TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KALANKI, KATHMANDU

A Minor Project Final Defense Report

On

"COMPARING CNN, VISION TRANSFORMERS AND EFFICIENTNET FOR TOMATO DISEASE CLASSIFICATION"

[CT654]

SUBMITTED BY:

PARTIK NEUPANE	(ACE077BCT053)
RABIN OSTI	(ACE077BCT063)
RAVIN MARASINI	(ACE077BCT065)

SUPERVISOR:

Er. LAXMI PRASADH BHATT

Er. YUKESH THAPA

A Minor Project Proposal report submitted to the Department of Electronics and Computer Engineering in the partial fulfillment of the requirements for degree of Bachelor of Engineering in Computer Engineering

Kathmandu, Nepal

March 4, 2024

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING

APPROVAL LETTER

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a project report entitled "COMPARING CNN, VISION TRANSFORMERS AND EFFICIENTNET FOR TOMATO DISEASE CLASSIFICATION"

Submitted by:

PARTIK NEUPANE	(ACE077BCT053)
RABIN OSTI	(ACE077BCT063)
RAVIN MARASINI	(ACE077BCT065)

In partial fulfillment for the degree of Bachelor in Computer Engineering.

……………………… ………………………

Project Supervisor Project Supervisor

....

External Examiner

....

Er. Laxmi Prasad Bhatt

Academic Project coordinator

Department of Computer and Electronics Engineering

Date: March 4, 2024

ACKNOWLEDGEMENT

We take this opportunity to express our deepest and sincere gratitude to our Academic Project Coordinator **Er. Laxmi Prasad Bhatt**, Department of Electronics and Computer Engineering for his insightful advice, motivating suggestions, invaluable guidance, help and support in this project selection and also for his constant encouragement and advice throughout our journey till the date.

We express our deep gratitude to **Er. Prem Chandra Roy**, Head of Department of Electronics and Computer Engineering, **Er. Bikash Acharya** and **Er. Dhiraj Pyakurel**, Deputy Head, Department of Electronics and Computer Engineering for their regular support, co-operation, and coordination.

The in-time facilities provided by the department are also equally acknowledgeable.

We would like to convey our thanks to the teaching and non-teaching staff of the Department of Electronics & Communication and Computer Engineering, ACEM for their invaluable help and support hitherto. We are also grateful to all our classmates for their help, encouragement and invaluable suggestions.

PARTIK NEUPANE (ACE077BCT052)
RABIN OSTI (ACE077BCT063)
RAVIN MARASINI (ACE077BCT065)

ABSTRACT

Image classification has become relevant due to its ability to automate the process of recognizing and categorizing objects or patterns within images, facilitating tasks such as disease identification in agriculture, facial recognition in security systems, and content filtering in social media, among numerous other applications. In this report, we delve into the exploration of deep learning models, specifically Convolutional Neural Networks (CNNs), Vision Transformers (ViT), and Efficient Net for the classification of tomato diseases based on image data. The motivation behind employing these models is rooted in their potential to offer a nuanced understanding of intricate patterns within images, thereby augmenting the accuracy and efficiency of disease identification. The dataset utilized for tomato disease classification is sourced from Kaggle, a reputable platform known for hosting diverse and well-curated datasets. This report aims to shed light on the strengths and weaknesses of each model, emphasizing factors such as accuracy and training efficiency. By dissecting tomato images and classifying diseases, we seek to contribute valuable insights to the realm of image classification, particularly in the context of agriculture and plant pathology.

Index Terms – Convolutional Neural Networks (CNN), Deep Learning, EfficientNet, Image Classification, Vision Transformers (ViT)

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATION

AI	:	Artificial Intelligence
ANN	:	Artificial Neural Network
CNN	:	Convolutional Neural Network
Efficient Net	:	Efficient Neural Network
ML	:	Machine Learning
NLP	:	Natural Language Processing
ReLU	:	Rectified Linear Unit
ViT	:	Vision Transformer

CHAPTER-1: INTRODUCTION

1.1 Background

The agricultural sector, critical for global food security, confronts persistent challenges in preserving crop health, with tomato crops particularly susceptible to various diseases that can jeopardize yields. Timely and accurate disease classification is pivotal for effective agricultural management. In response to the limitations of traditional diagnostic methods, there has been a growing emphasis on integrating advanced technologies to enhance disease identification processes.

In recent years, the integration of deep learning models, including Convolutional Neural Networks (CNNs), Vision Transformers (ViT), and EfficientNet, has emerged as a promising avenue for revolutionizing disease detection. Lots of studies show that these models can be really good at figuring out diseases in images. They've made progress in being more accurate and efficient. As there are many models for disease classification, it is crucial to develop a smart and automated system capable of swiftly and accurately identifying tomato diseases through visual analysis. The diversity of available deep learning models necessitates a comprehensive comparison to discern their relative strengths and weaknesses. The comparison aims to identify the most effective model for tomato disease classification based on factors such as accuracy and training efficiency.

1.2 Motivation

The motivation is grounded in the challenges faced by farmers in timely and accurately identifying diseases that can impact tomato crops, potentially leading to substantial yield losses. Traditional identification methods are often slow and labor-intensive, necessitating the exploration of more efficient technological solutions.

1.3 Statement of the Problem

The central issue addressed in this report is the inefficiency and time-consuming nature of traditional methods employed in the identification of diseases affecting tomato crops. Current practices rely heavily on manual labor and visual inspection, resulting in delayed detection and response to emerging threats. The inadequacy of existing approaches calls for a paradigm shift towards the integration of deep learning models such as Convolutional Neural Networks (CNNs), Vision Transformers (ViT), and EfficientNet—to streamline the process of disease identification in tomato crops. We aim to bridge the existing gap between conventional methods and cutting-edge technology, offering a solution that not only expedites disease detection but also enhances the overall resilience and productivity of tomato cultivation.

1.4 Objective

- Identify the best model comparing CNNs, ViT and EfficientNet and build tomato disease classification web app integrating the model.

1.5 Significance of the study

This research holds vital importance for farmers and the agricultural community by introducing advanced deep learning models, including Convolutional Neural Networks (CNNs), Vision Transformers (ViT), and EfficientNet, for efficient tomato disease classification. The study aims to enhance disease management, prevent yield losses, optimize resource use, and modernize agricultural practices. By providing timely and accurate information, the research contributes to increased crop yield, sustainable farming, and the integration of cutting-edge technology in agriculture. Ultimately, the study's findings will benefit farmers positioning them at the forefront of technological advancements in the agricultural sector.

CHAPTER-2: LITERATURE REVIEW

The Food and Agriculture Organization of the United Nations (FAO) predicts that there will be 9.1 billion people on Earth by 2050. Food consumption will rise as the population grows. Meanwhile, it is difficult for nutrient levels to increase due to a lack of agriculture and the availability of clean water. At worst, agricultural crop diseases may destroy harvests and result in starvation in a nation, especially in underdeveloped countries with low wages. Plant evaluations are often carried out with the assistance of field consultants, but this job is difficult and time-consuming and requires the involvement of local professionals. It is difficult for people to analyze all crops individually, and such procedures of agronomic appraisal are seen as being unreliable. Therefore, it is crucial to detect plant diseases precisely and quickly. In order to address issues associated with manual processes, automated systems ae developed by the researchers [1].

The general process of using traditional image recognition processing technology to identify plant diseases used the K-means clustering method to segment the lesions regions, and combined the global color histogram (GCH), color coherence vector (CCV) ,local binary pattern (LBP), and completed local binary pattern (CLBP) was used to extract the color and texture features of apple spots, and three kinds of apple diseases were detected and identified based on improved support vector machine (SVM), and the classification accuracy reached 93%.

Chai et al. studied four tomato leaf diseases, including early blight and late blight leaf mildew and leaf spot, and extracted 18 characteristic parameters such as color, texture, and shape information of tomato leaf spot images, using stepwise discriminant and Bayesian discriminant principal component analysis (PCA), respectively. Principal component analysis and fisher discriminant methods were used to extract the characteristic parameters and construct the discriminant model. The accuracy of the two methods reached 94.71% and 98.32%, respectively.

Li and He selected 5 kinds of apple leaf diseases (speckled deciduous disease, yellow leaf disease, round spot disease, mosaic disease, and rust disease) as the research objects. By extracting 8 features of the apple leaf spot image, such as color, texture, and shape. The BP neural network model was used to classify and recognize the diseases, and the average recognition accuracy reached 92.6%.

In short, it can be concluded that studies on plant disease recognition based on traditional image processing technology have achieved certain results, with high accuracy of disease recognition, but there are still deficiencies and limitations as follow:

1. The research links and processes are cumbersome, highly subjective, time-consuming and labor-consuming.
2. It is heavily dependent on spot segmentation.
3. It is heavily dependent on artificial feature extraction.
4. It is difficult to test the disease recognition performance of the model or algorithm in more complex environments.

Therefore, it is of great significance to realize intelligent, rapid, and accurate plant leaf disease recognition.

