarios.

2.3.6 Drawbacks:

- **High Computational Demands**: YOLOv8 necessitates significant computational resources for both training and inference due to its complexity.
- **Complex Implementation**: Implementation of YOLOv8 may pose challenges for users unfamiliar with advanced deep learning models, requiring expertise in the field.
- **Limited Interpretability**: The intricate architecture of YOLOv8 makes it challenging to interpret decision-making processes, reducing its interpretability.
- **Susceptibility to Image Quality**: Performance of YOLOv8 may be impacted by variations in image quality, potentially leading to decreased accuracy.
- **Hyperparameter Sensitivity**: Achieving optimal performance with YOLOv8 is heavily dependent on selecting appropriate hyperparameters, posing challenges for users.
- **Data Quantity Requirements**: YOLOv8 relies on extensive labeled data for training, limiting its applicability in scenarios with limited data availability.

2.3.7 Applications:

YOLOv8 finds applications across various domains, including:

- **Object Detection**: Identifies and locates objects in images, offering a versatile solution across different domains.

- **Surveillance and Security**: Enhances surveillance systems by enabling real-time detection of objects, bolstering security measures.
- **Autonomous Vehicles**: Contributes to the development of autonomous vehicles by accurately detecting and tracking objects on roads.
- **Medical Imaging**: Assists in medical diagnostics by detecting abnormalities in images, aiding in disease diagnosis and treatment planning.
- **Industrial Automation**: Integrates into manufacturing processes for quality control, facilitating defect detection and prevention.
- **Agricultural Applications**: Supports precision agriculture by identifying and managing crop diseases, optimizing yield and resource utilization.

2.3.8 Summary

In this comprehensive project, we leveraged the power of YOLOv8 to advance the realm of precision agriculture. Our model was trained to accurately detect various crop diseases, ensuring timely identification and mitigation to promote crop health and productivity.

This report highlights the successful integration of deep learning models into agricultural practices, offering significant benefits such as early disease detection. Utilizing platforms like Google Colaboratory facilitated the development process, streamlining model training and deployment.

Ultimately, our precision agriculture system exemplifies the transformative impact of technology on the farming industry, driving efficiency and sustainability. By embracing innovative solutions like YOLOv8, we pave the way for a more resilient and productive agricultural sector, poised to meet the challenges of tomorrow.

Chapter 3

DATASET ANALYSIS

3.1 Dataset Description

The datasets were obtained from *Kaggle* [11] [12] and *RoboFlow* [13].

3.1.1 Plant Village Dataset

The **Plant Village Dataset** is renowned for its extensive use in *plant disease identification*. It offers a collection of over **54,300 high-quality images spanning 26 disease categories** (as shown in Fig. 3.1), making it a rich and diverse dataset. This dataset has been curated with precision by *David Hughes* and *Marcel Salathe* at *Cornell University* in New York, USA, ensuring its trustworthiness. Thanks to its well-labeled content and diversity, *this dataset is an excellent choice for researchers in machine learning and agricultural projects*, providing a reliable foundation.

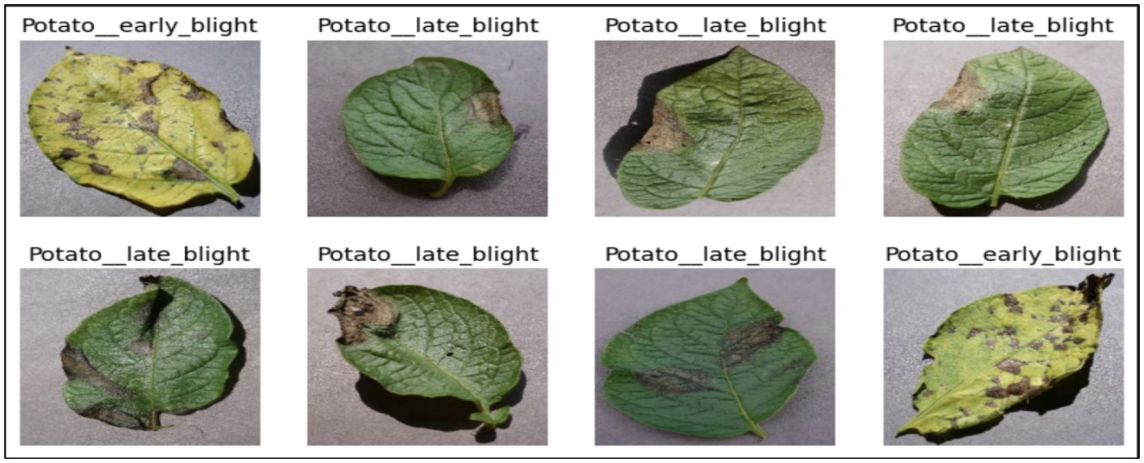


Figure 3.1: Potato Plant Images from Plant Village Dataset

3.1.2 New Plant Diseases Dataset

The **"New Plant Diseases Dataset"** (Fig. 3.2) has gained increasing popularity in the field of *plant disease research*, primarily due to its continuously expanding and updated image collection. This dataset encompasses high-quality images of various plant species, including both **healthy plants** and those affected by diseases, offering a comprehensive and versatile resource.

Typically, the *images are of high resolution and come with detailed annotations, complemented by valuable metadata such as plant species and disease type*. This feature makes it an invaluable asset for *machine learning applications*. The dataset is a collaborative effort involving various researchers and agricultural organizations, ensuring its continued relevance in contemporary disease identification projects.

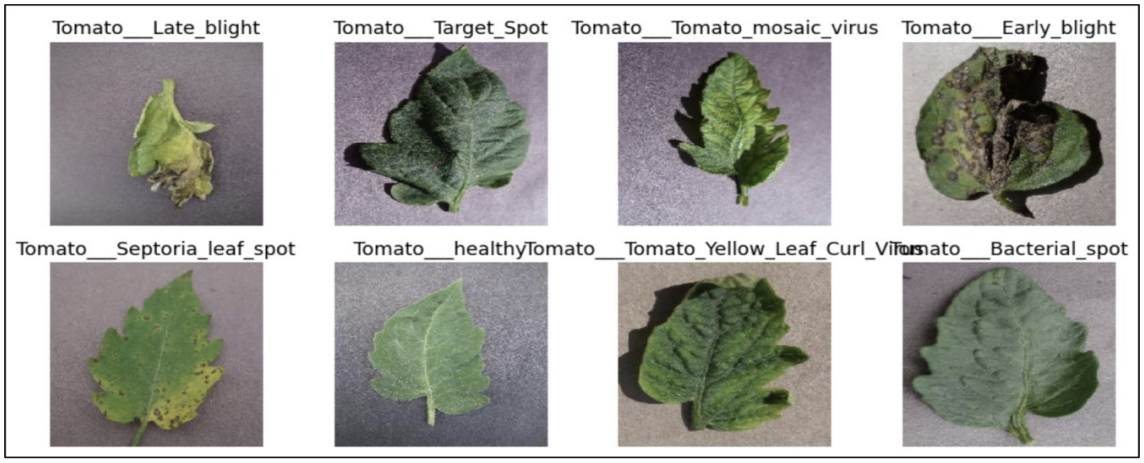


Figure 3.2: Tomato Plant Images from New Plant Diseases Dataset

3.1.3 PlantDoc Dataset

The **PlantDoc dataset** (Fig. 3.3), developed at the *Indian Institute of Technology (IIT)*, is an important tool for identifying plant diseases. **It contains 2,569 images representing 13 plant species and 30 classes**, allowing for easier image classification and object detection. *Curated by Pratik Kayal, the dataset is available on GitHub and comprises 8,851 labels, facilitating nuanced analysis of plant health*. Recognizing the significant *economic impact of plant diseases, estimated at $220 billion per year* worldwide, the dataset prioritizes early disease detection, potentially increasing crop yield. PlantDoc is utilized for benchmarking in *agricultural computer vision* and trains models such as **MobileNet, Faster-RCNN, VGG16, InceptionV3, and InceptionResNet V2**, making it an invaluable resource for a wide range of research applications in the field.

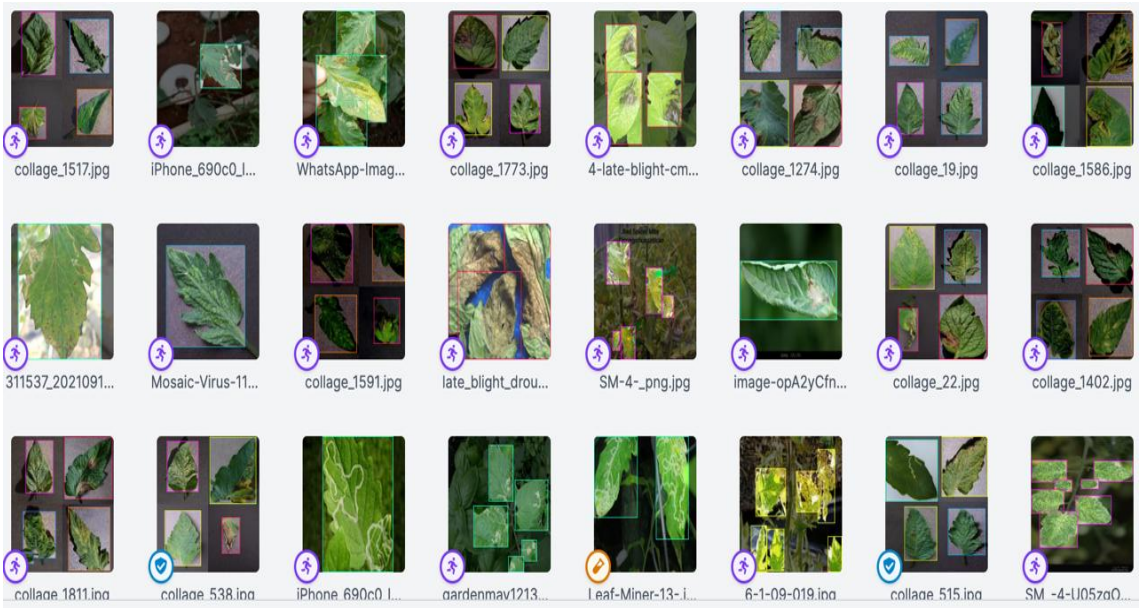


Figure 3.3: Plant Images from RoboFlow Dataset for Object Detection

Table 3.1: Dataset Features

S.No	Subject	Tomato Plant
1	No. of Diseases	9
2	No. of Images	11000

3.2 Working on the Dataset

3.2.1 Data Preprocessing and Augmentation

In the field of *machine learning*, efficient data preprocessing is critical. The code uses the `ImageDataGenerator` class to carefully curate the dataset for training. This class excels at data augmentation, a technique that dynamically improves a dataset to aid model learning. To address the various blurs encountered in drone-captured images, **Gaussian blur** (Fig. 3.4) and **Motion blur** (Fig. 3.5) have been incorporated into the preprocessing pipeline. *Scaling pixel values to ensure they uniformly span the normalized range of 0 to 1 is an important aspect of deep learning model convergence and performance.*

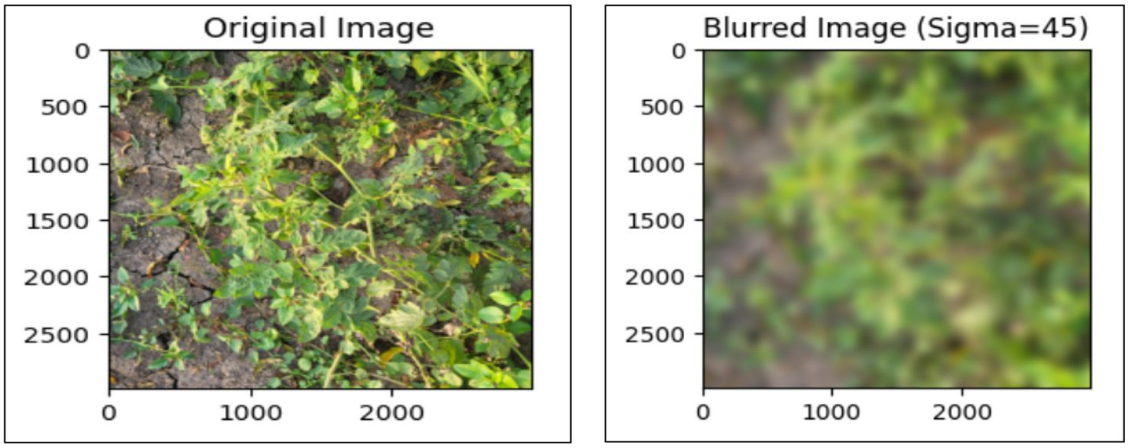


Figure 3.4: Data Preprocessing - Gaussian Blur application

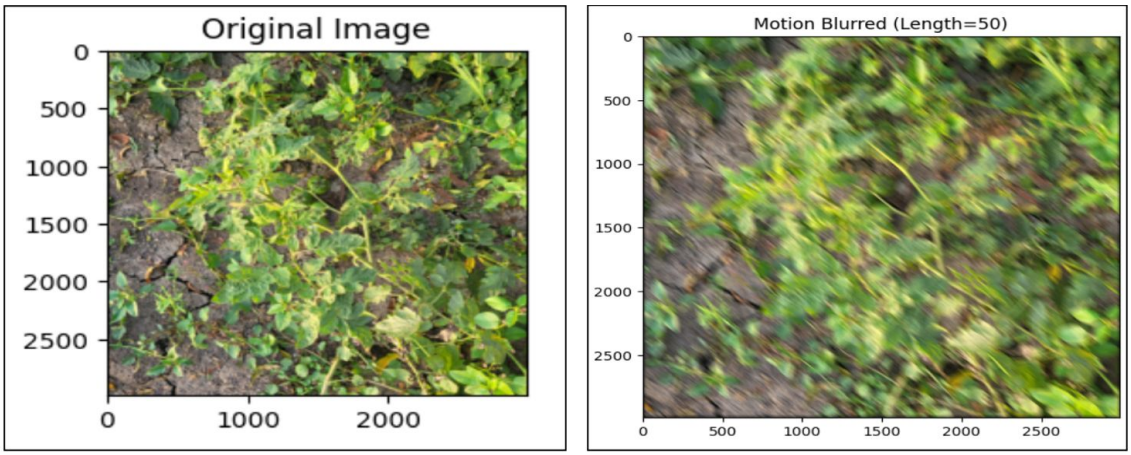


Figure 3.5: Data Preprocessing - Motion Blur application

In addition to traditional techniques, the preprocessing strategy employs Gaussian blur and motion blur to handle the various blurs present in images captured by a drone-mounted camera. Gamma correction is applied to adjust the lighting in leaf images (Fig. 3.6), while diverse zoom and shear transformations account for variations in the distance between the drone and the captured image. Flip and rotation range augmentations handle multiple orientations, resulting in a more robust dataset (Fig. 3.7). The dataset is stratified, with 80% for training and 20% for validation, allowing for the evaluation of the model's ability to generalize to new and previously unseen data.