In recent years, deep learning technology in the study of plant disease recognition made more progress. Deep learning technology in the face of the user is transparent, the researchers of plant protection and statistics professional level is not high, can be automatically extracted image features and classification of plant disease spot, eliminating the traditional image recognition technology of feature extraction and classifier design a lot of work, can express original image characteristics, has the characteristics of the end-to-end. These characteristics make deep learning technology in plant disease recognition-obtained-widespread attention, and it has become a hot research topic. This is due to three factors: the availability of larger datasets, the adaptability of multicore graphics processing units (GPUs), and the development of training deep neural networks and supporting software libraries, such as the computing unified device architecture (CUDA) from NVIDIA [2].

Various researchers have used cutting-edge technology such as machine learning and neural network architectures like Inception V3 net, VGG 16 net, and SqueezeNet to construct automated disease detection systems. These use highly accurate methods for identifying plant disease in tomato leaves. A pre-trained network model for detecting and classifying tomato disease has been proposed with 94–95% accuracy. The Tree Classification Model and Segmentation is used to detect and classify six different types of tomato leaf disease with a dataset of 300 images. Authors used leaf diseases to identify and achieve 87.2% classification accuracy through color space analysis, color time, histogram, and color coherence.AlexNet and VGG 19 models have been used to diagnose diseases affecting tomato crops using a frame size of 13,262. The model is used to achieve 97.49% precision. A transfer learning and CNN Model is used to accurately detect diseases infecting dairy crops, reaching 95%. A neural network to determine and classify tomato plant leaf conditions using transfer learning as an AlexNet based deep learning mechanism achieved an accuracy of 95.75% [3].

In 2017, Vaswani et al. introduced Transformers as a revolutionary model for machine translation, rapidly evolving to become the state-of-the-art method in numerous natural language processing (NLP) tasks. This transformative success has sparked considerable interest in leveraging similar methodologies for computer vision tasks, particularly in the context of image recognition. Despite the paradigm shift brought about by Transformers in NLP, convolutional architectures have maintained their dominance in computer vision. The robustness and efficacy of convolutional neural networks (CNNs) in capturing spatial hierarchies and patterns within images have made them the go-to choose for image-related tasks. Inspired by the triumph of Transformers in NLP, recent initiatives have aimed to bridge the gap between convolutional and self-attention architectures in computer vision. Some pioneering approaches have even ventured into entirely replacing convolutions with self-attention mechanisms, driven by the aspiration to harness the expressive power of self-attention in visual recognition tasks. The exploration of such hybrid architectures lays the groundwork for addressing the limitations of traditional convolutional models. In particular, the Vision Transformer (ViT) emerged as a remarkable milestone in this pursuit.

Drawing upon the success of Transformers in NLP and inspired by scaling strategies, ViT directly applies a standard Transformer to images, treating image patches as analogous to tokens in NLP [4].

The Vision Transformer model has gained a lot of recognition because of its ability to utilize an NLP-based transformer model to image classification problems by doing very few modifications in the transformer model. The ViT model has excellent performance on large-sized datasets as compared to mid-sized datasets. A CNN uses kernels to aggregate very local information in each layer. CNNs start to look very local and their receptive fields become global in each layer. Therefore, CNNs have a high locality inductive bias because of the shared weights between the neighboring pixels which helps them to localize objects. However, vision transformers tend to have a low locality inductive bias that leads to poor performance in small-sized datasets. Many ways have been introduced to deal with this low local inductive bias such as the introduction of the convolution layers in the ViT model. VIT modifies the transformer model used for the natural language processing task, to introduce the concept of self-attention in image classification giving outstanding results on large sized datasets but gives a lower performance on medium sized datasets because of its low locality inductive bias. After that many models have been proposed to deal with this problem and the outcomes confirmed the efficiency of Transformers by showing that the suggested Transformer-based model significantly outperforms the baseline approaches. The shifted patch tokenization and localities self-attention addresses the issue of the absence of locality inductive bias and allows the system to learn entirely from scratch even on small-size datasets. In order to enable volumetric medical image registration, ViT-V-Net was presented, a bridge between ViT and ConvNet. The experimental findings show that the suggested design outperforms many top registration techniques [5].

In 2020, the study by Mingxing et al. used neural architecture search to design a new baseline network and scale it up to obtain a family of models, called EfficientNets, which achieve much better accuracy and efficiency than previous Convolutional Neural Networks. The study proposed a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient. By balancing network depth, width, and resolution, EfficientNet showcased enhanced computational efficiency, making it a compelling candidate for tomato disease classification [6].

One of the advantages of using EfficientNet in disease detection on tomato leaves is its ability to capture features better than other algorithms used for disease detection. This is because EfficientNet has an architecture that is more efficient in capturing important features and can save on the number of parameters used. In addition, EfficientNet also has the ability to transfer learning, which is to utilize knowledge obtained from other datasets to be used in new datasets. This can improve accuracy in disease detection on tomato leaves. The EfficientNet model architecture is based on the concept of scaling up the model's architecture while also scaling up the input image size. This is achieved by increasing the resolution of the input image, and also increasing the depth and width of the network layers. Additionally, the model also uses depth-wise separable convolutions which are more efficient than standard convolutions. The compound scaling method is used to balance the trade-off between accuracy and efficiency. This method adjusts the model's architecture, input image resolution, and model's parameters to optimize the model's performance. The result is a family of models, each with a different level of accuracy and efficiency [7].

CHAPTER-3: REQUIREMENT ANALYSIS

In order to develop Tomato Disease Classification System, we aim to leverage the capabilities of CNN, EfficientNet, and Vision Transformer models. The central focus of this project is the seamless integration of the most effective model into the FastAPI backend, ensuring the creation of a high-performance web application capable of accurate and efficient tomato disease classification. The project requirements involve a careful consideration of both hardware and software elements.

3.1 Technical Requirements

3.1.1 Hardware Requirements:

1. CPU/GPU:
   - Multi-core processor (e.g., Intel i5 or i7, AMD Ryzen).
   - NVIDIA GPU with CUDA support (optional but recommended).
2. RAM:
   - 16GB or higher.

3.1.2 Software Requirements:

1. Operating System:
   - Window 10 or above.
   - Python 3.7 or later.
2. Development Environment:
   - Anaconda for managing Python environment.
3. Frameworks and Libraries:
   - TensorFlow for deep learning.
   - FastAPI for API development.
   - ReactJS for frontend development.

4. Additional Libraries:
   - Data processing: NumPy, Pandas
   - Image processing: OpenCV

3.2 Functional Requirements:

1. Image Upload: Enabling users to upload tomato images for disease classification.
2. Model Integration: Implementing functionality to dynamically integrate and switch between different machine learning models based on performance evaluation.
3. Classification Output: Displaying the classification results to users, indicating the identified tomato diseases.

3.3 Non-functional Requirements:

1. Performance: Ensuring fast response times for image classification and maintaining an overall responsive application.
2. Scalability: Designing the application architecture to handle potential increases in user traffic without compromising performance.
3. Security: Implementing secure data transmission practices and ensuring data privacy for user uploads and model outputs.
4. User Interface Design: Creating an intuitive and visually appealing interface for a seamless user experience.
5. Compatibility: Ensuring cross-browser and cross-device compatibility to broaden accessibility for users.

3.4 Feasibility Study

The feasibility study for the proposed project involves a comprehensive examination of technical, economic, and operational aspects to ascertain the viability and success of the endeavor.

3.4.1 Technical Feasibility

Assessing the availability of technical resources required for web application development, machine learning model implementation, and system deployment. Evaluating the technical infrastructure's scalability to accommodate potential increases in user traffic and data volume.

3.4.2 Economic Feasibility

- Budget Constraints: Analyzing budgetary considerations for hardware, software, development tools, and potential licensing fees.
- Return on Investment (ROI): Estimating the potential benefits, such as increased crop yield for farmers, against the costs incurred in developing and maintaining the system.
- Cost-Benefit Analysis: Conducting a cost-benefit analysis to ensure the project's economic viability.

3.4.3 Operational Feasibility

- User Acceptance: Assessing the acceptance and adaptability of the proposed system by end-users, including farmers and agricultural stakeholders.
- Ease of Use: Ensuring the user interface is intuitive and user-friendly for seamless interaction with the application.
- Training Requirements: Identifying any training requirements for end-users or administrators to effectively utilize and maintain the system.

3.4.4 Legal and Ethical Considerations

- Data Privacy: Ensuring compliance with data protection regulations and implementing measures to safeguard user data and privacy.

- Intellectual Property: Addressing any legal considerations related to the use of machine learning models, ensuring compliance with licensing and intellectual property rights.

3.4.5 Risk Analysis

- Identification of Risks: Identifying potential risks, such as technical challenges, delays in development, or unexpected issues in model performance.
- Mitigation Strategies: Developing strategies to mitigate identified risks and minimize their impact on project timelines and outcomes.

Chapter-4: SYSTEM DESIGN AND ARCHITECTURE

In order to classify tomato disease, data processing is done in image dataset. Multiple machines learning models, including CNNs, Vision Transformers, and EfficientNet, undergo training on the preprocessed data. The models are then evaluated based on performance metrics, and the highest-performing model is selected. The chosen model is integrated into the FastAPI backend and web application is developed using ReactJS.