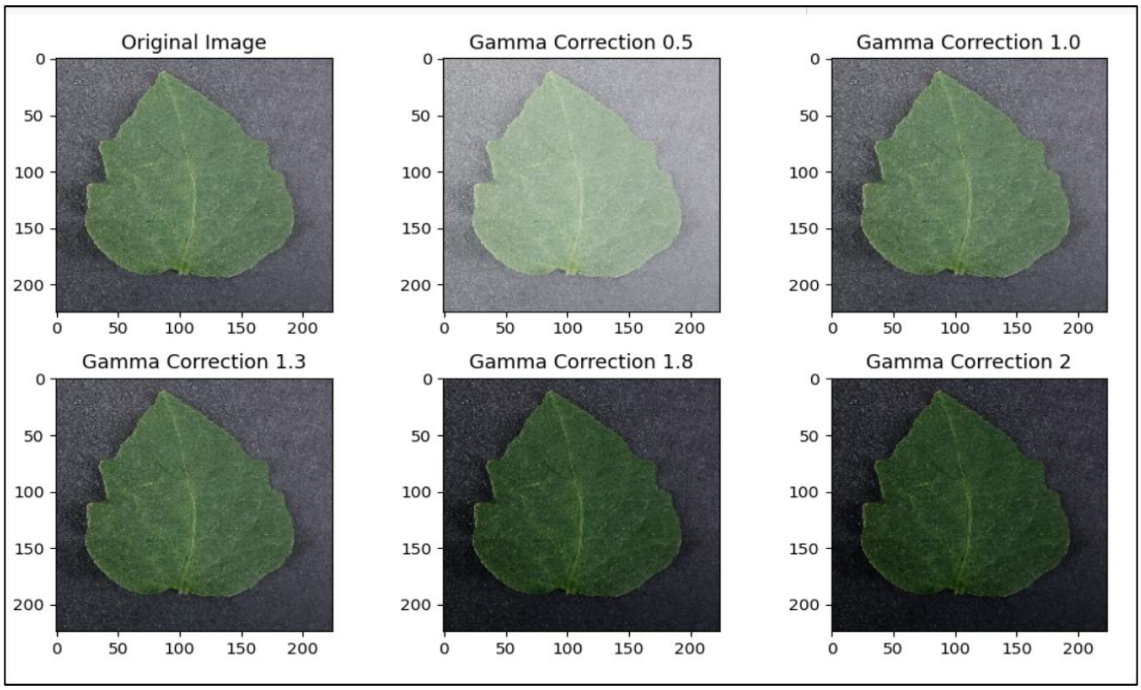


Figure 3.6: Data Augmentation - Gamma Correction

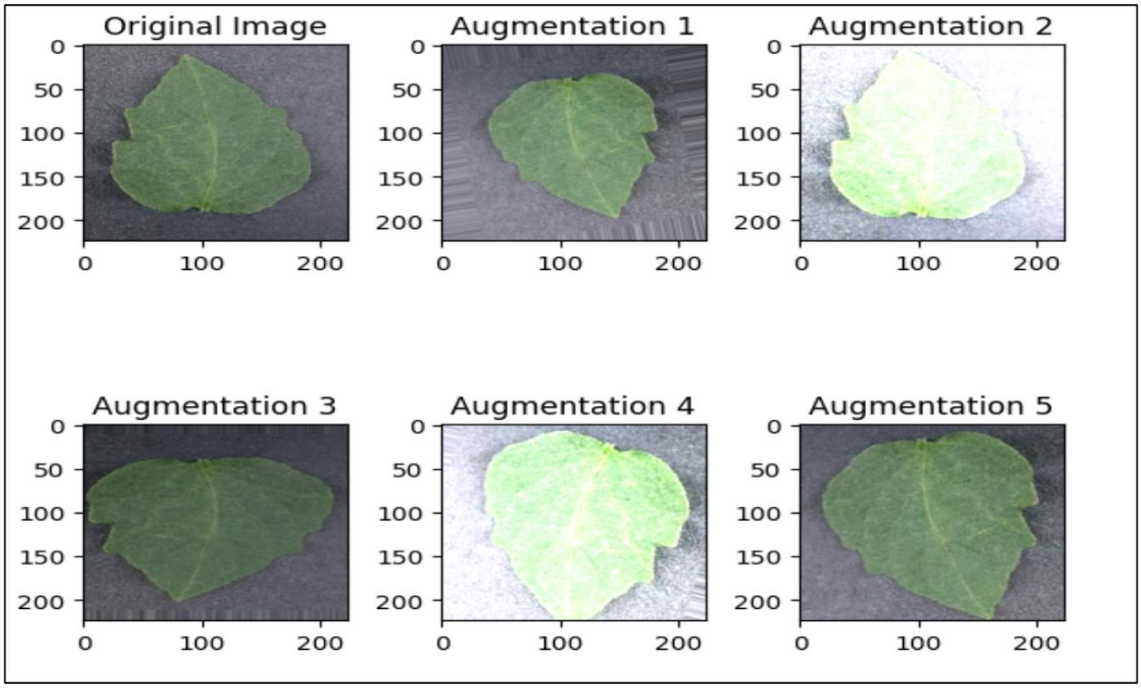


Figure 3.7: Data Augmentation - Dynamic Image Augmentation

3.2.2 Model Architecture

The model architecture employs a *Capsule Network*, which is implemented using the **Keras Sequential Model**. It is made up of a convolutional layer, primary capsules, digit capsules, and a fully connected layer for classification. The convolutional layer uses ReLU activation, while the output layer assigns class probabilities using the softmax activation function. The model is built using categorical crossentropy loss and the Adam optimizer. Data preprocessing entails image augmentation, such as rescaling, shear range, zoom range, and horizontal flip. Callbacks to reduce learning rate and stop early are part of the training process. The test set is used to calculate evaluation metrics like accuracy, confusion matrix, and classification report. The trained model is saved, and an example image is submitted for classification. The model architecture is illustrated in Fig. 3.8

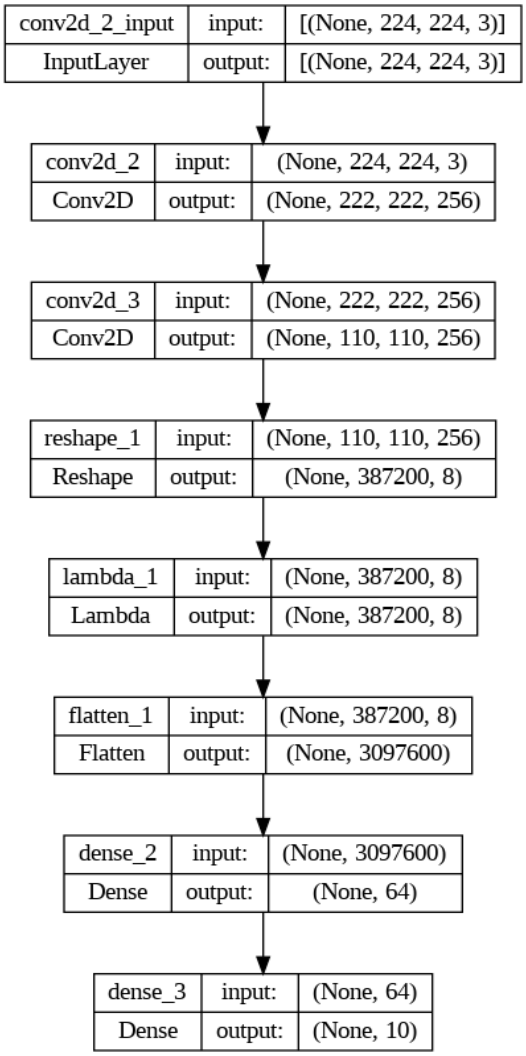


Figure 3.8: Capsule Neural Network Model Architecture

3.2.3 Model Training

Training, an iterative process, is the focal point of model development. Throughout training, the model's parameters are systematically adjusted to achieve precise predictions. The `fit` method orchestrates the training, utilizing the training dataset. Each epoch signifies one full cycle of training data, and at each step, the model learns to discern intricate patterns and features within the dataset. The process is punctuated by recurrent evaluations on the validation dataset to thwart overfitting. This dynamic evaluation guides the model to fine-tune its learning strategy. Training is a resource-intensive endeavor, but it is the bedrock for cultivating a model that can yield dependable image classifications.

3.2.4 Image Prediction

Upon completing its training journey, the model becomes equipped for image predictions. The code at this stage loads an image, subjecting it to the requisite preprocessing for compatibility with the model's input criteria. Subsequently, the model executes predictions. The predictions involve assigning the image to a specific class label based on the knowledge acquired during training. This step is pivotal as it furnishes insights into the model's practicality and its potential to make accurate classifications when presented with real-world images.

3.2.5 Confusion Matrix

A confusion matrix is a diagnostic tool that offers an in-depth assessment of a classification model's performance. In the context of this code, it serves as a mirror reflecting the model's efficacy in the realm of disease classification. The matrix comprises four distinct regions, each representing true positive, true negative, false positive, and false negative predictions. Its significance lies in the illumination of any confusion or misclassification between different disease classes. Through this matrix, the areas that necessitate improvement come to the forefront, ensuring a refined model.

3.2.6 Accuracy and Loss Plots

In the realm of machine learning, tracking the model's progression is essential for effective model development. The code employs two primary metrics: accuracy and loss. Visualized across the epochs, the accuracy and loss plots divulge significant insights into the model's journey. By monitoring the training and validation accuracy, we can discern whether the model tends to overfit or underfit. Furthermore, the loss curves indicate whether the model is converging to a minimum loss, a vital aspect of successful training. These plots serve as invaluable guides for the fine-tuning of hyperparameters and ascertaining the model's optimal training trajectory. They equip data scientists and machine learning practitioners with the means to ensure the model's efficiency and effectiveness.

3.2.7 Summary

In this chapter, we delved into the dataset used for training and evaluating the Capsule Neural Network (CapsNet) model for plant disease identification. We explored three key datasets: the Plant Village Dataset, the New Plant Diseases Dataset, and the PlantDoc Dataset, each offering unique characteristics and advantages for disease identification tasks. Following the dataset description, we discussed the preprocessing and augmentation techniques applied to enhance the quality and diversity of the training data. This included methods such as Gaussian blur, motion blur, gamma correction, and various geometric transformations. Subsequently, we examined the architecture of the CapsNet model, emphasizing its distinctive components and the training process involved. Additionally, we discussed image prediction, the use of confusion matrices for model evaluation, and the significance of accuracy and loss plots in tracking model performance. Overall, this chapter provides a comprehensive overview of the dataset, preprocessing steps, model architecture, and training process essential for effective plant disease identification using Capsule Networks.

Chapter 4

PLANT DISEASE DIAGNOSIS MODELLING

4.1 Introduction

Capsule Networks (**CapsNets**) represent a novel approach to image recognition, offering promising capabilities in preserving spatial hierarchies and facilitating effective classification. This study explores the architecture of CapsNets and their application in *image recognition tasks*.

4.2 Capsule Network Architecture

In our project, we implemented a Capsule Network model for image classification using **TensorFlow** and **Keras** within a Google Colab environment. **Capsule Networks** represent a novel architecture designed to address limitations of traditional convolutional neural networks (CNNs) in capturing hierarchical spatial relationships within images. Unlike CNNs, Capsule Networks employ *dynamic routing mechanisms* to efficiently model part-whole relationships among image features, enabling improved generalization and robustness to affine transformations.

The Capsule Network architecture used in this study consists of convolutional layers followed by **primary capsules** and **digit capsules**. The primary capsules are responsible for detecting simple visual patterns within localized receptive fields, while the digit capsules aim to encapsulate pose information and spatial relationships between higher-level features. Through *iterative dynamic routing*, the digit capsules refine their outputs based on agreement between lower-level capsules, facilitating efficient routing of information through the network.

By incorporating Capsule Networks into the *image classification pipeline*, we aim to leverage their unique ability to preserve spatial hierarchies and pose relationships within images. This enables the model to better capture complex visual structures and improve classification accuracy, particularly in scenarios involving occlusion, deformation, or viewpoint variations. Through empirical evaluation and comparison with traditional CNN architectures, we seek to demonstrate the efficacy of Capsule Networks in addressing challenges inherent in *image classification tasks*.

4.3 Encoder and Decoder Components

In the proposed Capsule Network architecture, a *sequential model* is instantiated using the **Keras framework**. The architecture comprises several layers tailored to extract and manipulate features hierarchically. Initially, **a convolutional layer with 256 filters and a kernel size of 3x3 is employed to convolve input images**, utilizing the *rectified linear unit (ReLU)* activation function to introduce non-linearity. Subsequently, **primary capsules are formed through another convolutional layer with 32*8 filters and a stride of 2x2 to ensure spatial down-sampling**. This layer, also activated by ReLU, aims to detect basic visual patterns within localized receptive fields.

Following the extraction of primary capsules, digit capsules are introduced to encapsulate pose information and spatial relationships among higher-level features. This is achieved by reshaping the output tensor and applying a lambda layer to *iteratively refine the capsule outputs* through dynamic routing mechanisms. Notably, the output shape is tailored to facilitate efficient routing and aggregation of information across capsule layers. Finally, **a fully connected layer with 64 neurons and ReLU activation is utilized for classification, followed by a softmax activation layer to produce class probabilities**.

Upon defining the Capsule Network architecture, the model is compiled using *categorical cross-entropy loss* and the *Adam optimizer*, with *accuracy* serving as the evaluation metric. This configuration enables the model to learn and optimize its parameters during training to effectively classify input images into distinct classes. Overall, the Capsule Network architecture leverages its unique design principles to capture hierarchical spatial relationships within images, facilitating improved generalization and robustness in *image classification tasks*.

The CapsNet architecture incorporates both encoder and decoder components, as depicted in Figures 4.1 and 4.2, respectively. The encoder captures hierarchical structures of features through convolutional and capsule layers, while the decoder reconstructs input images from learned representations.

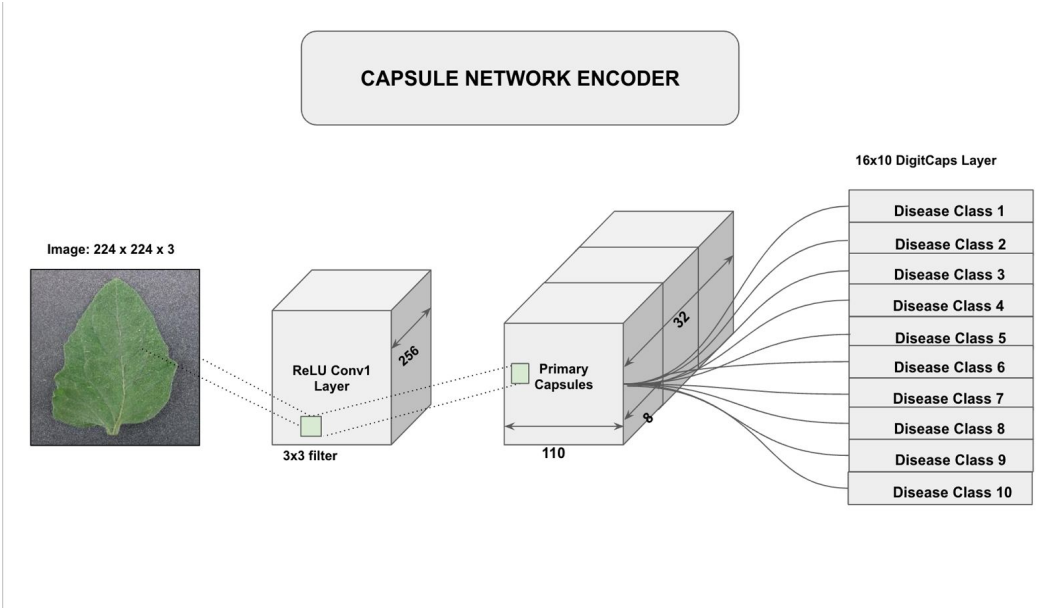


Figure 4.1: Design of the Encoder Component

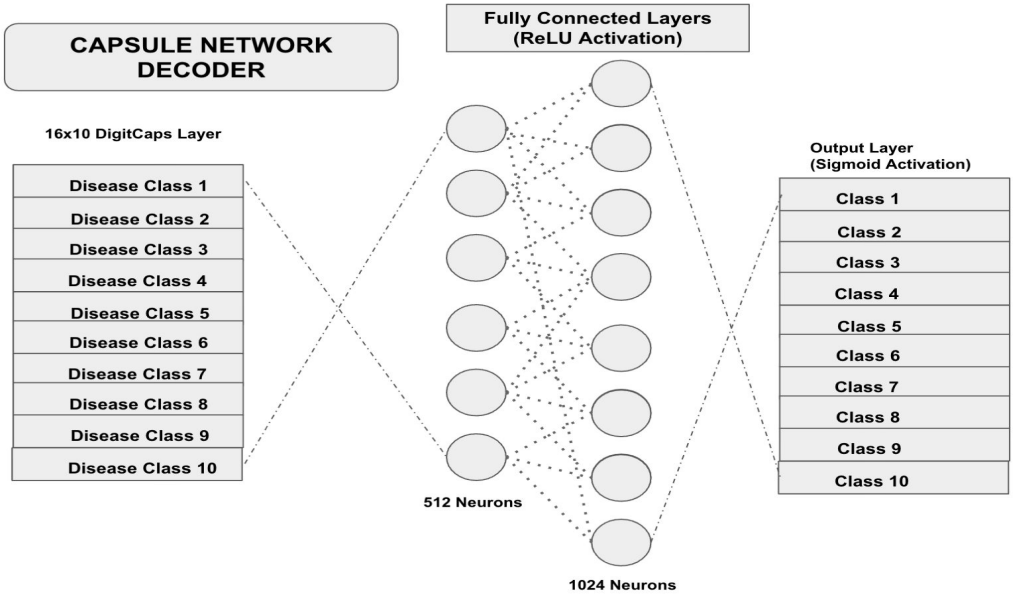


Figure 4.2: Design of the Decoder Component

4.4 Summary

The chapter explores Capsule Networks (CapsNets) as a novel approach to image classification, emphasizing their distinct architecture and advantages over conventional convolutional neural networks (CNNs). CapsNets, characterized by convolutional layers, primary capsules, and digit capsules, excel in capturing hierarchical spatial relationships and pose information within images, enhancing generalization and robustness. The encoder component comprises convolutional layers and primary capsules for feature extraction, while digit capsules encapsulate spatial relationships via dynamic routing mechanisms. A fully connected layer facilitates classification, followed by a softmax layer for probability estimation.

Employing categorical cross-entropy loss and the Adam optimizer, the model optimizes parameters during training. Decoder components complement CapsNets by reconstructing input images from learned representations, further extending their utility. Illustrated in Figures 4.1 and 4.2, the encoder and decoder designs elucidate the CapsNet's hierarchical feature extraction and reconstruction processes, underscoring its efficacy in complex image recognition tasks.

Chapter 5

OBJECT DETECTION

5.1 Yolo v8 Model

The architecure of YoLo v8 model is illustrated in Fig. **??** below:

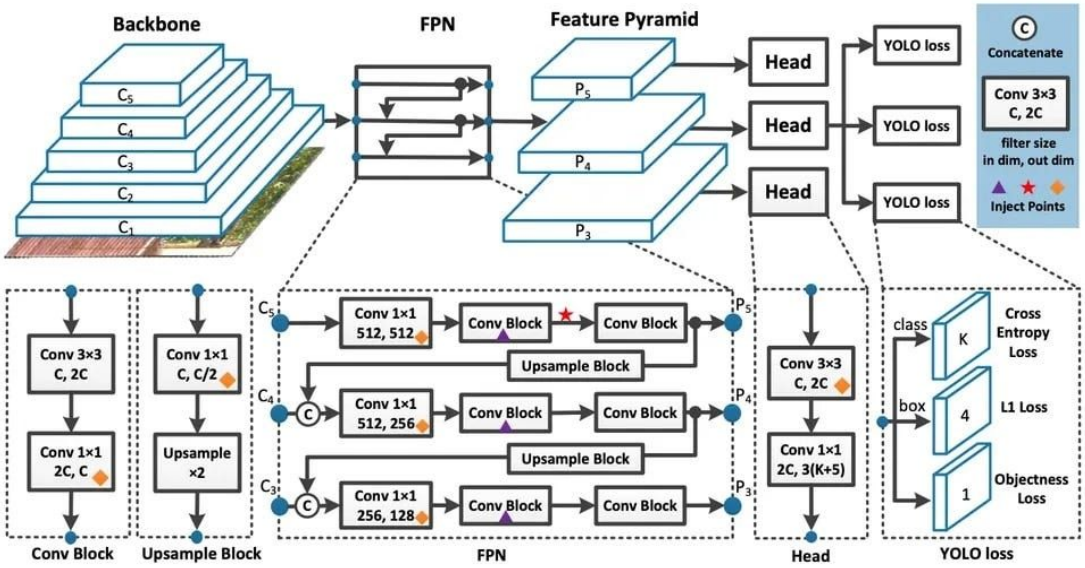


Figure 5.1: YoLo v8 Model Architecture

5.2 Data Preprocessing and Augmentation

The preprocessing and augmentation pipeline for YOLOv8 includes the following steps:

- **Auto-Orient**: Images are auto-oriented to ensure proper alignment.
- **Resize**: Images are resized to 640x640 resolution using the Fill method, with center cropping to maintain aspect ratio.

The augmentation settings for YOLOv8 are configured as follows:

- **Outputs per Training Example**: Each training example generates 3 augmented outputs.
- **Flip**: Horizontal and vertical flips are applied for data augmentation.

- **Saturation**: Saturation levels are adjusted between -25
- **Brightness**: Brightness levels are adjusted between -15
- **Blur**: Images are blurred with a maximum blur radius of 1.1 pixels to simulate motion or out-of-focus effects.

The YOLOv8 dataset attributes include:

- `Image/encoded`: Encoded binary representation of images in JPEG format.
- `Image/height`: Height of each image in the dataset.
- `Image/width`: Width of each image in the dataset.
- `Image/object/bbox/xmin`: X-coordinate of the top-left corner of the bounding box.
- `Image/object/bbox/xmax`: X-coordinate of the bottom-right corner of the bounding box.
- `Image/object/bbox/ymin`: Y-coordinate of the top-left corner of the bounding box.
- `Image/object/bbox/ymax`: Y-coordinate of the bottom-right corner of the bounding box.
- `Image/object/class/label`: Labels associated with each bounding box indicating the class of objects within the images.

The `BBOX_FORMAT` is set as ”xyxy” to indicate the bounding box format converted from relative XYXY to absolute XYXY.

5.3 YOLOv8 Model Description

The YOLOv8 model is characterized by its anchor-free architecture, representing a departure from traditional anchor-based object detection methods. Instead of predicting offsets from pre-defined anchor boxes, YOLOv8 directly predicts the center of an object (**see Figure 5.2**).

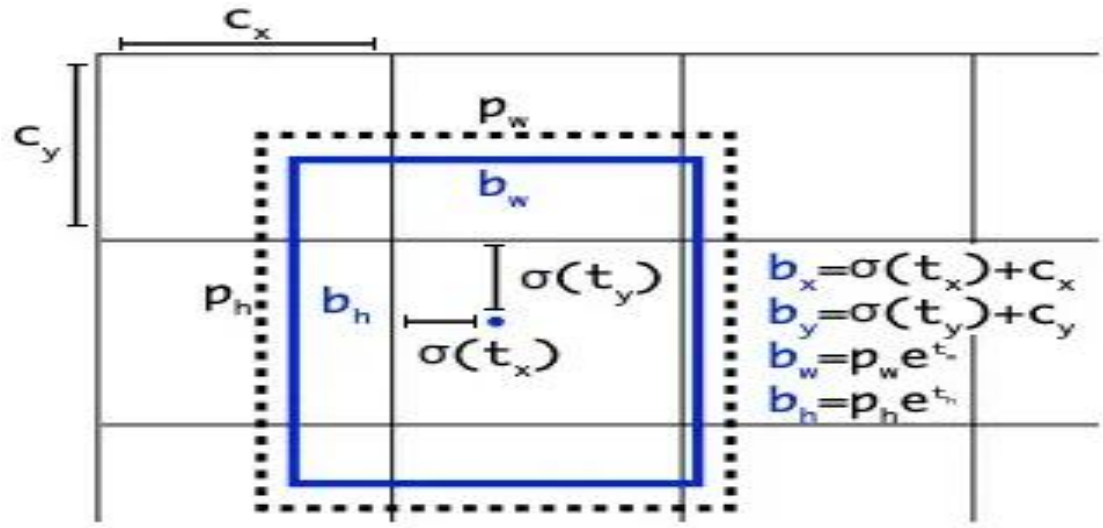


Figure 5.2: Anchor Box visualisation of YoLo v8 Model

5.3.1 Backbone or Feature Extractor

The backbone of YOLOv8 serves as the core of the model, responsible for extracting features from input images. It consists of layers that progressively capture features at different levels of abstraction, from basic shapes to more complex patterns. Notably, YOLOv8 employs a pre-trained YOLO model as its backbone solely for feature extraction, dissociating it from the object detection process.

5.3.2 Feature Pyramid Network (FPN)

YOLOv8 incorporates a Feature Pyramid Network (FPN) architecture to generate a hierarchical set of feature maps at different scales. This enables the detection of objects of varying sizes by integrating low-resolution, semantically rich features with high-resolution, semantically weaker features through a top-down pathway and lateral connections.

5.3.3 Task-Specific Subnetworks

Task-specific subnetworks in YOLOv8 operate independently on each level of the feature pyramid. One subnetwork is dedicated to object classification, while the other focuses on predicting bounding boxes. It's important to note that these subnetworks are not pre-trained and require training from scratch within the YOLOv8 framework (**see Figure 5.3**).

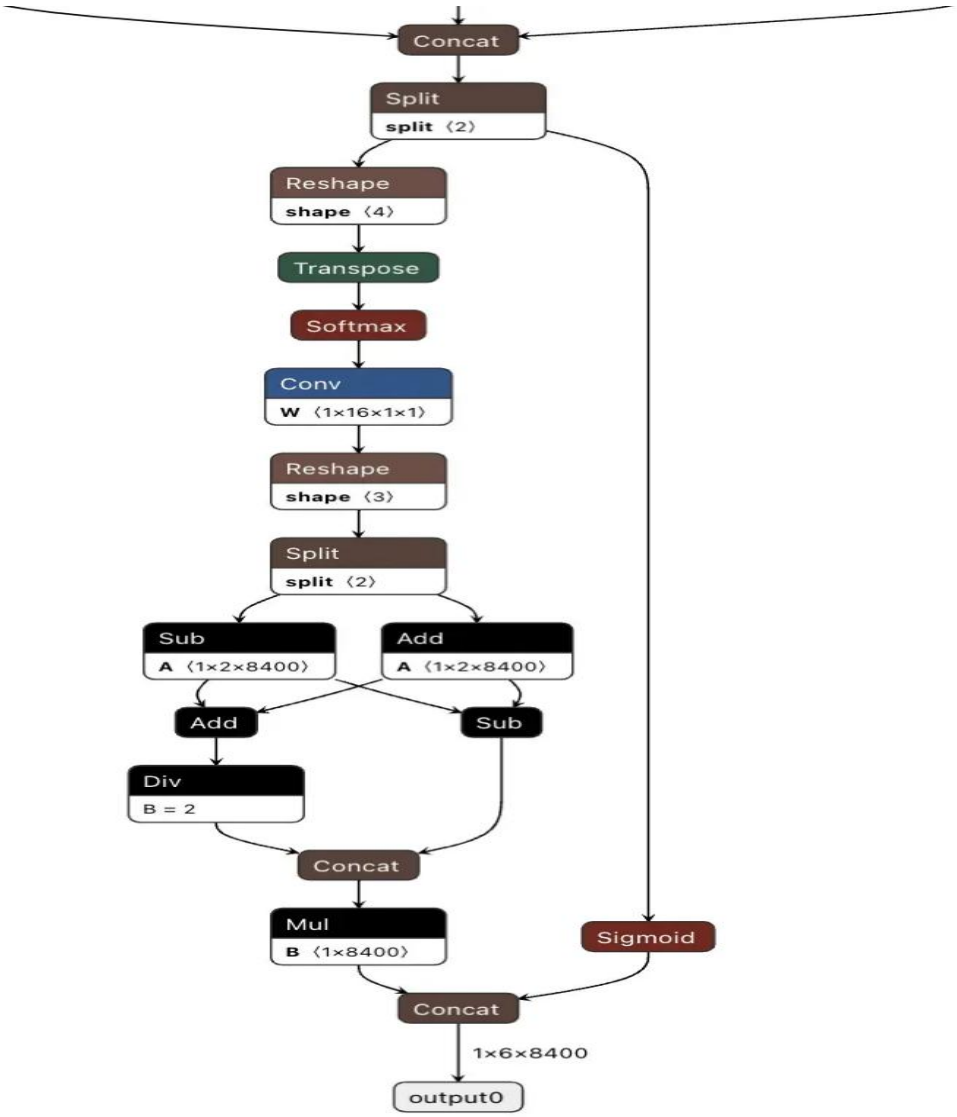


Figure 5.3: Detection Head for YoLo v8 Model

In summary, YOLOv8's anchor-free architecture and efficient feature extraction process make it a versatile and powerful model for object detection tasks.

5.4 Training Strategies

Throughout the training process of YOLOv8, various strategies are implemented to enhance efficiency and monitor performance:

Mosaic Augmentation: YOLOv8 augments images dynamically during training, presenting slight variations of the provided images at each epoch. One of these augmentations, mosaic augmentation, stitches four images together, exposing the model to objects in new locations, partial occlusions, and different surrounding pixels. However, empirical evidence suggests that this

augmentation can degrade performance if used throughout the entire training routine. Hence, it is advantageous to disable mosaic augmentation for the last ten training epochs.

Loss Function: The loss function embedded in YOLOv8 is a critical component in the model's training regimen. These meticulously crafted loss functions play an important role in honing the model's detection and classification capabilities. The loss function guides the model towards refining its predictions and improving overall performance by precisely assessing the differences between predicted and actual bounding boxes, as well as class labels.

Early Stopping: Training is terminated if the validation loss remains unchanged for six consecutive epochs, preventing overfitting.

Model Checkpoint: The model's validation loss is monitored after each epoch, and if the current loss surpasses that of previous epochs, the model weights are saved.

Lambda Callback: Custom Python functions, defined using lambda callbacks, are executed during training at various points within each epoch. These functions compute and print custom metrics such as training and validation loss, along with mean Average Precision (mAP), for ease of monitoring.

Visualization of Detections: After every five epochs, YOLOv8 generates a 4 by 8 grid of images with predicted bounding boxes. This visualization aids in evaluating the model's performance and improvement trends throughout the training process, with the expectation that visualizations become more accurate as training progresses.

5.5 Summary

This chapter details the architecture, data preprocessing, augmentation, and training strategies of YOLOv8 for object detection. YOLOv8 comprises a backbone, Feature Pyramid Network (FPN), and task-specific subnetworks. Preprocessing involves image orientation and resizing. Augmentation includes flips, saturation, brightness adjustments, and blur. Training employs online augmentation, early stopping, model checkpointing, lambda callbacks, and visualization of detections. YOLOv8's anchor-free design, effective feature extraction, and robust training strategies render it versatile for diverse object detection tasks.

Chapter 6

IMAGE PROCESSING TECHNIQUES

6.1 Introduction

Data preprocessing is a fundamental and critical step in the preparation of datasets for deep learning structures. The utilization of effective preprocessing techniques significantly impacts the performance and robustness of deep learning models. This section delves into the application of preprocessing techniques, with a specific focus on their implementation in the context of the Tomato Leaves dataset.

6.2 Data Preprocessing Techniques

The Tomato Leaves dataset comprises self-acquired images exhibiting variations in both dimensions and quality. Some images in the dataset are pristine, capturing the essence of healthy tomato leaves, while others may contain artifacts, anomalies, or unwanted noise. The objective of preprocessing is to enhance the dataset's quality, reduce input image noises, and ensure that the subsequent deep learning models trained on this data exhibit optimal performance.

6.2.1 Gaussian Blur

Gaussian blur stands as a prominent non-uniform low-pass filtering operation employed to mitigate input image noises and achieve a smoother representation of edges. The two-dimensional Gaussian kernel function, mathematically defined as:

$$Gaussian(x, y) = \frac{1}{2\pi r^2} exp\left(-\frac{x^2 + y^2}{2r^2}\right)$$

where r represents the standard deviation, and x and y are pixel locations, plays a crucial role in determining the degree of smoothness in and around pixels. By controlling the variance around an average value, r aids in reducing noise in high-frequency components. This smoothing effect proves beneficial for images with variations in pixel intensity, contributing to a more uniform and noise-free dataset.

6.2.2 Motion Blur

Motion blur is a technique applied to create a sense of movement or dynamism in images. It simulates the effect of rapid motion, resulting in blurred streaks along the direction of the motion. In the context of the Tomato Leaves dataset, the application of motion blur introduces diversity by mimicking real-world scenarios where the leaves might be affected by external factors such as wind or natural swaying.

The motion blur operation involves capturing an object in motion over a specific period, causing its appearance to smear across the image. Mathematically, the motion blur can be represented as the convolution of the image with a motion kernel. The motion kernel's length and orientation dictate the extent and direction of the blurring effect. The motion blur formula is given by:

$$MotionBlur(x, y) = \frac{1}{N} \sum_{i=0}^{N} \delta(x - i \cdot \cos(\theta), y - i \cdot \sin(\theta))$$

where N is the length of the blur, and θ is the angle of the blur. This intentional blurring of the images contributes to a more comprehensive dataset, preparing the deep learning model to handle instances where the leaves are not perfectly still.

Motion blur aids in creating a dataset that is more representative of the challenges posed by actual environmental conditions. It helps the model become robust to variations in leaf appearance caused by factors like wind or other dynamic elements.

6.2.3 Data Augmentation

Data augmentation serves as a pivotal strategy to combat overfitting challenges in deep learning models, especially when faced with limited training data. The techniques employed in data augmentation aim to artificially increase the dataset size by introducing variations and diversities, ensuring that the model generalizes well to different scenarios.

Gamma Correction

Gamma correction emerges as a key augmentation technique, addressing issues related to pixel intensity and illumination variations. This method involves adjusting the intensity of pixel values, contributing to enhanced model generalization. Gamma correction proves particularly valu-

able when dealing with real-world scenarios where images may encounter varying lighting conditions.

Flipping, Scaling, Color Augmentation, and Rotation

The augmentation process extends beyond gamma correction to encompass flipping, scaling transformations, color augmentation, and rotation. These transformations introduce diversity into the dataset, simulating different angles, orientations, and color variations that the model may encounter during real-world scenarios. The images, initially captured under consistent illumination, undergo these transformations, effectively expanding the dataset size.

6.3 Application to Tomato Leaves Dataset

Informed and inspired by previous research on Grape leaves [21], these preprocessing techniques are strategically applied to the Tomato Leaves dataset. The original dataset, characterized by varied images depicting tomato leaves in different conditions, undergoes a comprehensive preprocessing pipeline. The application of Gaussian blur and data augmentation techniques contributes to the enhancement of image quality, noise reduction, and an increase in dataset size.