4.1 System Flow Chart

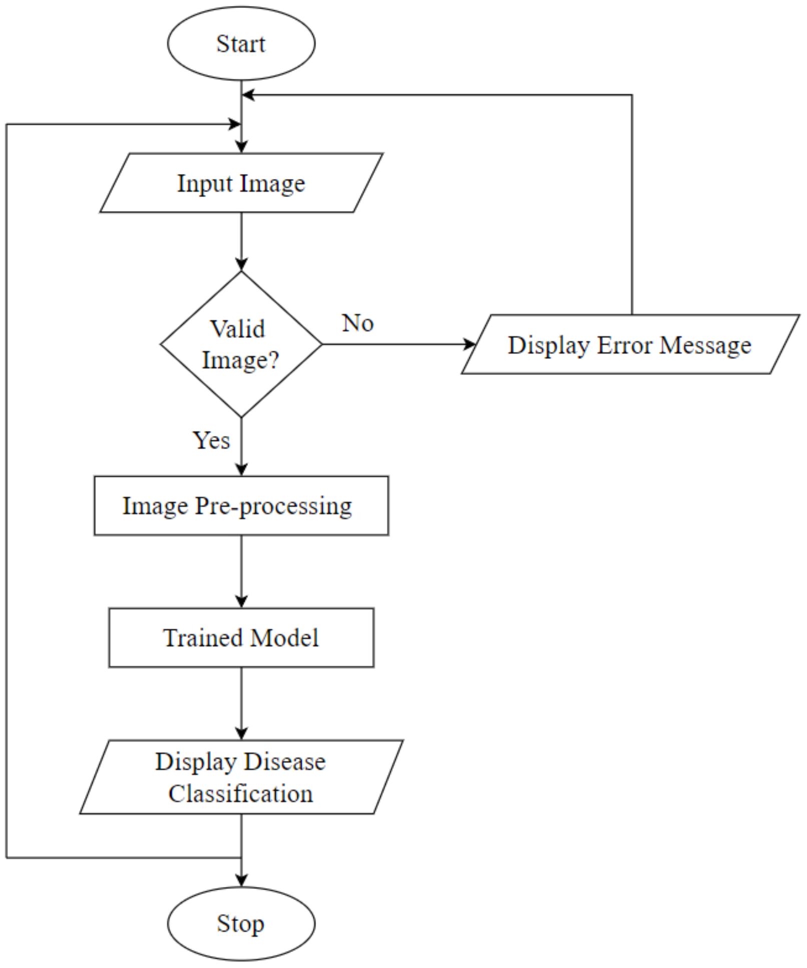


Figure 4.1: System Flow Chart

4.2 Data Flow Diagram

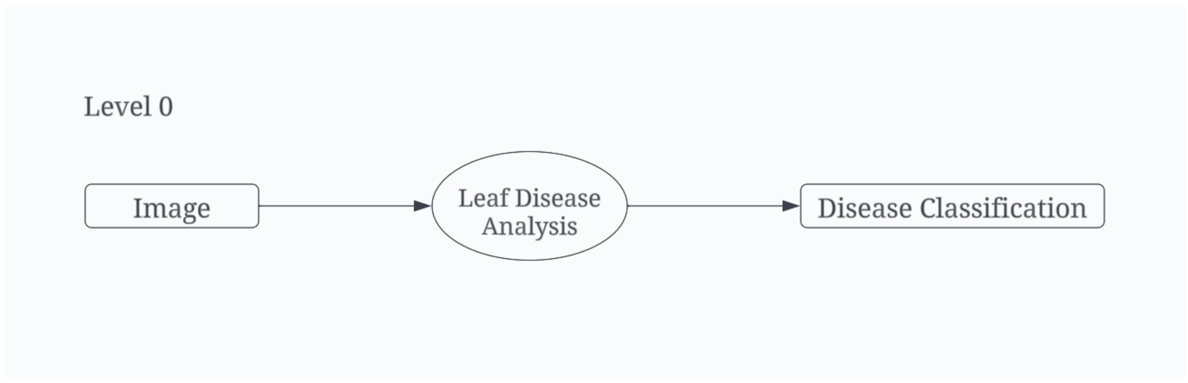


Figure 4.2: DFD level 0

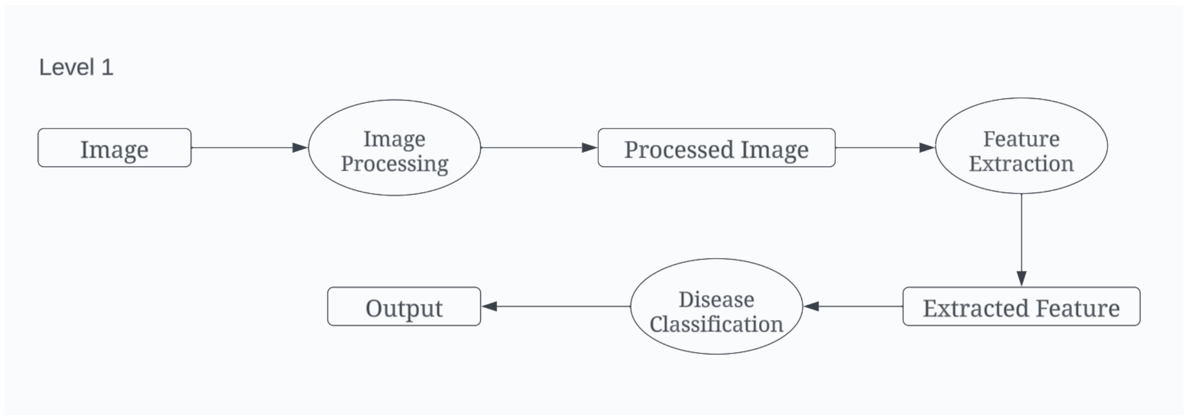


Figure 4.3: DFD level 1

4.3 Use Case Diagram

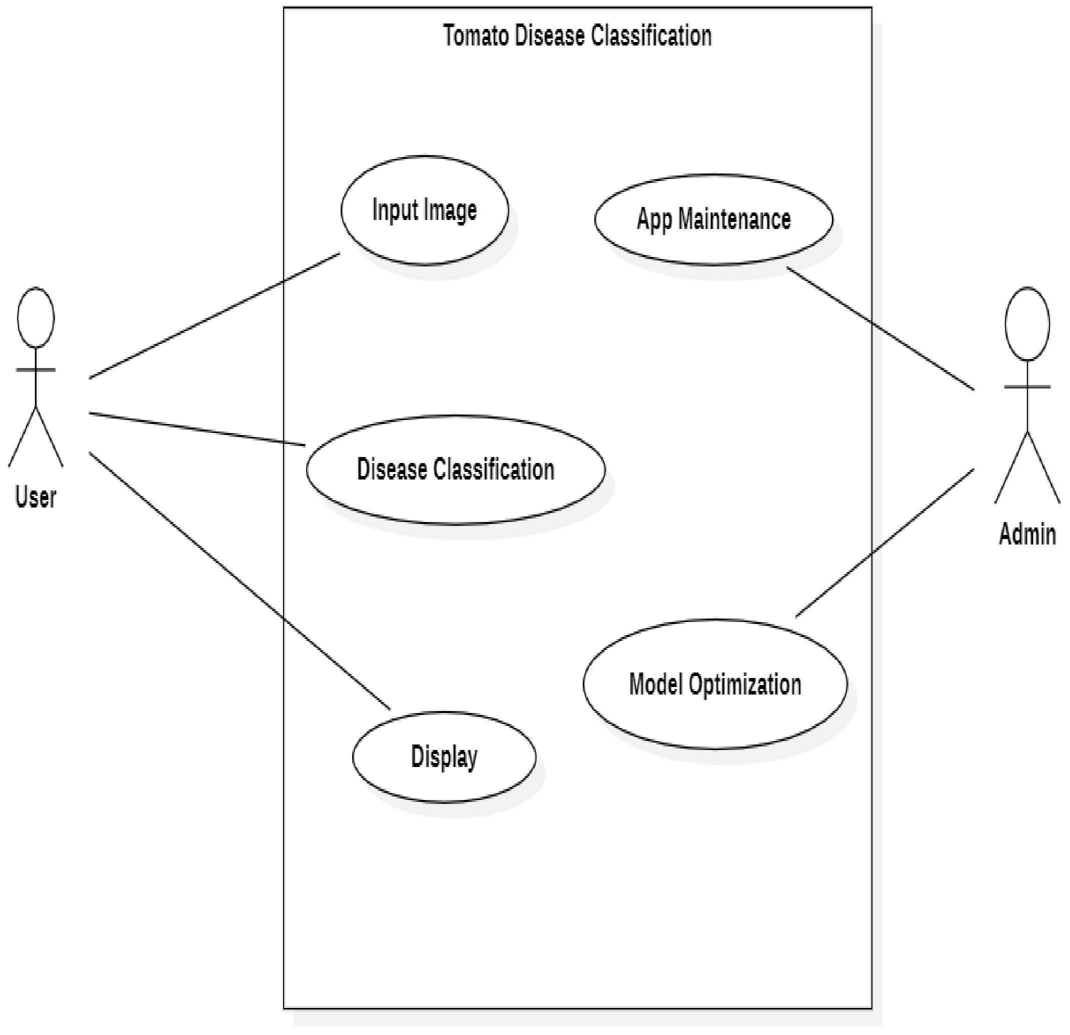


Figure 4.4: Use Case Diagram

CHAPTER-5: METHODOLOGY

5.1 Block Diagram

A block diagram is a visual representation of a system that uses simple, labeled blocks that represent single or multiple items, entities or concepts, connected by lines to show relationships between them.