6.3.1 Preprocessed Images

Original Image of the Plant:

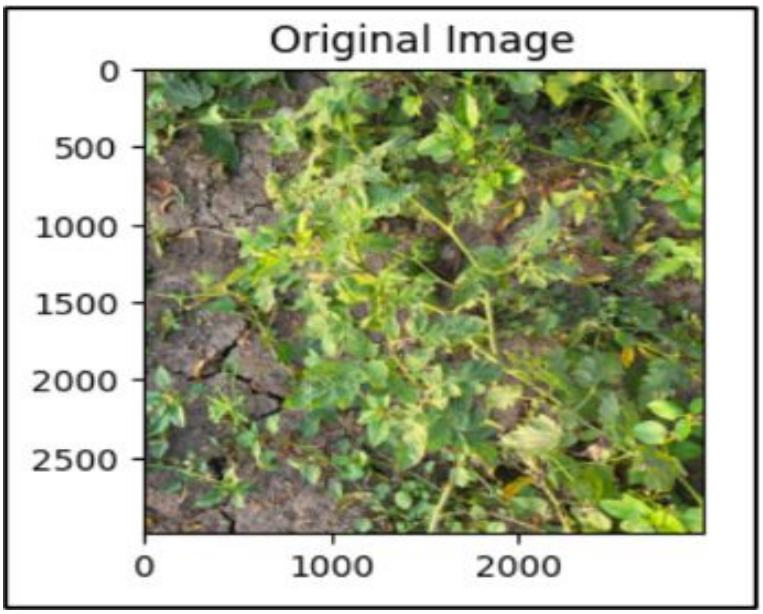


Figure 6.1: Original Image of the Plant

Image After Gaussian Blur:

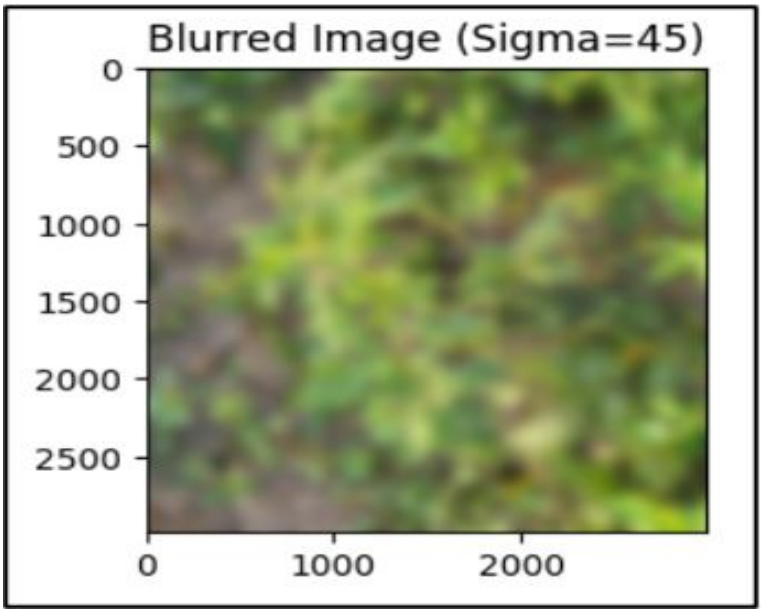


Figure 6.2: Image of the Plant After Gaussian Blur

Image After Motion Blur:

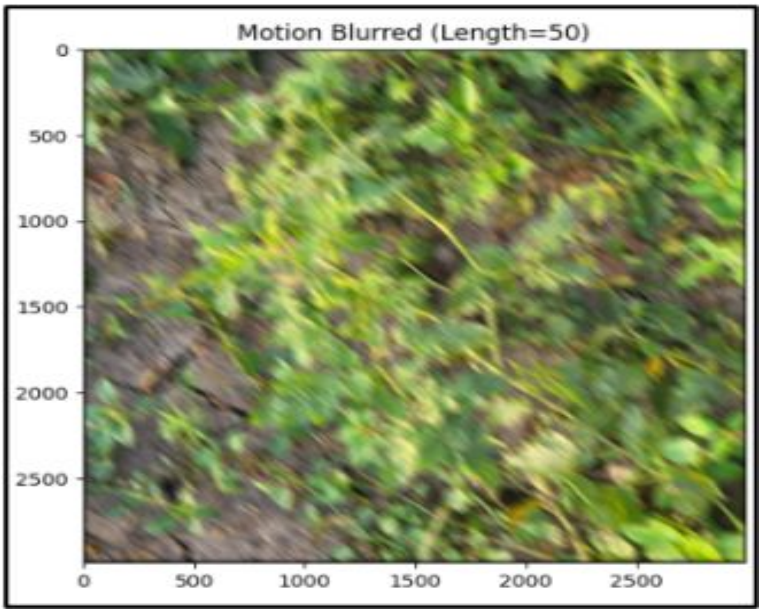


Figure 6.3: Image of the Plant After Motion Blur

Image After Gamma Correction:

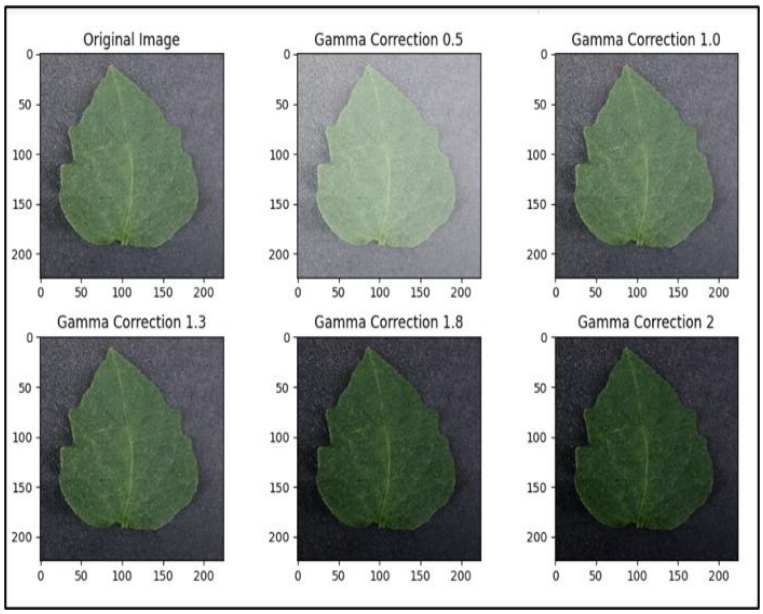


Figure 6.4: Image of the Plant After Gamma Correction

Image After Flipping, Zoom, and Gamma Correction:

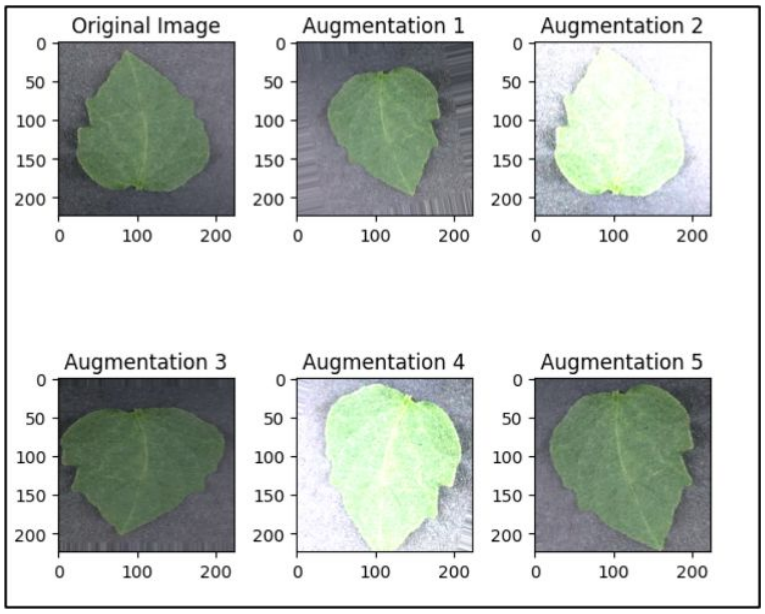


Figure 6.5: Image of the Plant After Flipping, Zoom, and Gamma Correction

6.4 Summary

The application of Gaussian blur and data augmentation techniques to the Tomato Leaves dataset holds paramount importance in the realm of deep learning model training and testing. The reduction of noise, improvement of image quality, and the augmentation of dataset size collectively contribute to the creation of a robust and diverse dataset. As a result, deep learning models trained on such enhanced datasets are better equipped to handle real-world scenarios, exhibiting improved generalization and performance.

Chapter 7

RESULTS

7.1 Capsule Neural Network Model:

7.1.1 Confusion Matrix

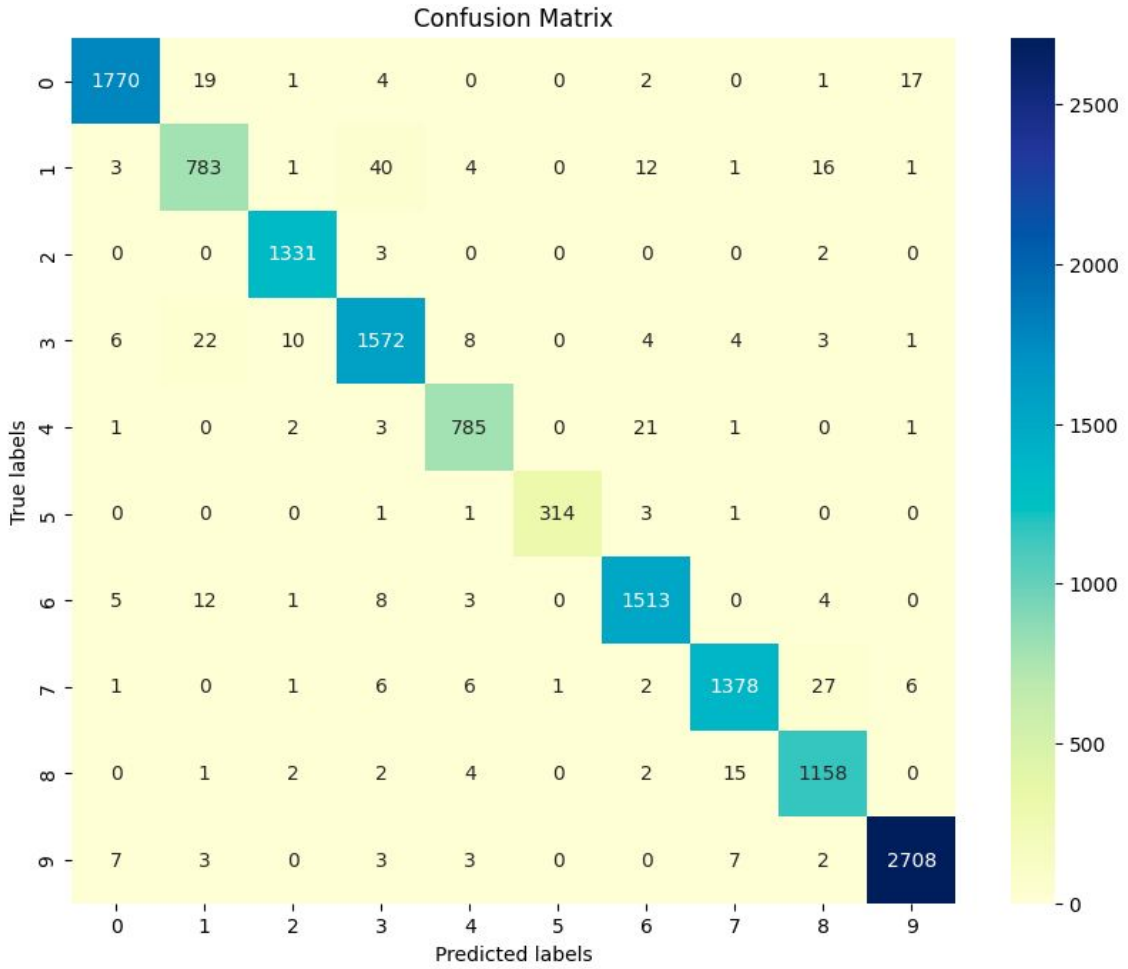


Figure 7.1: Confusion Matrix for CapsNet model

Class Distribution: The diagonal elements of the confusion matrix represent the instances where the predicted class aligns with the true class labels. Looking at the values along the diagonal in Fig. 7.1, we observe that the **model performs well for several classes, such as Tomato_bacterial_spot, Tomato_healthy, Tomato_late_blight, Tomato_leaf_mold, and Tomato_septoria_leaf_spot**, where the majority of predictions fall on the diagonal, indicating accurate classification. For instance, *Tomato_bacterial_spot has* 1770 *correct predictions out of* 1814 *instances (97.57%), showcasing a strong predictive capability*. Similarly, *Tomato_late_blight also demonstrates high accuracy with* 785 *correct predictions out of* 814 *instances (96.43%)*.

Challenges in Classification: While the model performs well for certain classes, it encounters difficulties in accurately classifying others. For instance, in Fig. 7.1 *Tomato_early_blight exhibits a lower accuracy compared to other classes, with only* 783 *correct predictions out of* 861 *instances (90.94%)*, indicating some misclassification or confusion with other classes. These challenges stem from overlapping features and insufficient training data since the disease are rare for certain classes. Misclassification primarily arises from the resemblance between symptoms of plant diseases and those indicative of nutrient deficiencies.

Misclassification Patterns: Analyzing off-diagonal elements provides insights into misclassification patterns within the model. For example in Fig. 7.1, *Tomato_early_blight shows misclassification with Tomato_late_blight, Tomato_leaf_mold, and Tomato_septoria_leaf_spot*, as indicated by the higher values in corresponding rows of the confusion matrix. Similarly, *Tomato_target_spot exhibits misclassification with Tomato_early_blight, Tomato_late_blight, and Tomato_spider_mites_(two_spotted_spider_mite)*, suggesting potential confusion between these classes. These misclassification patterns can guide model refinement efforts, such as feature engineering, data augmentation, or adjusting model hyperparameters to mitigate confusion between similar classes and improve overall performance.

7.1.2 Plot: Accuracy V/s No. of Epochs

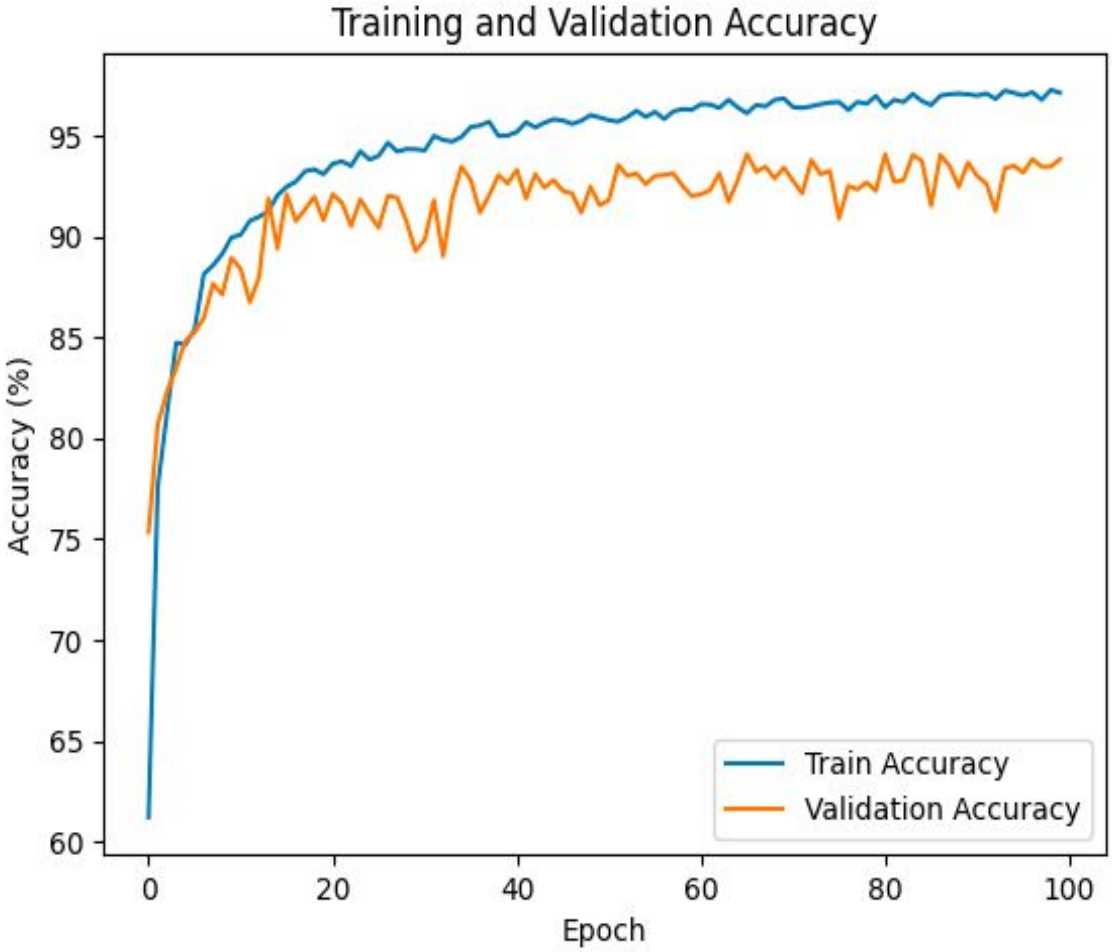


Figure 7.2: Plot: Accuracy V/s No. of Epochs for CapsNet model

The graph shown in Fig. 7.2, depicting the training and validation accuracy of our Capsule Neural Network model, reveals several significant observations. Notably, **the *training accuracy* consistently outperforms the *validation accuracy*, especially after 25 epochs**, indicating the model's proficiency in learning from the training data and its potential for making accurate predictions on previously unseen data.

Furthermore, the graph illustrates **a progressive upward trajectory with random fluctuations in *validation accuracy* over time until it reaches the plateau after 90 epochs**. This encouraging trend **signifies the model's ability to generalize effectively to novel data without succumbing to overfitting to the training dataset**.

In summary, **the CapsNet model designed offers an accuracy of *95.61%*** and the graph shown in Fig. 7.2, offers *compelling evidence of the CapsNet model's commendable performance, suggesting its suitability for making accurate predictions on new, real-world data*. It is imperative to acknowledge that the model's practical utility is intrinsically linked to the quality and representativeness of the training data. A faithful representation of real-world conditions in the training data is essential for the model to demonstrate robust performance in deployment.

7.1.3 Plot: Loss V/s No. of Epochs

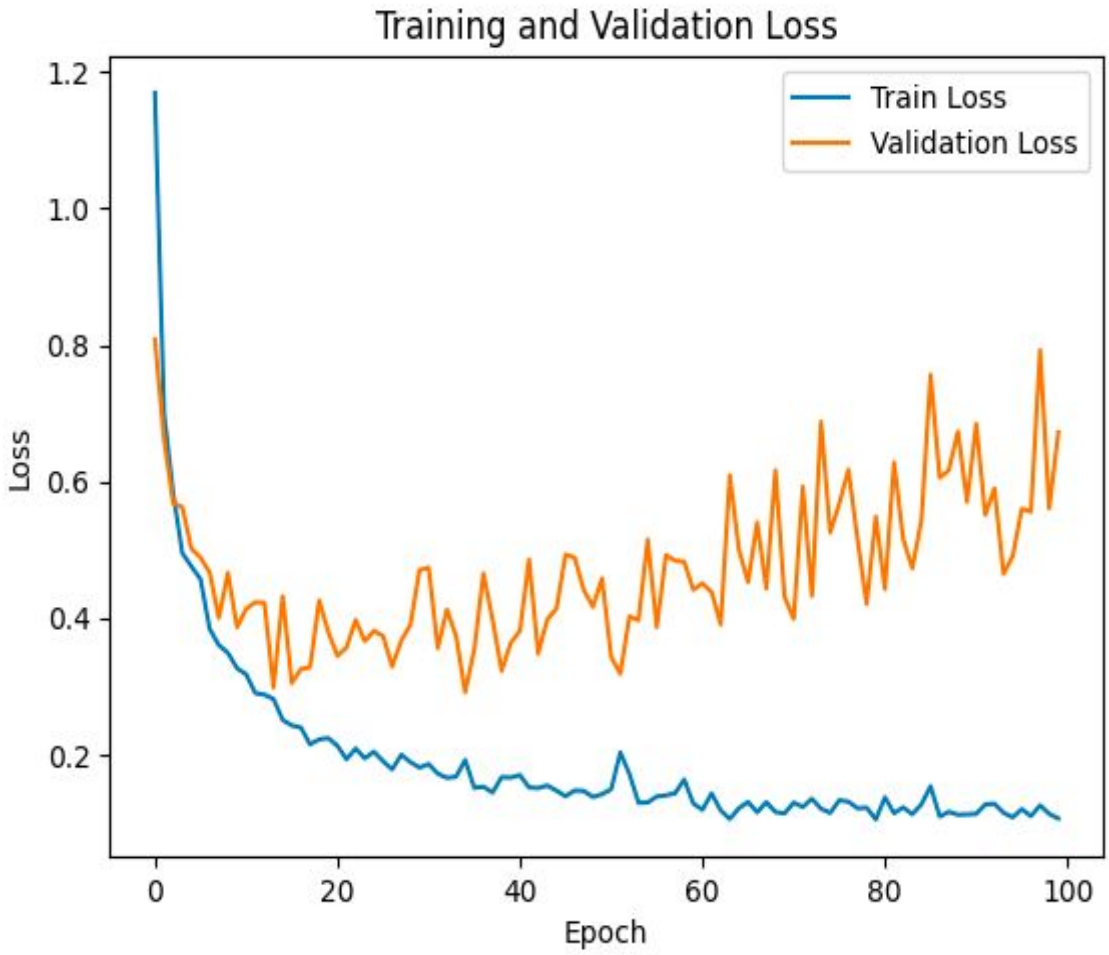


Figure 7.3: Plot: Loss V/s No. of Epochs for CapsNet model

In the graph illustrated in Fig. 7.3 of our Capsule Neural Network model's training and validation loss, several vital insights emerge. **The decreasing trend in *training loss* signifies that the model effectively learns from the training data, resulting in increasingly accurate predictions on the same dataset.**

However, **a concerning observation arises from the increasing *validation loss*, which is indicative of overfitting**. Overfitting materializes when the model becomes excessively attuned to the nuances of the training data, impeding its ability to generalize effectively to new data.

To mitigate overfitting, several strategies were employed, including:

- **Complexity:** Reducing the model's complexity.
- **Regularization:** Implementing regularization techniques such as L1 and L2 regularization.
- **Data Augmentation:** Augmenting the training dataset with additional data.
- **Dynamic Augmentation:** Leveraging data augmentation to create synthetic training data.

Further insights gleaned that by reducing the overfitting of the training data, the model's testing accuracy can be increased from 97.41% further to 98.50% or 99%. This prompts the need for enhancing the model's performance on new data by addressing overfitting concerns. **Ultimately, successful reduction of overfitting should yield a subsequent decline in validation loss, thereby enhancing the model's efficacy in handling new data**.

7.1.4 Classification Report

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.98	0.98	1814
1	0.93	0.91	0.92	861
2	0.99	1.00	0.99	1336
3	0.96	0.96	0.96	1630
4	0.96	0.96	0.96	814
5	1.00	0.98	0.99	320
6	0.97	0.98	0.97	1546
7	0.98	0.96	0.97	1428
8	0.95	0.98	0.97	1184
9	0.99	0.99	0.99	2733
accuracy			0.97	13666
macro avg	0.97	0.97	0.97	13666
weighted avg	0.97	0.97	0.97	13666

Figure 7.4: Classification Report for CapsNet model

The provided classification report illustrated in Fig. 7.4 offers a comprehensive evaluation of the Capsule Neural Network model's performance across ten classes of plant diseases. With an **overall accuracy of 97.5%, the model demonstrates robust classification capabilities**. Each class exhibits **high precision, recall, and F1-score values**, indicating the model's ability to accurately identify instances belonging to each class.

Particularly noteworthy is the **model's strong performance in classes with higher support**, such as classes 0, 2, 3, 6, 7, 8, and 9, where *precision, recall, and F1-score values consistently exceed 0.95*. This suggests that the **model effectively identifies instances of these diseases with high accuracy and minimizes both false positives and false negatives**.

While the model performs admirably across most classes, *slight variations are observed in the precision, recall, and F1-score metrics for classes with lower support, such as classes 1, 4, and 5.* Despite this, the **overall performance of the model remains impressive, with weighted averages of precision, recall, and F1-score all exceeding 0.97**.

To sum all the facts, the classification report underscores the efficacy of the Capsule Neural Network model in accurately classifying plant diseases. **Its high precision, recall, and F1-score values across most classes highlight its potential for real-world application in agricultural settings, where timely and accurate disease detection is critical for crop management and yield optimization**.

7.2 Results of YoLo v8 Model

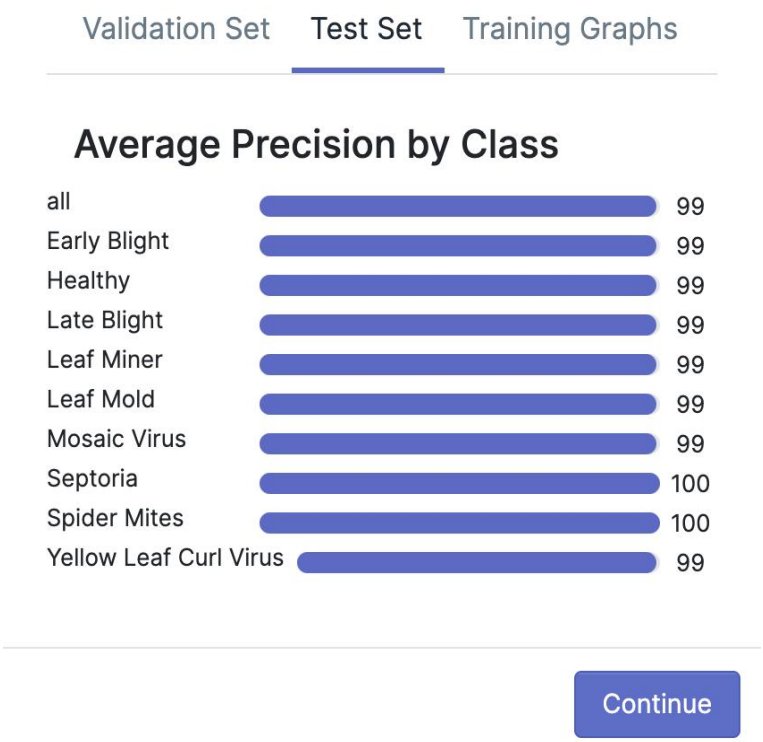


Figure 7.5: Average Testing Precision Results by Class for YoLo v8 model

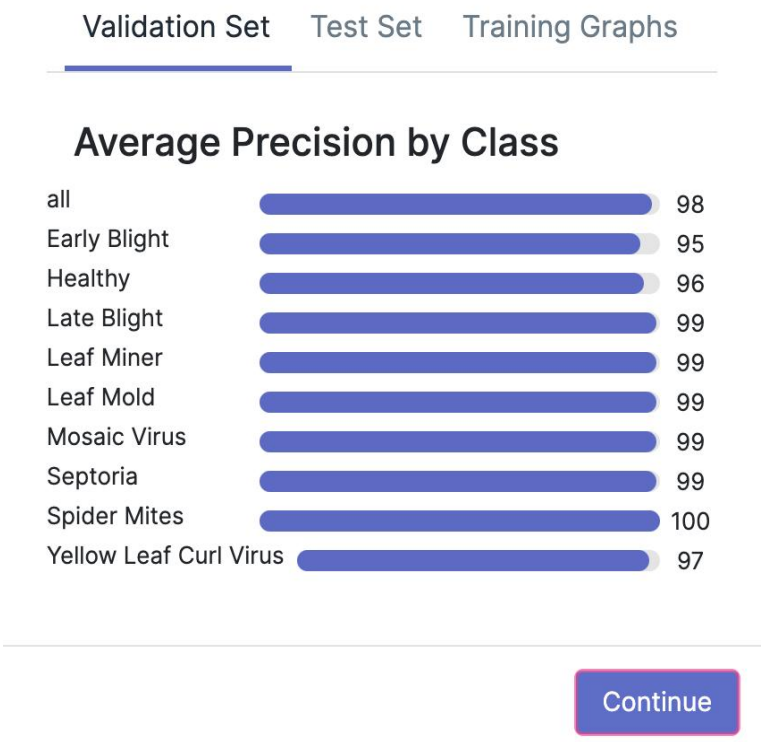


Figure 7.6: Average Validation Precision Results by Class for YoLo v8 model

7.2.1 Training Results of YOLOv8 Model

The training results of the YOLOv8 model are depicted in Figure 7.7. Different lines in the graph represent various loss functions that the model aims to minimize during training:

Box Loss: Measures the disparity between the predicted bounding boxes and the ground truth bounding boxes.

Cls Loss: Quantifies the model's accuracy in classifying objects correctly.

Dfl Loss: Specific to YOLO models, it combines classification and localization losses.

The loss curves illustrate the model's performance improvement on the training set as the training iterations progress. Ideally, these curves should consistently decrease over time.

Additionally, the bottom-left corner of the image displays the model's mean Average Precision (mAP) on a validation set. mAP is a vital metric for evaluating object detection models, considering both precision (correct detections) and recall (detected objects out of all actual ones).

The increasing trend of mAP throughout training suggests the model's effective generalization to unseen data.

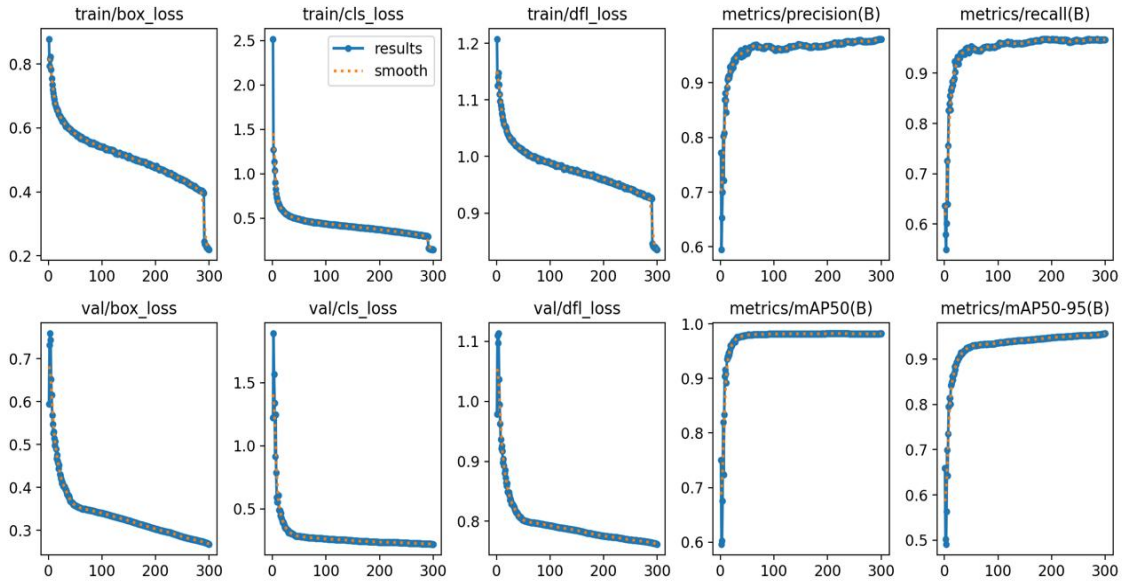


Figure 7.7: Training Graphs for trained YoLo v8 model

7.2.2 mAP Results of YOLOv8 Model

The mAP results of the YOLOv8 model are visualized in Figure 7.8. The line in the graph is purple and exhibits an upward trend throughout. The y-axis is labeled ”mAP,” ranging from 0.55 to 1.00, while the x-axis is labeled ”Epochs,” ranging from 0 to 300. At the bottom-left corner of the graph, there is text indicating ”1.00.”

The graph portrays that the model’s mean Average Precision (mAP) increases as training progresses, indicating effective generalization to unseen data.

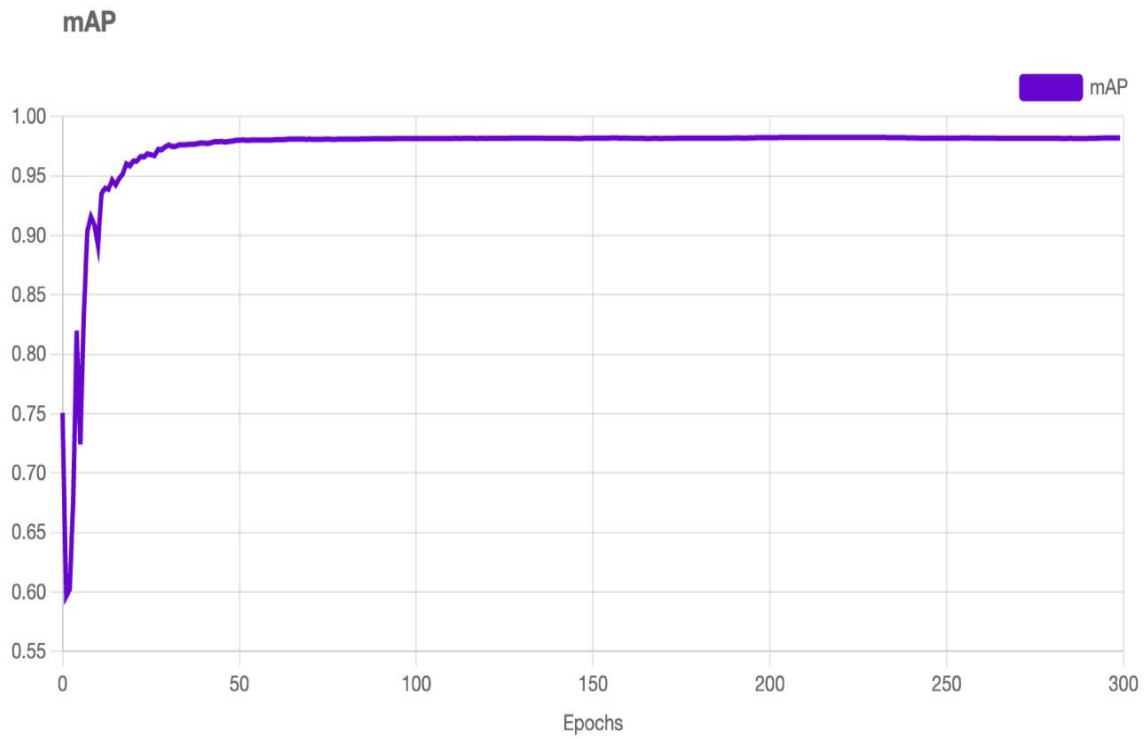


Figure 7.8: mAP Results of YOLOv8 Model

Loss Plots of Box, Class and Object for YoLo v8 Model

The loss plots for Box, Class, and Object of the YOLOv8 model are depicted in Figure 7.9. Here's a brief description of each loss function:

- **Box Loss**: Measures the deviation between predicted bounding boxes and ground truth bounding boxes. Lower box loss indicates improved accuracy in predicting the size and location of bounding boxes.
- **Class Loss**: Evaluates the likelihood of correct object classification. Lower class loss signifies enhanced classification accuracy.
- **Object Loss**: Represents a combination of classification and localization losses specific to YOLO models. Lower object loss indicates fewer overall mistakes by the model.

Ideally, all three loss curves should exhibit a consistent decrease over time as the model undergoes training. This indicates that the model is progressively improving its object detection capabilities.

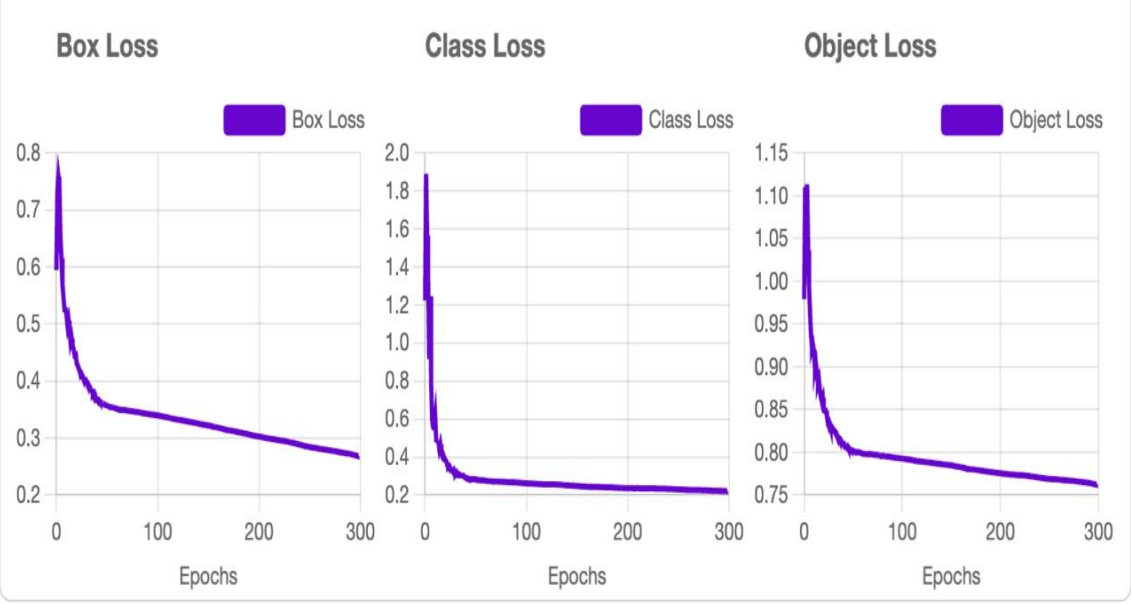


Figure 7.9: Loss Plots of Box, Object and Class of YOLOv8 Model

7.2.3 Overall Result Metrics of Trained YoLo v8 Model

- **mAP (mean Average Precision)**: A commonly used metric for evaluating object detection models, mAP considers both precision (correct detections) and recall (actual objects detected). A higher mAP indicates better performance.
- **Precision**: Measures the proportion of correct detections out of all detections made by the model.
- **Recall**: Measures the proportion of actual objects correctly detected by the model.