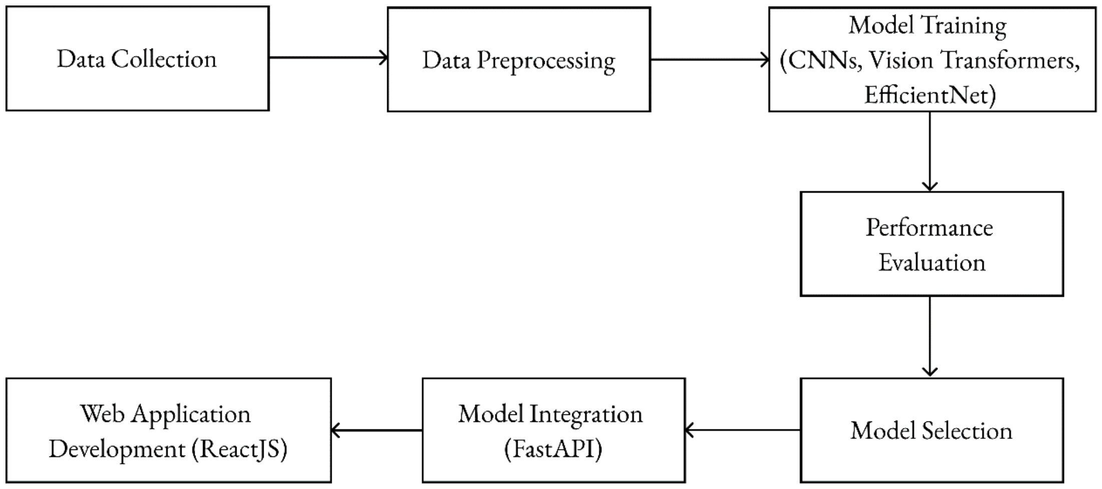

Data Collection
Data Preprocessing
Model Training (CNNs, Vision Transformers, EfficientNet)
Performance Evaluation
Model Selection
Model Integration (FastAPI)
Web Application Development (ReactJS)

Figure 5.1: Block diagram of the system

5.2 Data Collection

The dataset is taken from Kaggle, a reputable platform for machine learning datasets. The dataset consists of over 16,000 tomato leaf images that encapsulate various tomato disease classifications: Bacterial Spot, Curl Virus, Early Blight, Healthy, Late Blight, Leaf Mold, Mosaic Virus, Septoria Leaf Spot, Spider Mites, and Target Spot. The dataset's structure is organized into distinct folders, with each folder corresponding to a specific label, facilitating efficient annotation and model training.

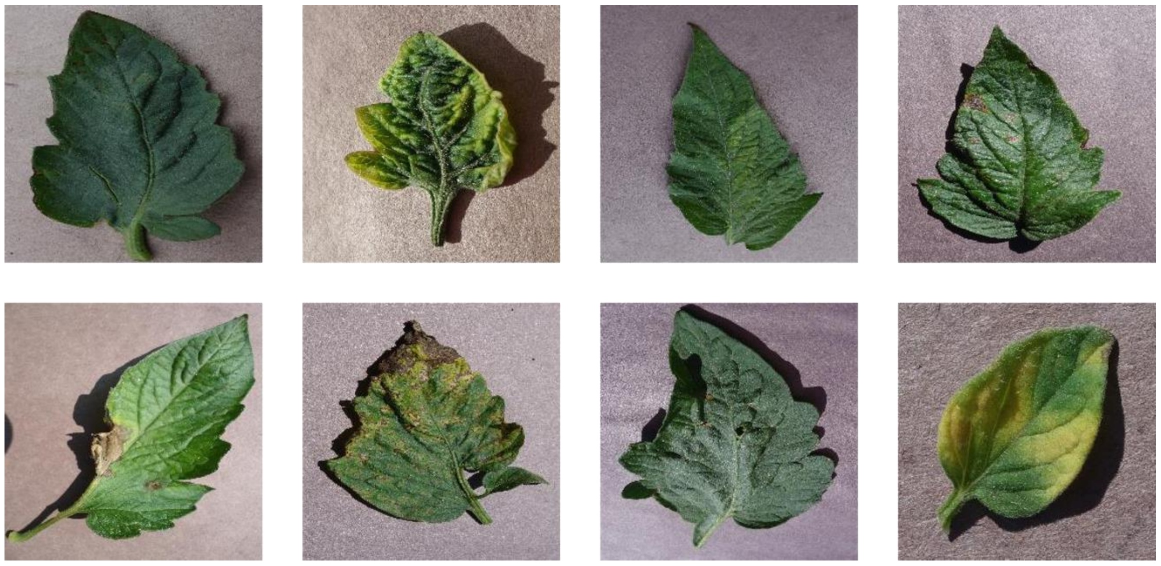


Figure 5.2: Sample images from dataset

(Source: https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf)

5.3 Data Preprocessing

In the data preprocessing phase of the tomato disease classification project, the curated dataset of over 16,000 tomato leaf images obtained from Kaggle undergoes essential transformations to ensure its suitability for machine learning model training. Using OpenCV, all images are resized to a standardized dimension of 256x256 pixels, promoting uniformity in the dataset. Pixel values are normalized to a common scale between 0 and 1, facilitating stable model convergence. To introduce variety and improve model generalization, augmentation techniques, implemented through TensorFlow ImageDataGenerator by applying random transformations such as rotation, flipping, and contrast.

The dataset is partitioned into training, validation, and test sets, with 80% of the data allocated for training, 10% for validation, and another 10% for testing. This partitioning scheme ensures that the model is trained on a majority of the data, validated on a separate subset to optimize hyperparameters, and finally tested on an independent set to assess generalization performance.

To address the imbalance in the dataset, where certain disease categories have a higher number of images compared to others, a strategic approach is implemented. Class weights were computed based on the distribution of classes in the training data. This balancing process aims to prevent the model from favoring overrepresented classes during training, ensuring fair and unbiased learning across all disease categories. Additionally, image augmentation techniques were applied to augment the dataset further, enhancing the diversity of training samples.

5.4 Model Training

The model training phase is a critical step in the development of the tomato disease classification system, involving the training of three distinct deep learning architectures: Convolutional Neural Networks (CNNs), Vision Transformers, and EfficientNet. This phase harnesses the preprocessed dataset to teach the models to classify various tomato diseases.

5.4.1 Convolutional Neural Networks (CNNs):

A Convolutional Neural Network (CNN) is a class of deep neural networks that has proven to be particularly effective in areas such as image recognition, computer vision, and pattern recognition. It employs convolutional layers to automatically and adaptively learn hierarchical representations of features, enabling effective pattern recognition and classification in tasks like image recognition and computer vision.

CNN Architecture

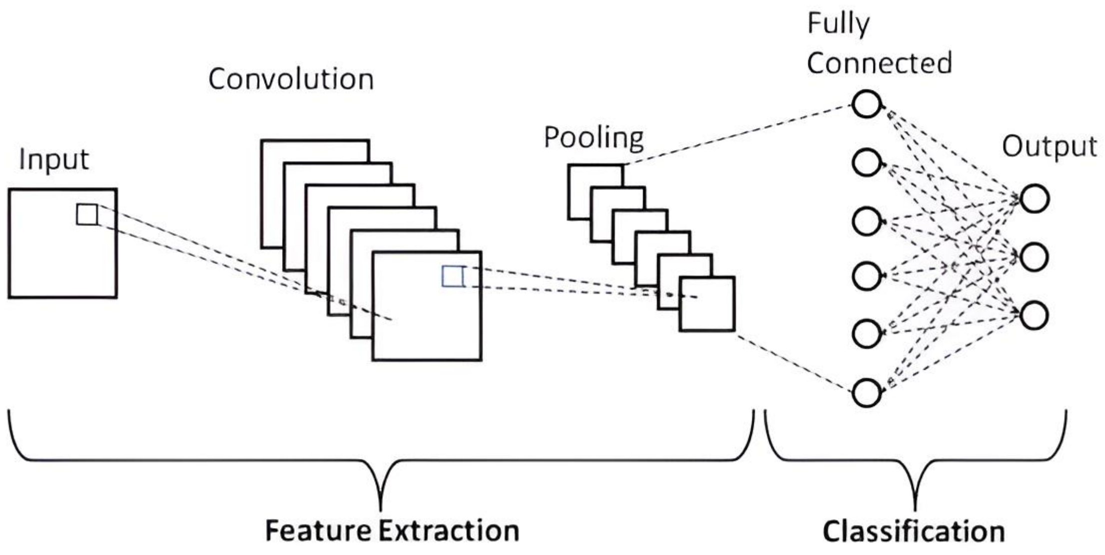


Figure 5.3: CNN Architecture

(Source: https://www.researchgate.net/figure/cnn)

1. Convolutional Layer:

Convolutional layers are the core building blocks of CNNs. They consist of filters (also called kernels) that slide or convolve across the input data to extract features. Each filter learns to detect specific patterns or features, such as edges, textures, or higher-level structures. Typically, a non-linear activation function like ReLU (Rectified Linear Unit) follows the convolutional layer to introduce non-linearity into the model.