The high values in the table suggest that the YOLOv8 model is performing well, with high mAP, precision, and recall. This indicates accurate detection of a significant proportion of objects in the test dataset.

Figure 7.10: Overall Result Metrics of the trained YOLOv8 Model

7.3 Combined Model Results - WorkFlow

- **Combining YOLO object detection** with **Capsule Network** classification offers a synergistic approach to enhance object understanding and accuracy.
- **YOLO** swiftly detects objects but lacks detailed insights, which are provided by **Capsule Network**'s discernment of hierarchical feature relationships.
- Integrating **YOLO**'s bounding boxes with **Capsule Network**'s detailed classification refines object recognition, minimizing false positives and elevating detection accuracy.
- **Capsule Networks** excel in capturing spatial relationships and pose intricacies within objects, making them ideal for analyzing cropped regions of interest (**ROIs**) identified by **YOLO**. Targeting only the **ROIs** marked by **YOLO** reduces computing needs, speeding up inference, optimizing resource usage, and proving beneficial for tasks requiring quick analysis or running on resource-constrained devices like drones.

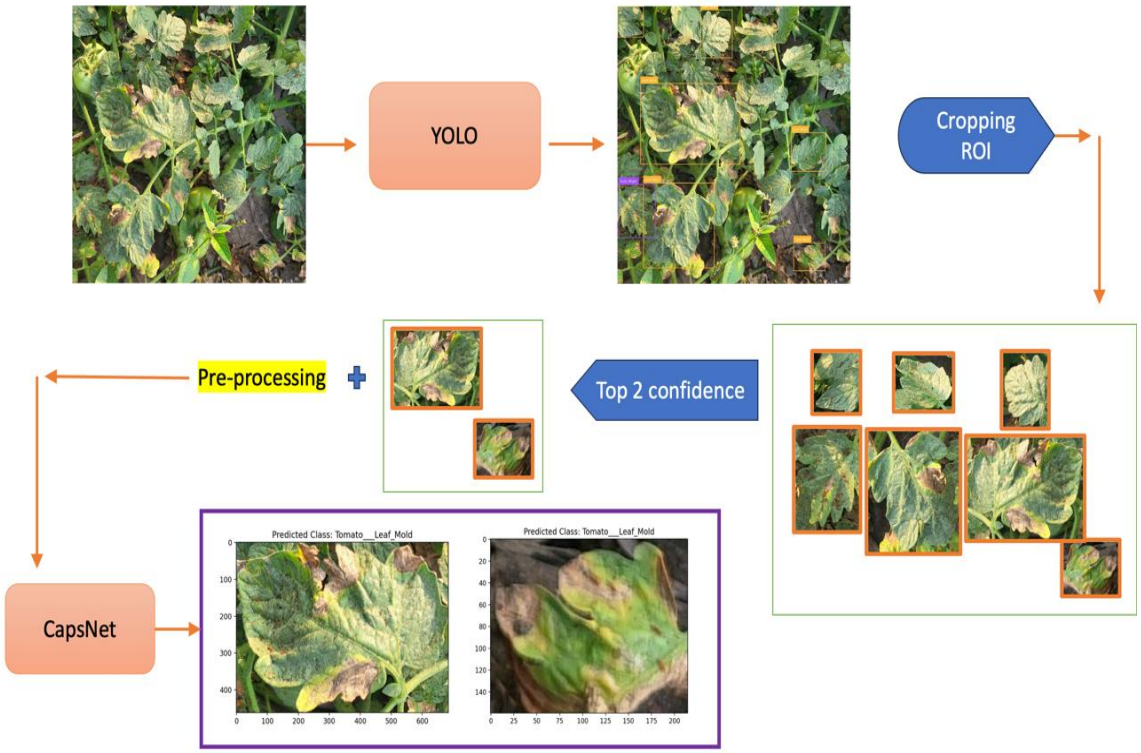


Figure 7.11: Overall Workflow of the Caps-YoLo Model

7.4 Summary

The Capsule Neural Network model demonstrates promising performance in both the classification of plant diseases and object detection tasks. By addressing challenges identified in the confusion matrix and optimizing training dynamics using accuracy and loss plots, the model exhibits potential for enhancing accuracy and robustness in real-world scenarios. This concise summary encapsulates the key findings and insights derived from the provided content, offering a comprehensive understanding of the model's effectiveness in both classification and object detection tasks.

The classification report further validates the efficacy of the Capsule Neural Network model in accurately identifying plant diseases. With consistently high precision, recall, and F1-score values across most classes, the model showcases its practical application potential in agricultural settings. Additionally, in object detection tasks, the model's performance underscores its utility and relevance, particularly in scenarios requiring timely and precise disease detection for effective crop management and yield maximization.

Chapter 8

DRONE IMPLEMENTATION

8.1 Introduction

The use of drones for disease detection and pesticide spraying has several advantages and disadvantages. **Targeted pesticide application** is now possible by using deep learning models to analyze images, **reducing pesticide use and costs** while minimizing environmental impact. Drones also improve efficiency by covering large areas quickly, **saving time and labor** over traditional methods. Furthermore, they protect farmers by eliminating the need for direct field entry, which **reduces exposure risks**. However, adopting UAV and professional drone technology presents challenges, necessitating extensive information gathering and analysis. Drone and GPS configuration is done using software mission planners, which require a lot of trial and error. Despite these challenges, the drone was successfully assembled and configured, allowing it to fly over fields and spray pesticides.

The drone, which is outfitted with **Crossflight flight control**, a **F450 frame**, **Flysky 1100kv motors, 40A Esc, and 10-inch propellers**, achieves the project's primary goal of carrying 1.5 to 2kg thrust, capturing images, and transmitting them to a local server for analysis.

8.2 Components and Features

- **F450 Frame:** The F450/Q450 Quadcopter Frame is made of 35% glass fibre, which ensures strength and durability. It has highly resistant Polyamide-Nylon arms with support ridges for stability and faster flight. The practical configuration includes a PCB for easy ESC wiring, which eliminates the need for additional PDB.
- **1100kv Motors:** The A2217 1100kv BLDC motors are designed for quadcopters and multirotors, offering exceptional performance and efficiency. With a 3mm banana male connector, they can be connected directly to a 40A ESC without soldering, providing up to 2150 gms of thrust with a 3-5S LiPo battery.
- **ReadytoSky 40A ESC:** Designed exclusively for quadcopters and multirotors, the ReadytoSky 40A 2-4S ESC delivers superior motor speed control and compatibility with 2-4s

LiPo batteries. Its special program and fast throttle response ensure excellent flight performance, while twisted throttle signal wires enhance stability.

- **Crossflight Flight Controller:** The Radiolink Crossflight Flight Controller offers accurate flight control for UAVs and drones. With advanced gyroscopic sensors and responsive flying algorithms, it provides precise handling for both beginners and experienced enthusiasts. Its sleek design and compatibility with Radiolink transmitters and receivers make it suitable for various drone applications.

- **FlySky FS-i6 Transmitter:** Operating in the 2.405 to 2.475GHz frequency band, the FlySky FS-i6 Transmitter ensures jam-free long-range radio transmission. Its high sensitivity receiver and low-power electronic components enhance interference immunity and safety. With multiple channel coding and error correction, it offers stable communication and reliable transmission distance.

- **TS100 GPS:** The Radiolink TS100 Mini GPS provides precise location and navigation data essential for accurate image acquisition during drone operations. Its compact and lightweight design reduces weight on the drone, while fast satellite acquisition and durability ensure efficient data collection in diverse conditions.

8.3 Modes and Uses

8.3.1 Loiter Mode in GPS

Loiter Mode: Holding your drone in place Loiter mode is a flight mode featured on many drones, especially those with autopilots or flight controllers. It enables the drone to keep a reasonably constant position in the air, including height, location, and direction. This makes it extremely handy for a variety of drone applications.

8.3.2 How Loiter Mode Works?

When you activate loiter mode, the drone's autopilot takes control and maintains its position using sensors such as GPS, compass, and gyroscope. Here is a breakdown of what loiter mode usually controls:

- *Altitude:* The drone retains the altitude it was at when loiter mode was activated.

- *Location:* The autopilot uses GPS data to maintain the drone within a certain horizontal radius of its starting location.

- *Heading:* The drone tries to face a specified direction, however wind may cause minor changes. It's crucial to understand that loiter mode isn't flawless. Wind and GPS accuracy can cause the drone to deviate somewhat from its original position. However, loiter mode lowers drift substantially more than manual control.

8.3.3 Benefits of Loiter Mode

Loiter mode has various benefits for drone pilots:

- *Composing Shots:* When taking images or movies, loiter mode allows you to focus on framing and composition rather than manually maintaining the drone's position. This is especially useful for recording high-resolution photos and exact video sequences.

- *Monitoring:* Loiter mode is useful for tasks such as tracking a certain location or object. The drone can maintain its position for an extended period of time, delivering a solid aerial picture.

- *Safer Operation:* Loiter mode can serve as a safety net for new pilots. If you lose control briefly, the drone will seek to keep its place rather than drift away.

8.4 Introduction to Raspberry Pi

The Raspberry Pi is a line of single-board computers designed in the United Kingdom to encourage the teaching of fundamental computer science in schools. It's a credit-card-sized computer with an impressive level of processing power, making it a popular choice for hobbyists, educators, and creators alike. Here are some important aspects of the Raspberry Pi:

- **Compact and Portable:** With its small size and lightweight design, it is suitable for applications with limited area.

- **Cost-effective:** Raspberry Pis are significantly less expensive than typical desktop computers, making them accessible to a wide spectrum of consumers.

- **Versatile:** Raspberry Pis are versatile and can be used for a range of activities, including learning to code, running web servers, and controlling robots. The Raspberry Pi Foundation encourages open-source software, which provides users with a wide range of operating systems and apps.
- **GPIO Pins:** Many Raspberry Pi versions have General-Purpose Input/Output (GPIO) pins, which allow you to add sensors, actuators, and other electronics to expand their usefulness.
- **Camera Connection:** Most Raspberry Pi models provide a dedicated Camera Serial Interface (CSI) connection for connecting Raspberry Pi cameras.

8.5 Introduction to the Raspberry Pi Camera Module 3 (Pi Cam 3)

The Raspberry Pi Camera Module 3 (Pi Cam 3) is the Raspberry Pi Foundation's most recent official camera module, which was introduced in January 2023. It's a considerable boost over its predecessor, the Pi Cam 2, with several improvements:

- **Higher Resolution:** The Pi Cam 3 uses a 12-megapixel Sony IMX708 sensor, which captures far more information than the Pi Cam 2's 8-megapixel sensor.
- **Improved Low-Light Sensitivity:** The Pi Cam 3 operates better in low-light situations, resulting in crisper photos in areas with less illumination.
- **Autofocus:** Unlike the Pi Cam 2, the Pi Cam 3 is available in standard and wide-angle configurations, with both boasting built-in phase-detection autofocus (PDAF) for crisper shots, particularly at different distances.
- **High Dynamic Range (HDR) Support:** The Pi Cam 3 supports HDR mode (up to 3-megapixel output), allowing it to capture more light and dark details in a scene.

8.5.1 Connecting Raspberry Pi with Pi Camera 3

The technique of connecting the Raspberry Pi and Pi Cam 3 is fairly simple:

1. **Gather Materials:** To connect your monitor, you'll need a Raspberry Pi (any model with a CSI port), a Pi Cam 3 (standard or wide-angle), and a micro HDMI connector (or any appropriate display cable). Prepare the Raspberry Pi by installing the newest Raspbian operating system or another suitable OS on its microSD card and booting it.

2. **Connect the Pi Cam 3:** Locate your Raspberry Pi's CSI connection (typically a small black plug near the USB ports). Carefully align the ribbon cable of the Pi Cam 3 with the CSI port and gently press it down to secure it.

8.5.2 Capturing Photos Using Pi Cam 3

Now that your Raspberry Pi and Pi Cam 3 are connected, you can start capturing photos using software:

- **Open Terminal:** Launch the terminal window on your Raspberry Pi.

- Use the command `sudo libcamera-jpeg-o test.jpeg` on Raspberry Pi captures and stores images in the operating system. These images are then transferred to a local computer and uploaded to a website.

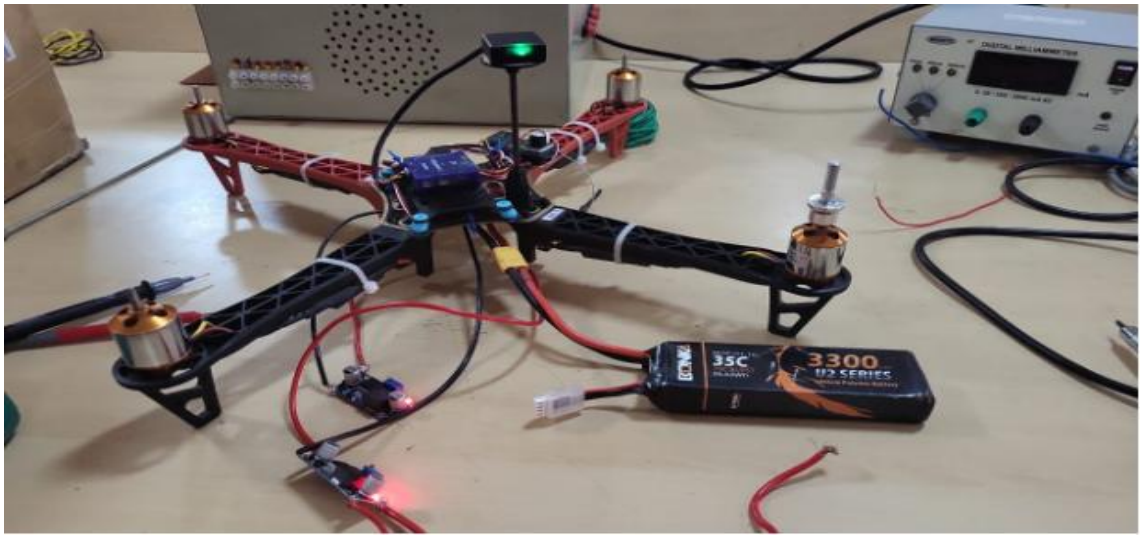


Figure 8.1: Image after assembling all parts of the Drone

8.6 Communication Setup

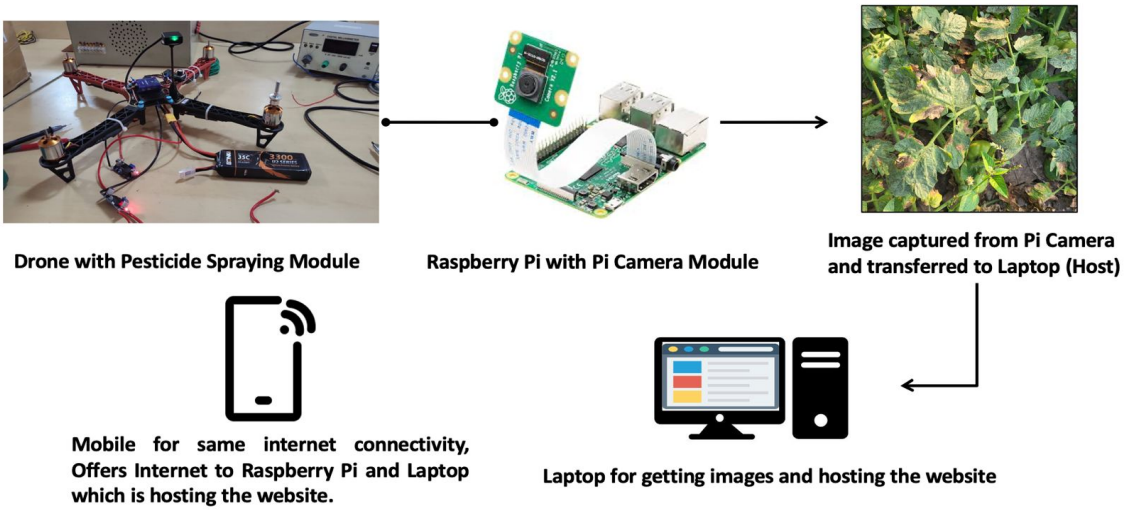


Figure 8.2: Communication Setup - Block Diagram

The communication setup for this project as shown in Fig. 8.2 involves several components working together seamlessly:

1. **Drone with Pesticide Spraying Module:** This component carries a pesticide tank and a pump for spraying pesticides on the crops. It can be deployed to specific areas identified as needing treatment based on the analysis of captured images.

2. **Raspberry Pi with Camera Module:** The Raspberry Pi computer, equipped with a camera module, captures images of the crops in the farm field. This serves as the primary means of gathering visual data from the field.

3. **Image Captured from Pi Camera:** Images captured by the Raspberry Pi camera provide visual data for analysis and monitoring of crop health and potential diseases.

4. **Mobile Phone for Internet Connectivity:** A mobile phone serves as a hotspot to provide internet connectivity to both the Raspberry Pi and the laptop, enabling data transfer and communication between the components.

5. **Laptop for Getting Images and Hosting the Website:**

   (a) **Getting Images:** The laptop receives the images captured by the Raspberry Pi camera, for further analysis or storage.

(b) **Hosting the Website:** The laptop hosts a website that allows users to predict the diseases available in the image captured through Pi Camera. This website may provide real-time updates on crop health, and other relevant information. This is explained in the next chapter.

Overall, this system enables automated monitoring of crop health using image data captured by a drone-mounted camera. Analysis of these images on a laptop, potentially using deep learning models, can detect crop diseases or other issues. The integration of a pesticide spraying system on the drone allows for targeted treatment of identified problem areas.

8.7 Capture and Transfer Functionality using Website

The webpage interface facilitates image acquisition through the Pi Camera using the Capture and Transfer buttons.

Upon clicking the **Capture Button**, the Pi Camera captures an image, which is then displayed on the website after a 5-second delay.

Subsequently, clicking the **Transfer Button** initiates the transmission of the captured image from the Raspberry Pi to the Local Laptop, which is hosting the website for further prediction and analysis. The webpage outline is shown in Figure 8.3

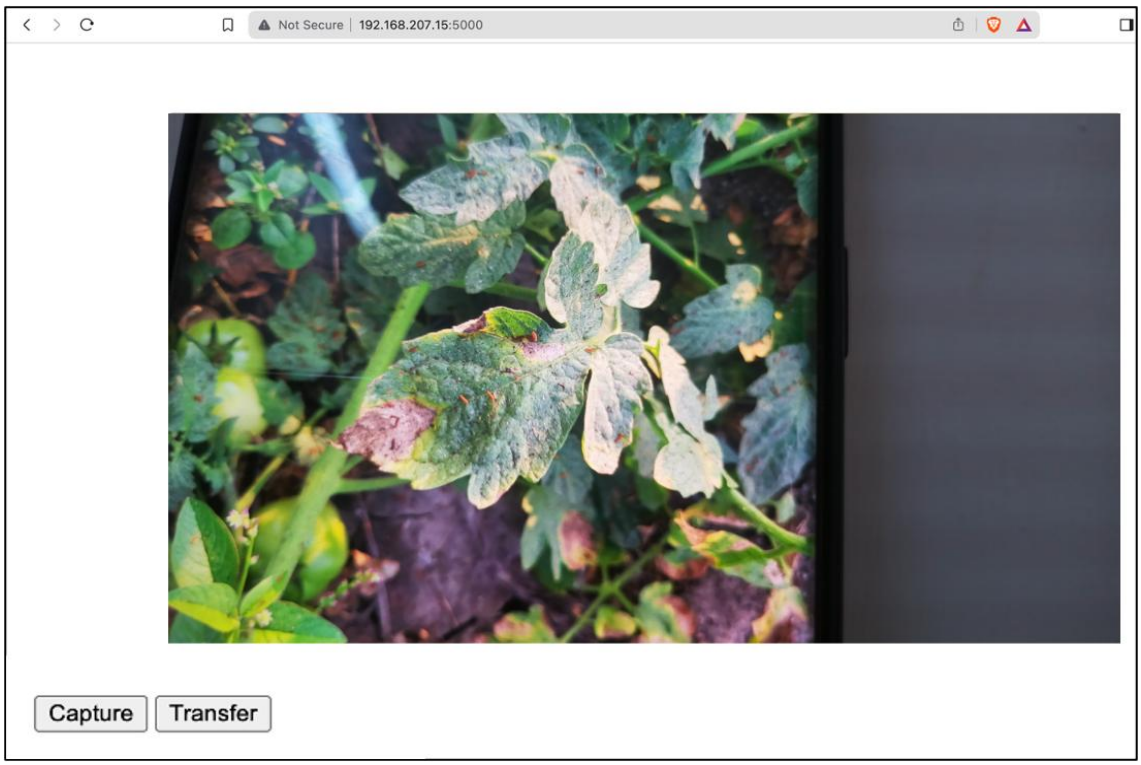


Figure 8.3: Webpage outline for Capture and Transfer Functionality

8.8 Summary

This chapter provides an overview of the Raspberry Pi, emphasizing its significance in computer science education and highlighting its key features such as compactness, affordability, GPIO pins, and camera connectivity. It also explores the advancements of the Pi Cam 3 over its predecessor, including higher resolution, improved low-light sensitivity, autofocus capabilities, and support for High Dynamic Range (HDR) imaging. Later, it discusses the functionality of loiter mode in drones, enabling them to maintain a steady position in the air. It explores the advantages of utilizing drones for targeted pesticide application, increased operational efficiency, and enhanced safety for farmers. It also addresses challenges encountered during implementation and the successful outcomes achieved. Finally, it describes the components and communication infrastructure of an automated crop monitoring system, involving drones, Raspberry Pi with a camera module, mobile phone for internet connectivity, and a laptop for image processing and website hosting.

Chapter 9

HOSTING PREDICTIVE MODEL USING WEBSITE

9.1 Advantages of Implementing ML Models on Websites

Integrating machine learning (ML) models into websites presents several advantages, **bolstering user experience, scalability, and accessibility**. Firstly, by embedding ML functionalities into websites, users gain the flexibility to access these models from any location with an internet connection, broadening the reach of ML applications. Websites offer a familiar interface, facilitating intuitive interaction with ML algorithms through well-designed user interfaces (UI). *This ensures users can input data, adjust parameters, and interpret results seamlessly, irrespective of their familiarity with underlying ML concepts.*

Real-time interaction is another significant advantage, enabling users to receive immediate feedback and results from ML models. This **instantaneous feedback loop enhances user engagement and facilitates swift decision-making based on model predictions or insights**. Scalability is inherent to websites, with cloud-based hosting solutions ensuring ML models can handle varying levels of user traffic and data input without compromising performance.

Integration capabilities are enhanced through website integration, enabling seamless incorporation of ML models with other technologies such as databases, APIs, and third-party applications. This integration enriches the *versatility and utility* of ML-powered features within the website ecosystem. Personalization is facilitated by ML models integrated into websites, enabling tailored user experiences through analysis of behavior, preferences, and historical data. *Such personalization encompasses customized recommendations, content filtering, and adaptive interfaces, fostering higher user satisfaction and engagement.*

The iterative nature of website development allows for continuous improvement and updates to ML models based on user feedback, performance metrics, and evolving data. This ensures that ML models remain relevant, accurate, and effective over time, adapting to changing user needs and preferences. Furthermore, websites offer robust analytics and tracking capabilities, empowering developers to monitor ML model performance, user interactions, and conversion metrics. *These insights inform optimization strategies, identify areas for improvement, and quantify the impact of ML-driven features on overall website performance.*

In summary, integrating ML models into websites yields numerous advantages including accessibility, user-friendliness, scalability, integration capabilities, personalization, continuous improvement, and analytics-driven insights. These advantages collectively contribute to a richer user experience, improved functionality, and an enhanced value proposition for website visitors.

9.2 Implementation Steps for a Machine Learning Model in a Website

To implement a machine learning (ML) model on a website, you'll need to follow several steps. Below is a high-level overview of the process:

1. **Select a Machine Learning Model:** Choose an appropriate ML model based on your specific use case, data, and objectives. Consider factors such as the complexity of the problem, the type of data available, and the desired prediction or classification task.

2. **Prepare and Clean Data:** Collect and preprocess the data that will be used to train and test the ML model. This may involve tasks such as data cleaning, feature engineering, and data normalization to ensure that the data is in a suitable format for training the model.

3. **Train the Model:** Use the prepared data to train the selected ML model. Depending on the complexity of the model and the size of the dataset, this step may require significant computational resources. Train the model using algorithms such as supervised learning, unsupervised learning, or reinforcement learning, depending on the nature of the problem.

4. **Evaluate Model Performance:** Assess the performance of the trained ML model using appropriate evaluation metrics and validation techniques. This step helps determine how well the model generalizes to unseen data and whether it meets the desired accuracy or performance thresholds.

5. **Integrate the Model with a Web Application:** Once you have a trained and evaluated ML model, integrate it into a web application. This typically involves developing backend services or APIs that expose the model's functionality to the web interface.

6. **Develop a Web Interface:** Design and develop a user-friendly web interface that allows users to interact with the ML model. This interface may include input forms for user input, visualization of model outputs, and interactive elements for exploring model predictions or insights.

7. **Implement Backend Functionality:** Implement the backend functionality required to process user input, invoke the ML model, and return the results to the user interface. This may involve writing server-side code in languages such as Python, Java, or Node.js, depending on your preferences and infrastructure.

8. **Deploy the Web Application:** Deploy the web application to a web server or a cloud platform such as AWS, Google Cloud, or Microsoft Azure. Ensure that the deployment environment is configured to support the ML model's runtime requirements and can handle anticipated user traffic.

9. **Monitor and Maintain:** Continuously monitor the performance of the deployed web application and ML model. Implement logging, error handling, and monitoring tools to detect and address any issues that may arise. Regularly update the model as new data becomes available or as improvements are made.

9.3 Technologies Used in Website Development

Here's a brief overview of the roles of each technology in website development:

- **HTML (Hypertext Markup Language):** HTML provides the structure and content of web pages. It defines the elements and layout of a webpage, including headings, paragraphs, images, links, and other multimedia elements.

- **CSS (Cascading Style Sheets):** CSS is used for styling and formatting web pages. It controls the presentation of HTML elements, including layout, colors, fonts, and spacing, to enhance the visual appearance and user experience of the website.

- **JavaScript:** JavaScript is a programming language used for adding interactivity and dynamic behavior to web pages. It enables developers to create interactive features such as animations, form validation, dynamic content loading, and user interface enhancements.

- **jQuery:** jQuery is a fast, lightweight JavaScript library that simplifies client-side scripting tasks. It provides a wide range of pre-built functions and plugins for DOM manipulation, event handling, AJAX requests, and animation, making it easier to develop interactive and responsive websites.

- **React.js:** React.js is a JavaScript library for building user interfaces, developed by Facebook. It allows developers to create reusable UI components and build single-page applications (SPAs) with a declarative and component-based approach. React.js is known for its performance, scalability, and efficient rendering through the use of a virtual DOM.

- **Python:** Python is a versatile and easy-to-learn programming language used for various purposes, including web development. It offers simplicity, readability, and a vast ecosystem of libraries and frameworks. Python's syntax and clean code make it well-suited for backend development, data processing, and scripting tasks in web development projects.

- **Django:** Django is a high-level Python web framework that enables rapid development of secure and scalable web applications. It follows the Model-View-Controller (MVC) architectural pattern and provides built-in features for authentication, URL routing, database abstraction, templating, and more. Django's "batteries-included" philosophy and robust ecosystem make it a popular choice for building complex web applications efficiently.

Each of these technologies plays a crucial role in different aspects of website development, from defining the structure and styling of web pages (HTML and CSS) to adding interactivity and dynamic functionality (JavaScript, jQuery, and React.js) and implementing backend logic and server-side processing (Python and Django). Combining these technologies allows developers to create modern, responsive, and feature-rich websites that meet the needs of users and businesses alike.

9.4 Features of the Website

- **User Authentication:** Ensures secure access to the database and main page, providing a safe environment for users.

- **Main Page:** Provides comprehensive information about the website's features and functionalities, guiding users through its usage.

- **Prediction Page:** Enables users to input data and receive accurate forecasts or outputs based on advanced predictive models, enhancing decision-making capabilities.

- **Privacy and Security:** Guarantees privacy and security through robust user authentication measures, safeguarding user data and interactions.

- **Seamless Experience:** Offers a seamless browsing experience, allowing users to effortlessly explore team members' profiles and utilize predictive capabilities.
- **Easy Upload:** Facilitates quick and hassle-free uploading of leaf images for analysis, streamlining the data input process.
- **Fast Analysis:** Delivers rapid analysis results, providing users with instant predictions of plant diseases for prompt decision-making.
- **Accurate Results:** Utilizes state-of-the-art deep learning technology to ensure high-precision predictions, enhancing the reliability of the generated outputs.
- **Accessible Anywhere:** Enables access from any device with an internet connection, facilitating on-the-go disease monitoring and management.
- **User-Friendly Interface:** Features an intuitive interface design, allowing users of all technical backgrounds to navigate the platform with ease.

9.5 Website Visualization - Front End

The development journey of the website has been meticulously orchestrated, focusing on delivering an immersive and user-centric experience. Leveraging HTML, CSS, and JavaScript, the front-end development ensures a seamless interface across all pages.

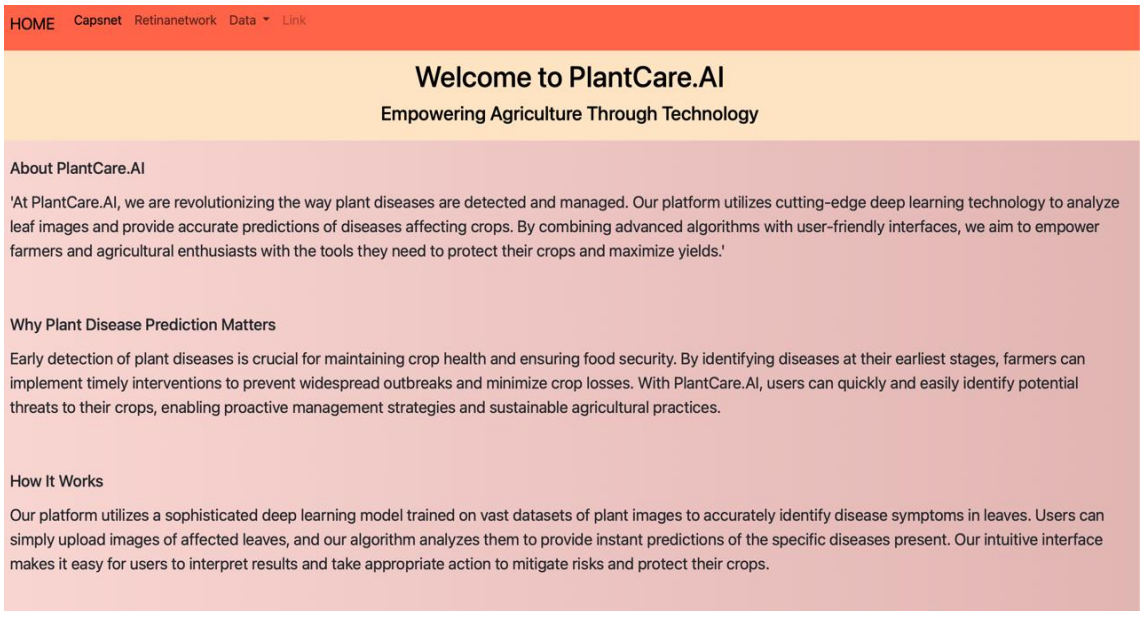


Figure 9.1: Upon hosting the website, the main page provides information about the features of the website.

At the start of the website lies a web page emphasising on the features of the project and about the website.

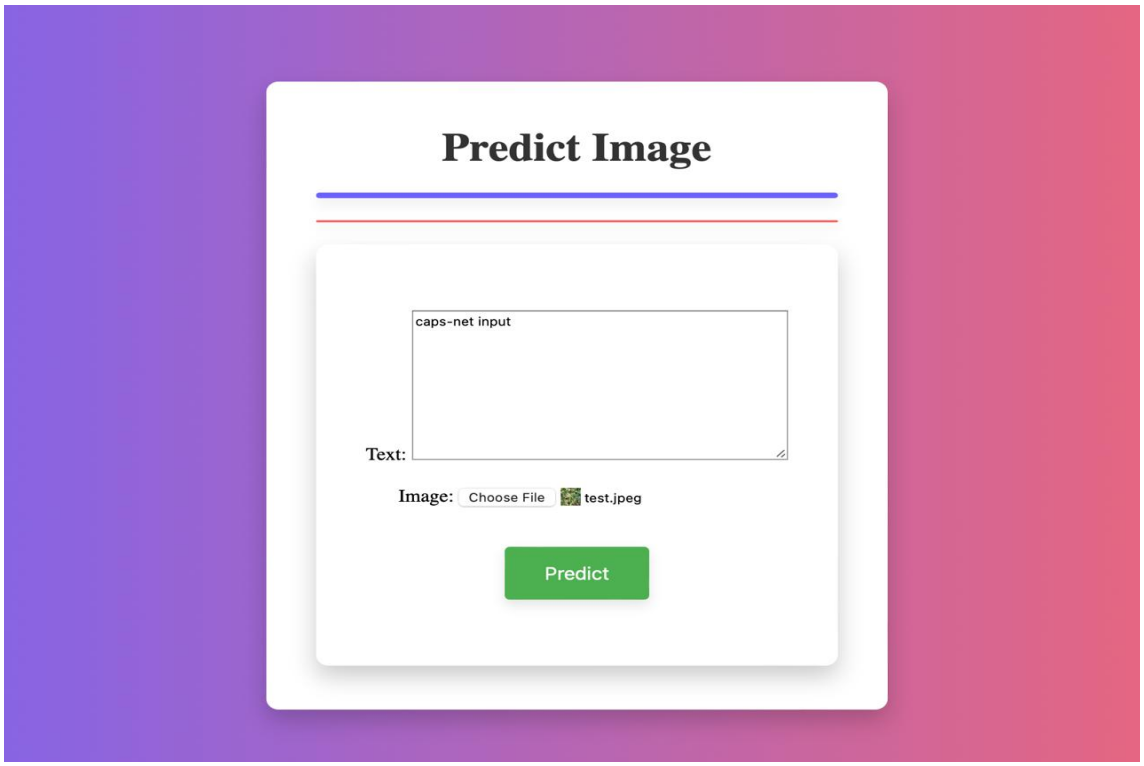


Figure 9.2: CapsNet Prediction Web page displayed for image to be uploaded

Upon accessing the CapsNet Prediction option in the homepage, users can upload the individual diseased leaf images separately as shown in Figure 9.2. These leaf images are then tested using the Capsule Network model and displays the Disease Name along with Remedial Measures to be followed in a new webpage as shown in Figure 9.3.