2. Pooling Layer:

Pooling layers are used to down sample the spatial dimensions of the input volume, reducing computation in the network and controlling overfitting. Common pooling operations include max pooling and average pooling.

3. Flattening:

The process of converting the final feature maps into a one-dimensional vector to be fed into the fully connected layers.

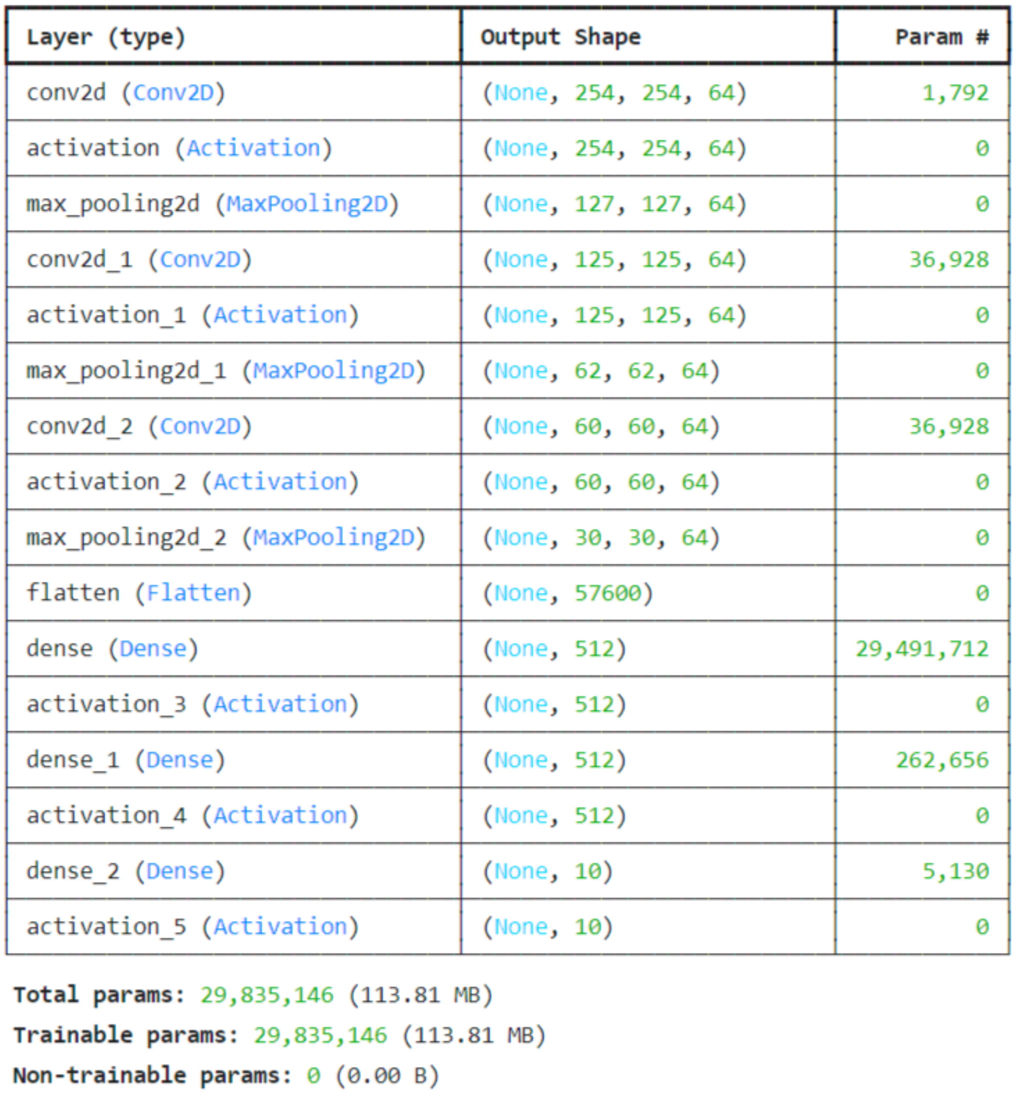
4. Fully Connected (Dense) Layers:

After several convolutional and pooling layers, the high-level reasoning is captured in the flattened output, which is then processed by one or more fully connected layers. These layers make predictions or classifications based on the learned features.

5. Dropout:

Dropout is a regularization technique where randomly selected neurons are ignored during training. This helps prevent overfitting.

Summary of CNN Model Layer



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 64)	1,792
activation (Activation)	(None, 254, 254, 64)	0
max_pooling2d (MaxPooling2D)	(None, 127, 127, 64)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	36,928
activation_1 (Activation)	(None, 125, 125, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 64)	36,928
activation_2 (Activation)	(None, 60, 60, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten (Flatten)	(None, 57600)	0
dense (Dense)	(None, 512)	29,491,712
activation_3 (Activation)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262,656
activation_4 (Activation)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5,130
activation_5 (Activation)	(None, 10)	0

```
Total params: 29,835,146 (113.81 MB)
Trainable params: 29,835,146 (113.81 MB)
Non-trainable params: 0 (0.00 B)
```

5.4.2 Vision Transformers

Vision Transformers (ViT) represent a paradigm shift in image classification by leveraging attention mechanisms to capture long-range dependencies within the tomato leaf image dataset. Unlike traditional Convolutional Neural Networks (CNNs), ViT focus on global relationships among image regions, enabling more nuanced feature extraction.

ViT Architecture

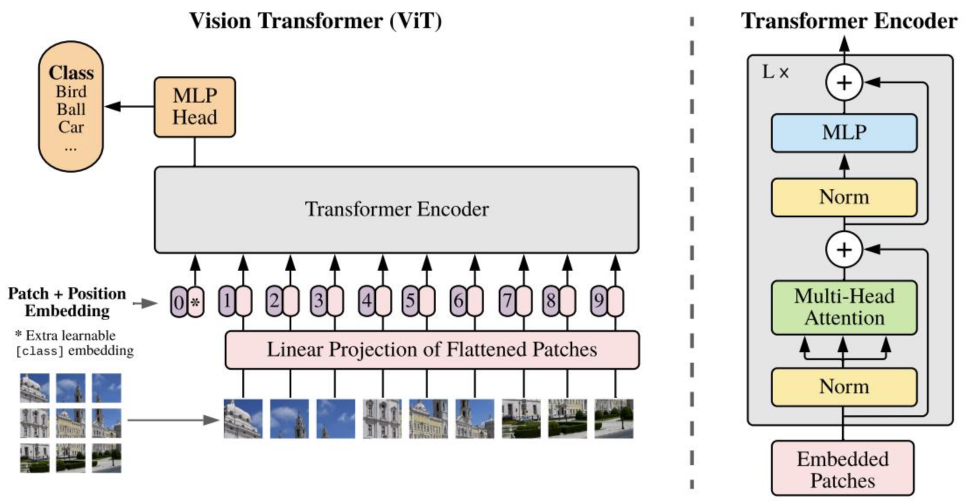


Figure 5.4: ViT Architecture

(Source: https://medium.com/mlearning-ai/an-image-is-worth-16x16-words)

1. Data Input:

Preprocessed tomato leaf images are divided into patches, and these patches are treated as sequential inputs for the ViT model.

2. Patch Embedding:

Patches undergo linear embedding, transforming them into feature vectors that retain spatial information.

3. Transformer Blocks:

The model consists of multiple Transformer blocks, each incorporating self-attention mechanisms. These mechanisms enable the model to weigh the significance of different patches when learning features.

4. Global Context Understanding:

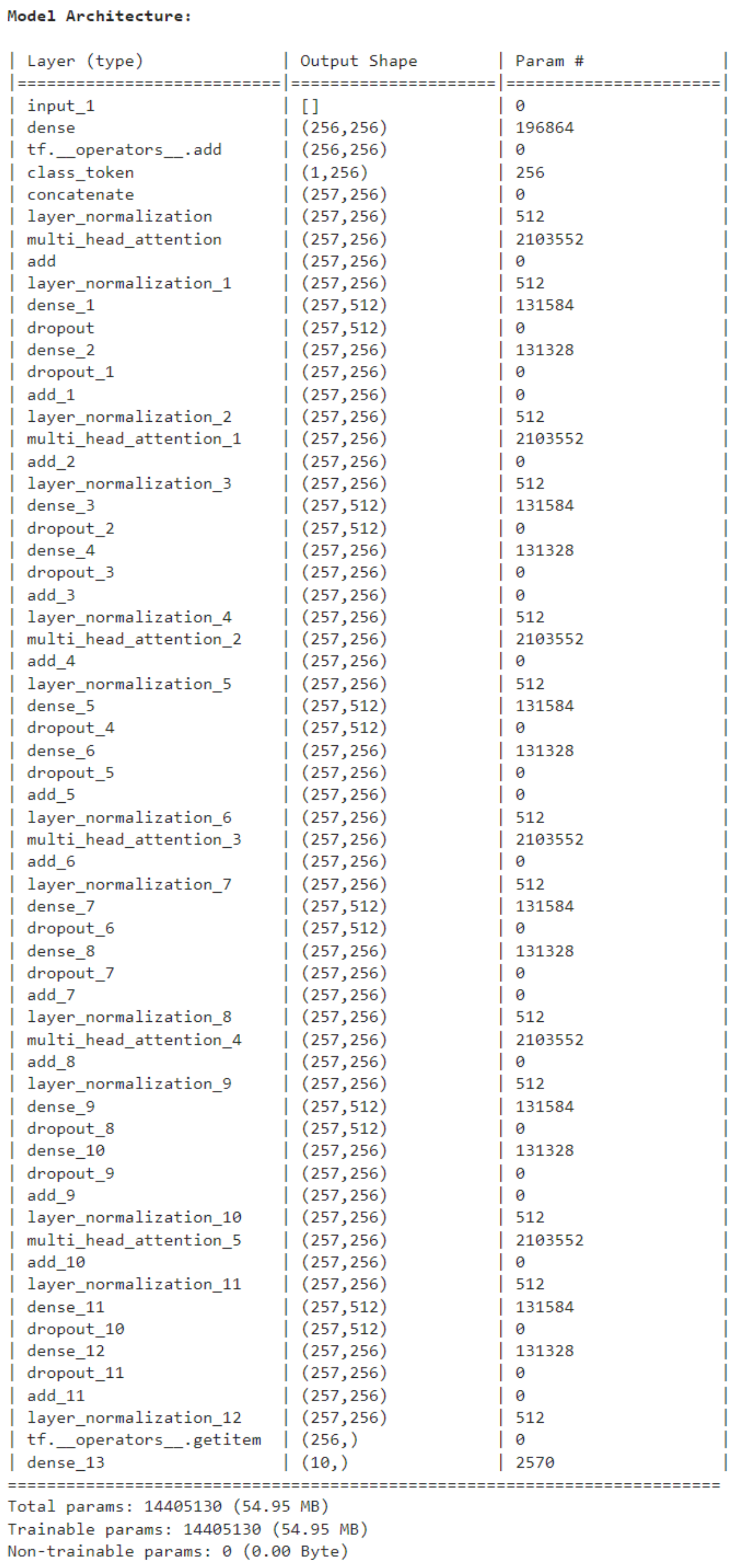
Unlike CNNs,ViTs excel at capturing long-range dependencies between patches, providing a holistic understanding of the image.

5. Classification Head:

Extracted features are passed through a classification head to predict the corresponding tomato disease category.

6. Backpropagation:

During training, the model adjusts weights and biases through backpropagation to minimize the difference between predicted and actual labels.



```
Model Architecture:

| Layer (type)               | Output Shape         | Param #              |
|============================|======================|======================|
| input_1                    | []                   | 0                    |
| dense                      | (256,256)            | 196864               |
| tf.__operators__.add       | (256,256)            | 0                    |
| class_token                | (1,256)              | 256                  |
| concatenate                | (257,256)            | 0                    |
| layer_normalization        | (257,256)            | 512                  |
| multi_head_attention       | (257,256)            | 2103552              |
| add                        | (257,256)            | 0                    |
| layer_normalization_1      | (257,256)            | 512                  |
| dense_1                    | (257,512)            | 131584               |
| dropout                    | (257,512)            | 0                    |
| dense_2                    | (257,256)            | 131328               |
| dropout_1                  | (257,256)            | 0                    |
| add_1                      | (257,256)            | 0                    |
| layer_normalization_2      | (257,256)            | 512                  |
| multi_head_attention_1     | (257,256)            | 2103552              |
| add_2                      | (257,256)            | 0                    |
| layer_normalization_3      | (257,256)            | 512                  |
| dense_3                    | (257,512)            | 131584               |
| dropout_2                  | (257,512)            | 0                    |
| dense_4                    | (257,256)            | 131328               |
| dropout_3                  | (257,256)            | 0                    |
| add_3                      | (257,256)            | 0                    |
| layer_normalization_4      | (257,256)            | 512                  |
| multi_head_attention_2     | (257,256)            | 2103552              |
| add_4                      | (257,256)            | 0                    |
| layer_normalization_5      | (257,256)            | 512                  |
| dense_5                    | (257,512)            | 131584               |
| dropout_4                  | (257,512)            | 0                    |
| dense_6                    | (257,256)            | 131328               |
| dropout_5                  | (257,256)            | 0                    |
| add_5                      | (257,256)            | 0                    |
| layer_normalization_6      | (257,256)            | 512                  |
| multi_head_attention_3     | (257,256)            | 2103552              |
| add_6                      | (257,256)            | 0                    |
| layer_normalization_7      | (257,256)            | 512                  |
| dense_7                    | (257,512)            | 131584               |
| dropout_6                  | (257,512)            | 0                    |
| dense_8                    | (257,256)            | 131328               |
| dropout_7                  | (257,256)            | 0                    |
| add_7                      | (257,256)            | 0                    |
| layer_normalization_8      | (257,256)            | 512                  |
| multi_head_attention_4     | (257,256)            | 2103552              |
| add_8                      | (257,256)            | 0                    |
| layer_normalization_9      | (257,256)            | 512                  |
| dense_9                    | (257,512)            | 131584               |
| dropout_8                  | (257,512)            | 0                    |
| dense_10                   | (257,256)            | 131328               |
| dropout_9                  | (257,256)            | 0                    |
| add_9                      | (257,256)            | 0                    |
| layer_normalization_10     | (257,256)            | 512                  |
| multi_head_attention_5     | (257,256)            | 2103552              |
| add_10                     | (257,256)            | 0                    |
| layer_normalization_11     | (257,256)            | 512                  |
| dense_11                   | (257,512)            | 131584               |
| dropout_10                 | (257,512)            | 0                    |
| dense_12                   | (257,256)            | 131328               |
| dropout_11                 | (257,256)            | 0                    |
| add_11                     | (257,256)            | 0                    |
| layer_normalization_12     | (257,256)            | 512                  |
| tf.__operators__.getitem   | (256,)               | 0                    |
| dense_13                   | (10,)                | 2570                 |
=========================================================================
Total params: 14405130 (54.95 MB)
Trainable params: 14405130 (54.95 MB)
Non-trainable params: 0 (0.00 Byte)
```

5.4.3 EfficientNet

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. For example, if we want to use 2^N times more computational resources, then we can simply increase the network depth by α^N, width by β^N, and image size by γ^N, where α, β, γ are constant coefficients determined by a small grid search on the original small model. EfficientNet uses a compound coefficient Φ to uniformly scales network width, depth, and resolution in a principled way.

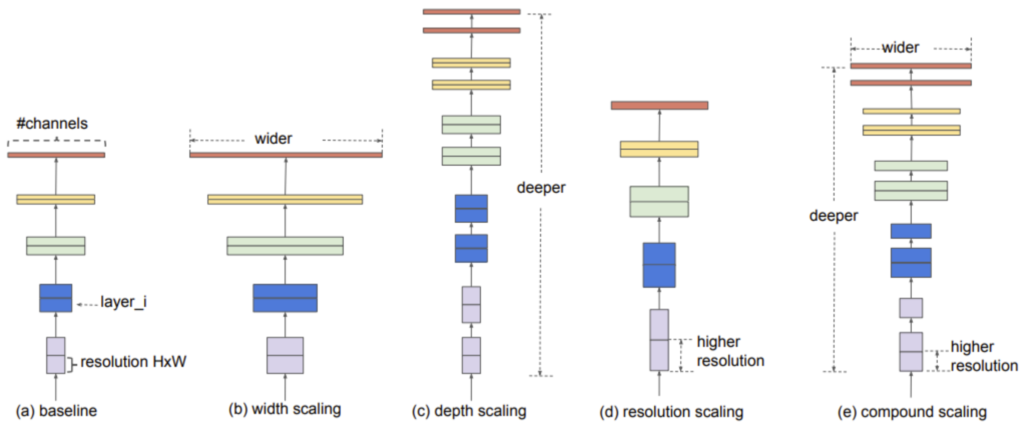


Figure 5.5: EfficientNet Architecture

(Source: https://medium.com/efficientnet-the-most-powerful-cnn-architecture)

1. Data Input

Preprocessed tomato leaf images are input into the EfficientNet architecture during the training process.

2. Compound Scaling

The model undergoes compound scaling, adjusting depth, width, and resolution simultaneously to optimize efficiency.

3. Baseline Network

The baseline network serves as the foundation, and scaling coefficients determine the specific architecture size.

4. Depth, Width, Resolution Balancing

EfficientNet dynamically balances depth (number of layers), width (number of channels), and resolution to achieve an optimal configuration.