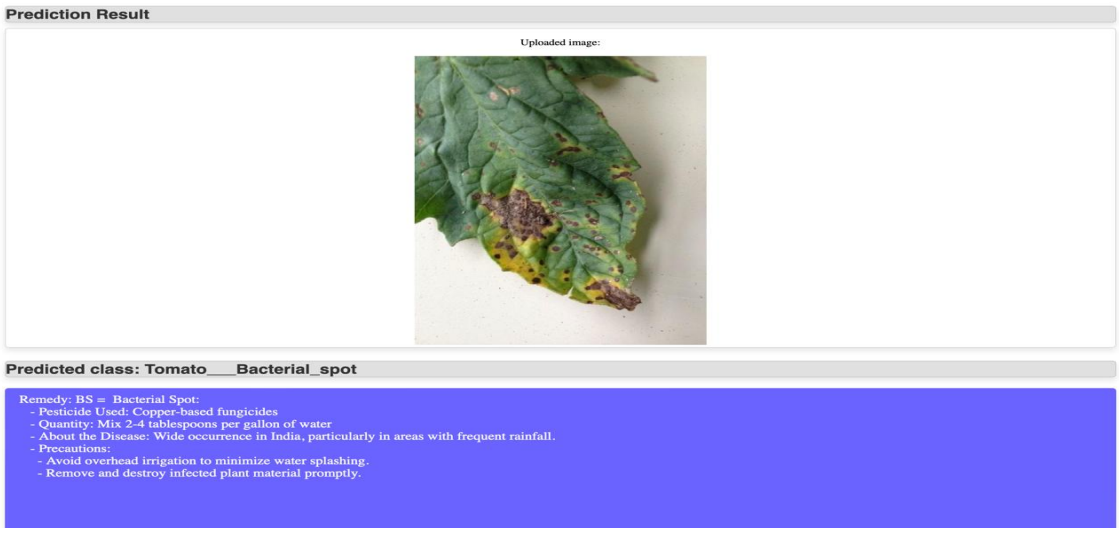


Figure 9.3: Disease class along with Remedial Measure displayed in the Webpage for CapsNet Model

Now in order to predict the diseases which are present on the image captured through Pi Camera using a Respberry Pi mounted on a Drone, we use the Object Detection Model Webpage which extracts the Region of Interests (ROI's). Based on the top 2 confidence scores, those ROI's are sent to the CapsNet Model for prediction, and its predictions are displayed in a similar fashion as showcased in Figure 9.3. This process is showcased in Figure 9.4

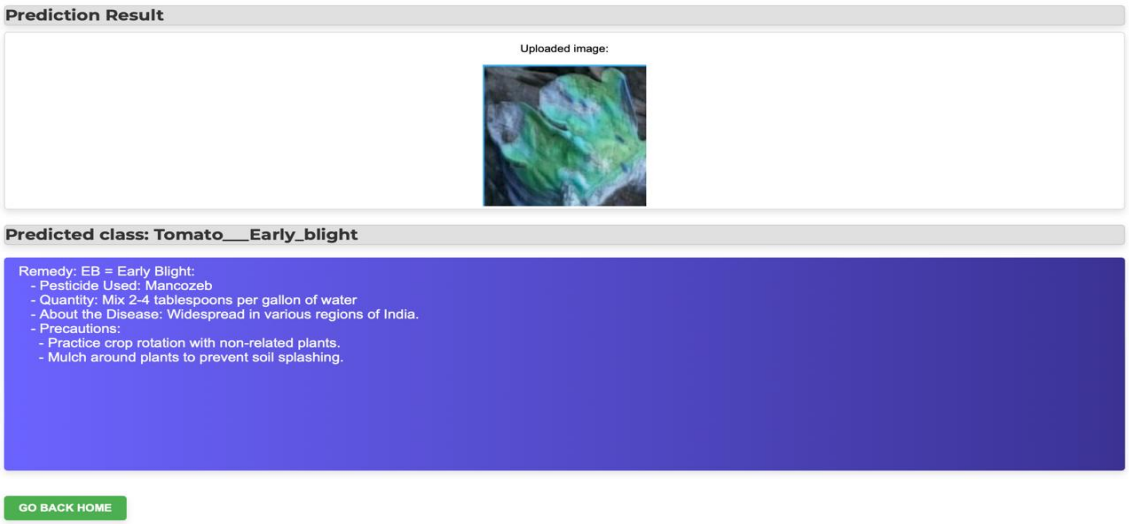


Figure 9.4: Disease class along with Remedial Measure displayed in the Webpage for Combined Model

Figure 9.4 is an extracted ROI from the image illustrated in Figure 9.5. This image is converted to smaller ROI's and sent to the Capsule Network model which happens in the Back End of the website.

Figure 9.5: Farm Image captured through Raspberry Pi Camera with various ROI's

9.6 Website Visualization - Back End

After establishing the website's foundational structure, focus shifted to enhancing functionality through a robust backend system. Python, with Django, became the cornerstone. Django facilitated communication between HTML webpages and Python code.

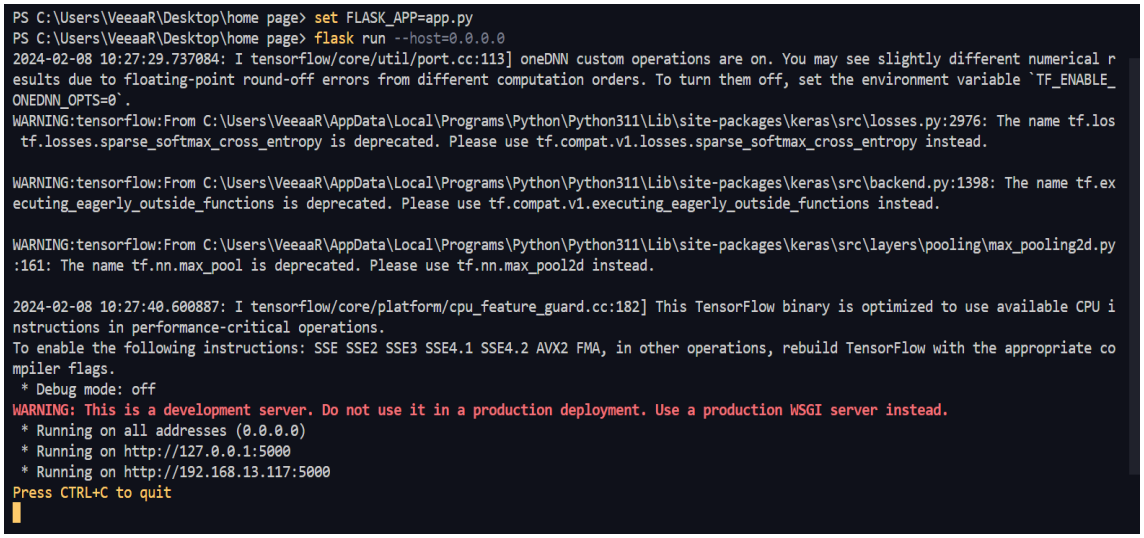


Figure 9.6: Terminal snapshot providing insights into devices accessing the hosted website.

The backend architecture streamlined data flow and operations. Images submitted through the webpage were received by JavaScript and relayed to the server. Here, among code files and model repositories, images underwent meticulous analysis through the .h5 model file.

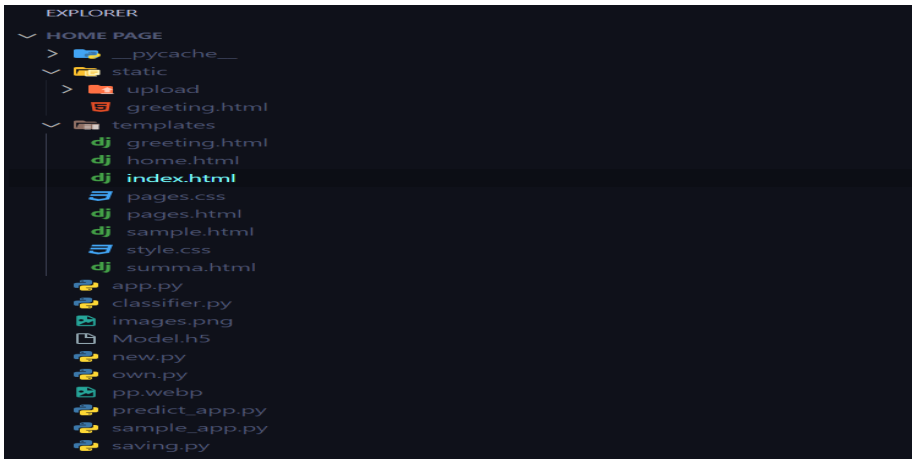


Figure 9.7: Backend Folder Directory in the Host Machine

Using Python, the model executed predictive algorithms, generating diagnostic insights. These outputs were transmitted back to the webpage through Django's communication channels, offering users real-time diagnostic results.

The website's visitors encountered a seamless interface facilitating image submission and real-time diagnostic results. Through Django, Python, and JavaScript integration, the website transcended traditional boundaries, offering an unparalleled user experience.

To enrich functionality, a new dimension was introduced: remedies for various diseases. Separate code files were created, each dedicated to a specific disease, housing detailed information about corresponding remedies.

The figures in this text illustrate various aspects of the website's front-end and back-end, providing visual aids for better understanding.

9.7 Summary

In this chapter, we explored the integration of machine learning (ML) models into websites, highlighting the advantages, implementation steps, and technologies involved. Leveraging HTML, CSS, and JavaScript, the front-end development ensures a seamless interface, with robust user authentication mechanisms ensuring security. Through intuitive sections like "Team" and "Project," users can explore comprehensive profiles and engage in transformative experiences, such as disease prediction facilitated by ML models. On the backend, Python with Django streamlines data flow and model execution, delivering real-time diagnostic results. Visual aids, including the Cloud Console Interface Module and Storage Bucket details, provide insights into Google Cloud's capabilities. This chapter underscores the transformative potential of ML-integrated websites, offering a user-centric approach and empowering organizations to leverage data effectively for various business needs.

Chapter 10

CONCLUSION

10.1 Comparison with Existing Results

This project involved developing and comparing different architectures for Capsule Network Models, focusing on their training accuracy, testing accuracy, and the number of epochs required. The main goal was to minimize the number of epochs needed to attain peak accuracy. The findings include:

Table 10.1: Comparative Study - Capsule Network Model

S.No	No. of Epochs	Training Accuracy	Testing Accuracy
1	10	70.14%	68.32%
2	30	90.96%	85.96%
3	50	95.54%	96.06%
4	100	97.11%	97.41%

10.2 Discussions

The table 10.1 shows the effect of different epochs on the Capsule Network Model's performance. Increasing epochs from 10 to 100 progressively improves both training and testing accuracy. However, the difference in training and testing accuracy suggests that prolonged training with increasing epochs may result in overfitting. After 50 epochs, increase in test accuracy becomes insignificant, indicating a possibility of overfitting with extending the number of epochs. The optimal performance balance between computational efficiency and accuracy occurs around 50-100 epochs. This emphasises the importance of selecting the right number of epochs to achieve satisfactory model generalisation. Overall, the table emphasises the importance of carefully controlling number of epochs, accuracy and computation time in order to achieve superior model performance while avoiding overfitting.

10.3 Conclusion

This project demonstrates the effectiveness of reducing epochs in deep learning training, resulting in higher accuracy with lower computational demands. This optimisation provides a practical solution for accurate disease detection in precision agriculture. The integrated system, which includes manual drone control and deep learning models such as YoLo and CapsNet, achieves high accuracy rates while easing farm management.

This project presents a disease prediction method for tomato leaf disease detection and classification that combines YoLo v8 with Capsule Neural Network, or CapsNet. Accurate disease identification was obtained by training a deep learning model on a dataset with broad collection of pictures of tomato leaves that included nine common diseases as well as healthy specimens. The suggested method addresses the convenience and effectiveness of disease detection in agricultural settings by providing a GUI based, user-friendly web platform for uploading the images of Tomato leaves from a farm, that were captured using a drone and obtaining real-time disease forecasts. Its speedy processing of a high number of photos makes it a useful tool for a wide range of agricultural applications.

The system optimises resource utilisation by incorporating disease detection-based pesticide spraying, thereby improving crop health and sustainability. The significance of drone-based disease detection in deep learning comes from its potential to revolutionise agricultural practices. This technology improves crop management efficiency while also addressing global issues like food security and sustainable agriculture.

REFERENCES

[1] D. Andujar, ”Back to the Future: What Is Trending on Precision Agriculture?” *Agronomy*, vol. 13, no. 8, p. 2069, 2023, doi: 10.3390/agronomy13082069.

[2] A. Pandey, S. K. Sain, and P. Singh, ”A Perspective on Integrated Disease Management in Agriculture,” *Bio Bulletin*, vol. 2, pp. 13-29, 2016.

[3] A. D. Boursianis *et al.*, ”Internet of Things (IoT) and Agricultural Unmanned Aerial Vehicles (UAVs) in smart farming: A comprehensive review,” *Internet of Things*, vol. 18, pp. 100187, 2022, ISSN 2542-6605, doi: 10.1016/j.iot.2020.100187.

[4] A. Hafeez, M.A. Husain, S.P. Singh, A. Chauhan, et al., ”Implementation of drone technology for farm monitoring & pesticide spraying: A review,” *Computers and Electronics in Agriculture*, Elsevier, 2022.

[5] C.J. Chen, Y.Y. Huang, Y.S. Li, Y.C. Chen, C.Y. Chang, et al., ”Identification of fruit tree pests with deep learning on embedded drone to achieve accurate pesticide spraying,” *IEEE*, 2021.

[6] S. Souvanhnakhoomman, ”Review on application of drone in spraying pesticides and fertilizers,” arXiv preprint arXiv:2402.00020, 2024.

[7] L.M. Abouelmagd, M.Y. Shams, H.S. Marie, et al., ”An optimized capsule neural networks for tomato leaf disease classification,” *EURASIP Journal on ...*, Springer, 2024.

[8] J.D. Stamford, S. Vialet-Chabrand, I. Cameron, T. Lawson, ”Development of an accurate low-cost NDVI imaging system for assessing plant health,” *Plant Methods*, Springer, 2023.

[9] A. Soetedjo, E. Hendriarianti, ”Plant leaf detection and counting in a greenhouse during day and nighttime using a Raspberry Pi NoIR camera,” *Sensors*, 2021.

[10] N. Genze, M. Wirth, C. Schreiner, R. Ajekwe, M. Grieb, et al., ”Improved weed segmentation in UAV imagery of sorghum fields with a combined deblurring segmentation model,” *Plant Methods*, Springer, 2023.

[11] AbdallahAliDev, ”PlantVillage Dataset,” Kaggle, https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset, Accessed: October 10, 2023.

[12] Aline Dobrovsky, ”Plant Disease Classification Merged Dataset,” Kaggle, https://www.kaggle.com/datasets/alinedobrovsky/plant-disease-classification-merged-dataset, Accessed: October 10, 2023.

[13] Roboflow, ”PlantDoc Dataset,” Roboflow, https://public.roboflow.com/object-detection/plantdoc?ref=blog.roboflow.com, Accessed: October 10, 2023.

[14] H. Kibriya, R. Rafique, W. Ahmad and S. M. Adnan, ”Tomato Leaf Disease Detection Using Convolution Neural Network,” 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), Islamabad, Pakistan, 2021, pp. 346-351, doi: 10.1109/IBCAST51254.2021.9393311.

[15] B. S. Reddy, A. Mallikarjuna Reddy, M. H. D. S. Sradda, T. Mounika, S. Mounika and M. K, ”A Comparative Study on Object Detection Using Retinanet,” 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, 2022, pp. 1-6, doi: 10.1109/MysuruCon55714.2022.9972742.

[16] S. J. Pawan and Jeny Rajan, ”Capsule networks for image classification: A review,” *Neurocomputing*, vol. 509, pp. 102-120, 2022, ISSN 0925-2312, `https://doi.org/10.1016/j.neucom.2022.08.073`.

[17] W. Qi, ”Object detection in high resolution optical image based on deep learning technique,” *Natural Hazards Research*, vol. 2, no. 4, pp. 384-392, 2022, ISSN 2666-5921, `https://doi.org/10.1016/j.nhres.2022.10.002`.

[18] Y. Huang, Y. Qian, H. Wei, Y. Lu, B. Ling, Y. Qin, ”A survey of deep learning-based object detection methods in crop counting,” School of Software, Xinjiang University, Urumqi, 830000, China.

[19] G. Altan, ”Performance Evaluation of Capsule Networks for Classification of Plant Leaf Diseases,” Computer Eng., Iskenderun Technical University, 31200, Hatay, Turkey.

[20] L. W. Waweru, B. T. Kipyego, D. M. Muchangi, ”Classification of Plant Leaf Diseases Based on Capsule Network-Support Vector Machine Model,” Department of MCIT, University of Embu, Embu, Kenya.

[21] A. D. Andrushia, T. M. Neebha, A. T. Patricia, K. M. Sagayam, S. Pramanik, ”Capsule network-based disease classification for Vitis Vinifera leaves,” *Neural Computing and Applications*, vol. 36, pp. 757–772, 2024. doi: 10.1007/s00521-023-09058-y