5. Backpropagation

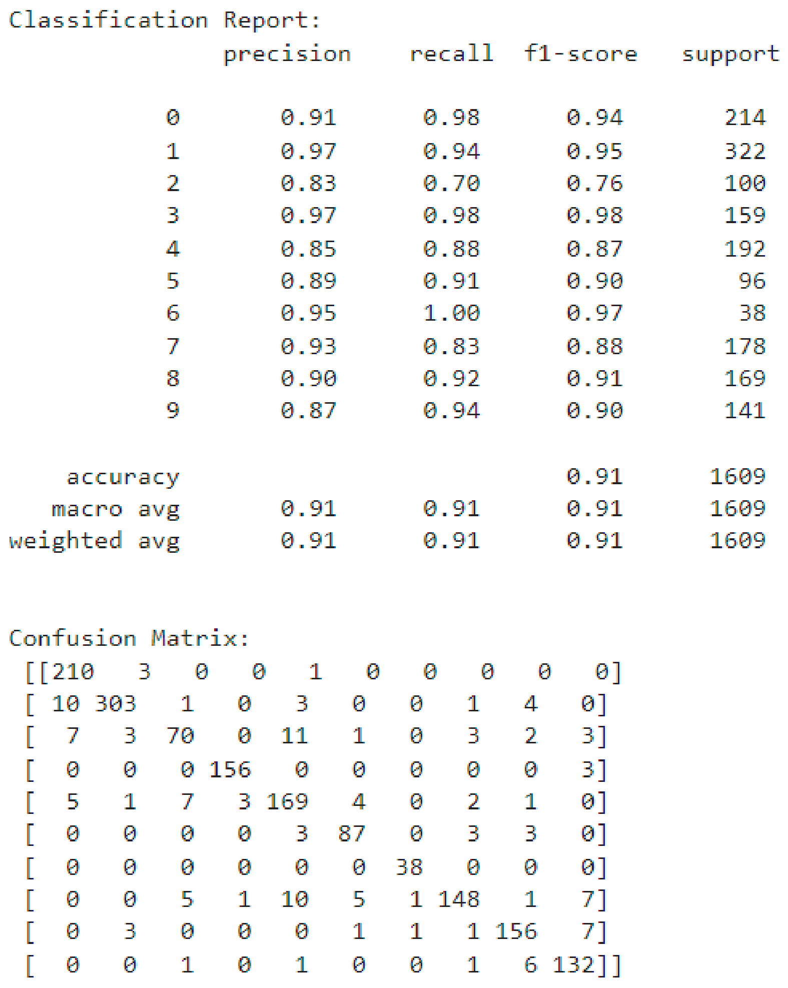
During training, the model adjusts weights and biases through backpropagation to minimize the difference between predicted and actual disease labels.

6. Optimization

Systematic optimization techniques, such as stochastic gradient descent, fine-tune model parameters for enhanced performance.

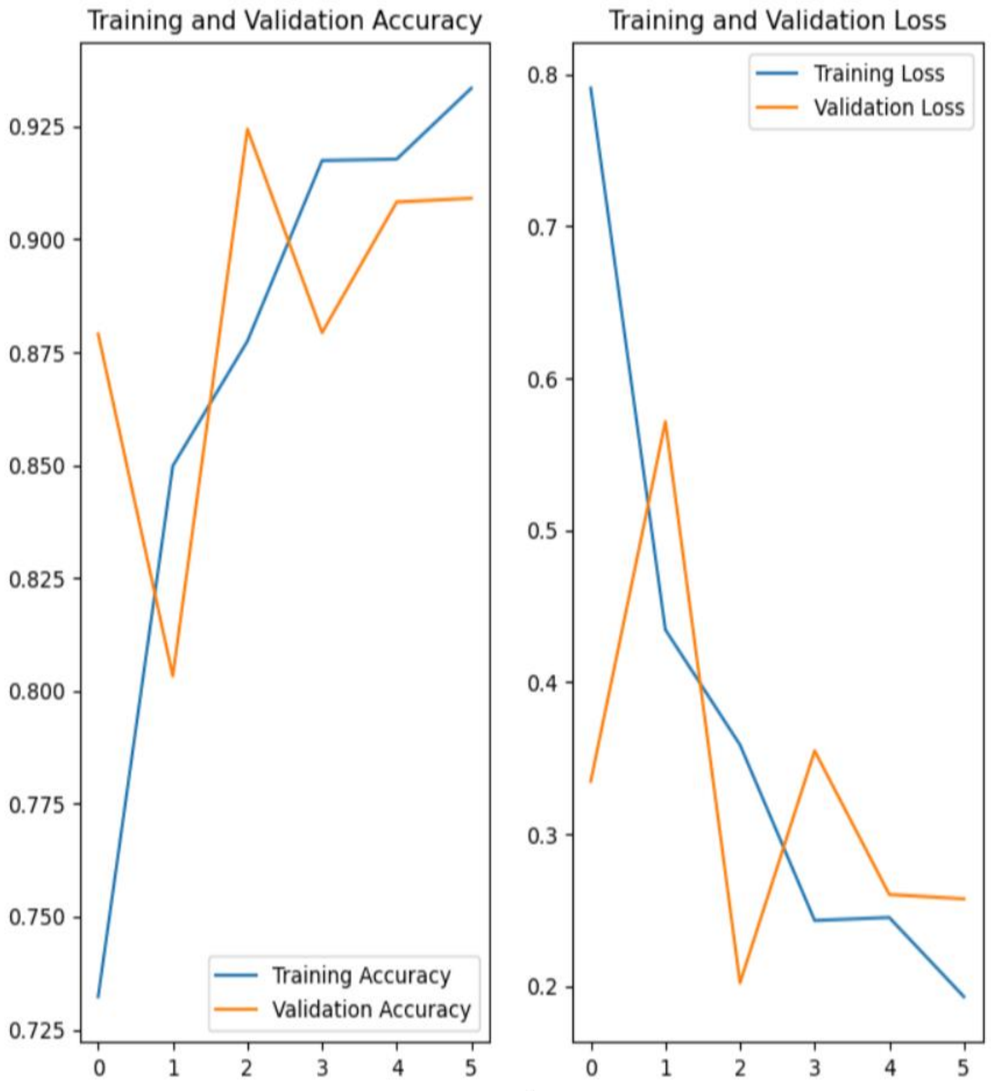
5.5 Performance Evaluation

5.5.1 CNN

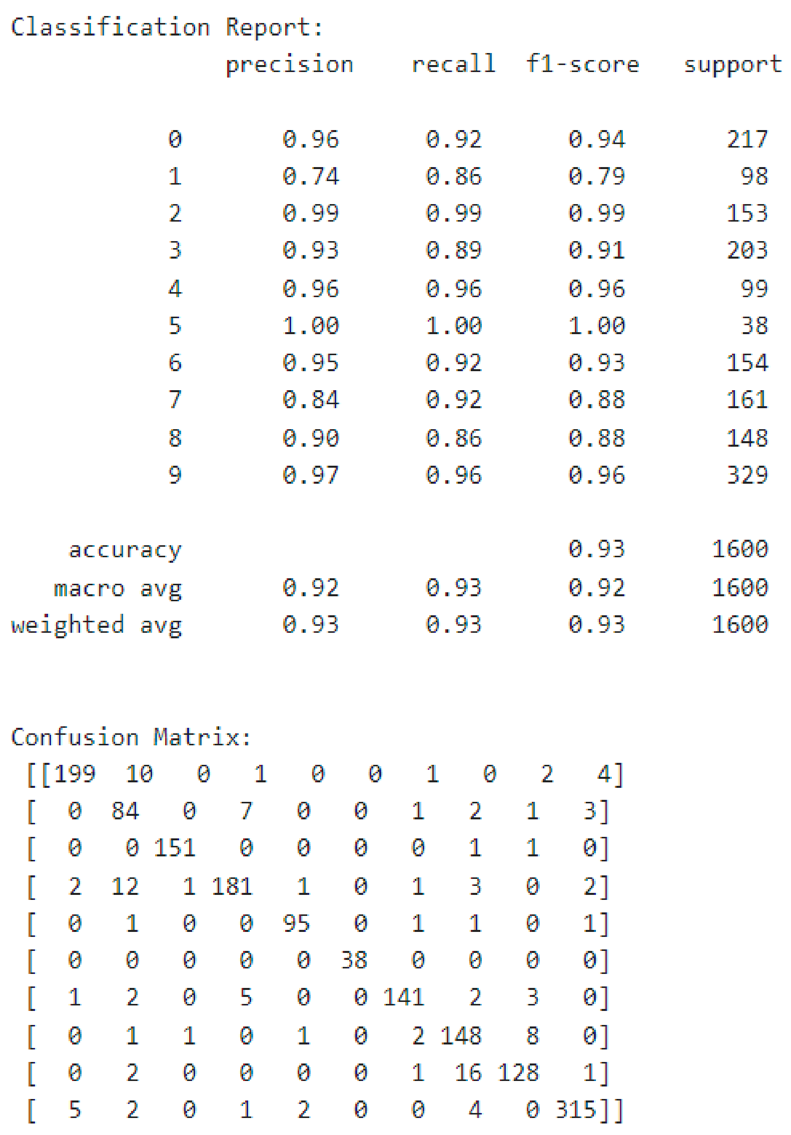


```
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       214
           1       0.97      0.94      0.95       322
           2       0.83      0.70      0.76       100
           3       0.97      0.98      0.98       159
           4       0.85      0.88      0.87       192
           5       0.89      0.91      0.90        96
           6       0.95      1.00      0.97        38
           7       0.93      0.83      0.88       178
           8       0.90      0.92      0.91       169
           9       0.87      0.94      0.90       141

    accuracy                           0.91      1609
   macro avg       0.91      0.91      0.91      1609
weighted avg       0.91      0.91      0.91      1609


Confusion Matrix:
 [[210   3   0   0   1   0   0   0   0   0]
 [ 10 303   1   0   3   0   0   1   4   0]
 [  7   3  70   0  11   1   0   3   2   3]
 [  0   0   0 156   0   0   0   0   0   3]
 [  5   1   7   3 169   4   0   2   1   0]
 [  0   0   0   0   3  87   0   3   3   0]
 [  0   0   0   0   0   0  38   0   0   0]
 [  0   0   5   1  10   5   1 148   1   7]
 [  0   3   0   0   0   1   1   1 156   7]
 [  0   0   1   0   1   0   0   1   6 132]]
```

Below are the line plots illustrating training validation accuracy and validation loss.

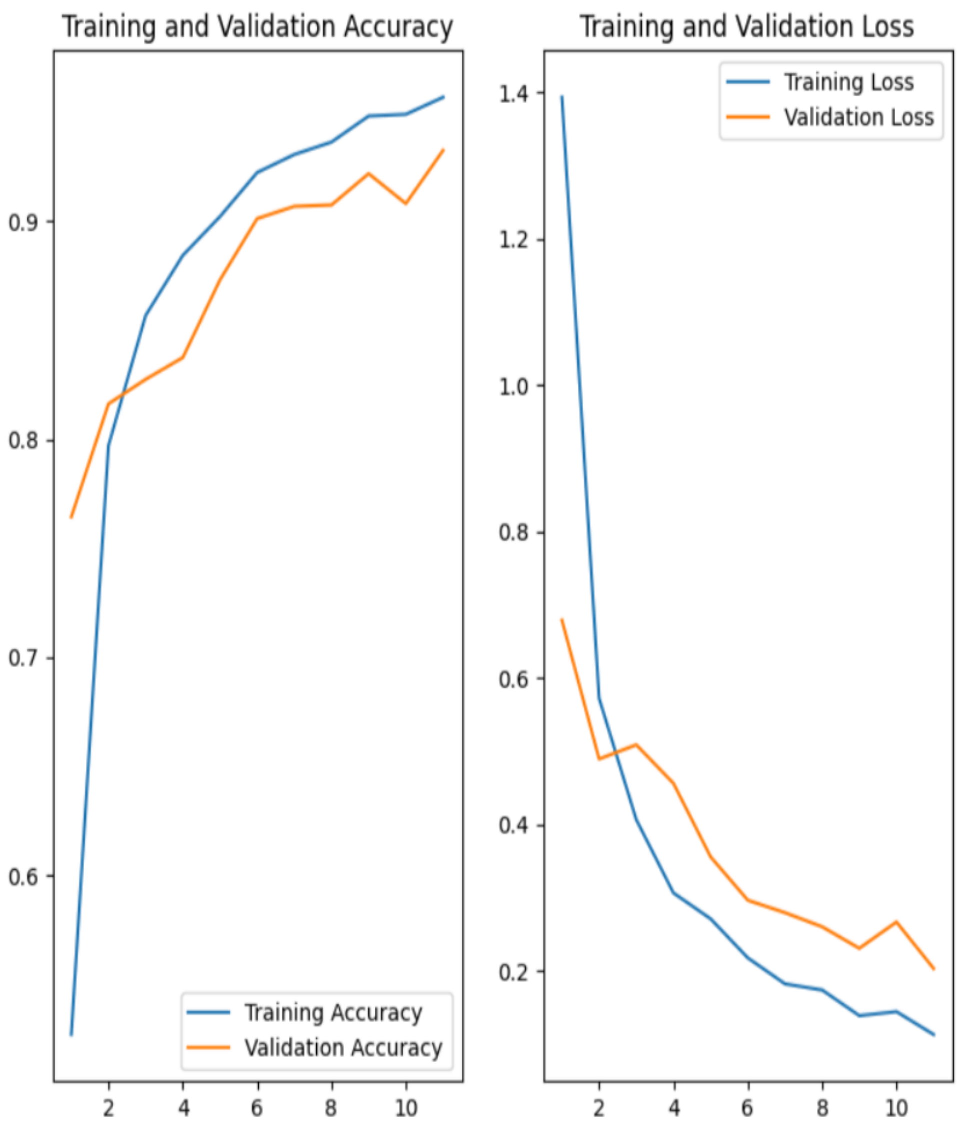


5.5.2 Vision Transformer

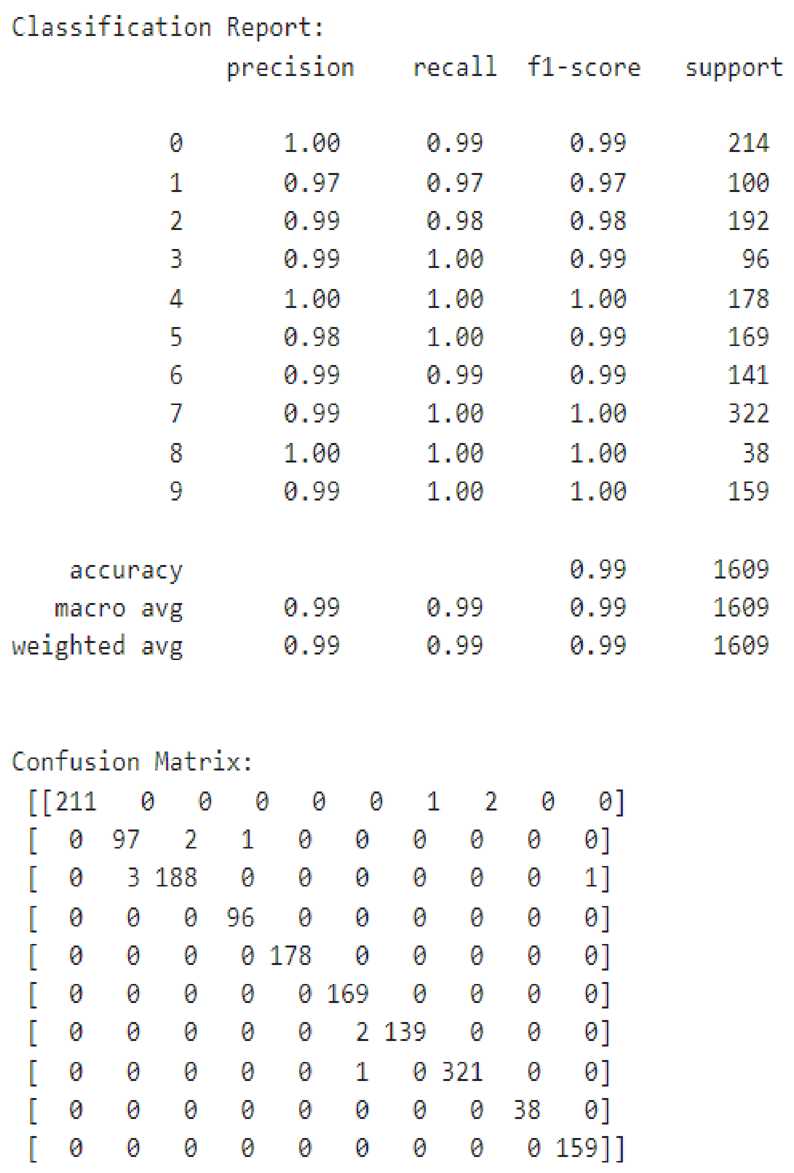


```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94       217
           1       0.74      0.86      0.79        98
           2       0.99      0.99      0.99       153
           3       0.93      0.89      0.91       203
           4       0.96      0.96      0.96        99
           5       1.00      1.00      1.00        38
           6       0.95      0.92      0.93       154
           7       0.84      0.92      0.88       161
           8       0.90      0.86      0.88       148
           9       0.97      0.96      0.96       329

    accuracy                           0.93      1600
   macro avg       0.92      0.93      0.92      1600
weighted avg       0.93      0.93      0.93      1600


Confusion Matrix:
 [[199  10   0   1   0   0   1   0   2   4]
 [  0  84   0   7   0   0   1   2   1   3]
 [  0   0 151   0   0   0   0   1   1   0]
 [  2  12   1 181   1   0   1   3   0   2]
 [  0   1   0   0  95   0   1   1   0   1]
 [  0   0   0   0   0  38   0   0   0   0]
 [  1   2   0   5   0   0 141   2   3   0]
 [  0   1   1   0   1   0   2 148   8   0]
 [  0   2   0   0   0   0   1  16 128   1]
 [  5   2   0   1   2   0   0   4   0 315]]
```

Below are the line plots illustrating training validation accuracy and validation loss.

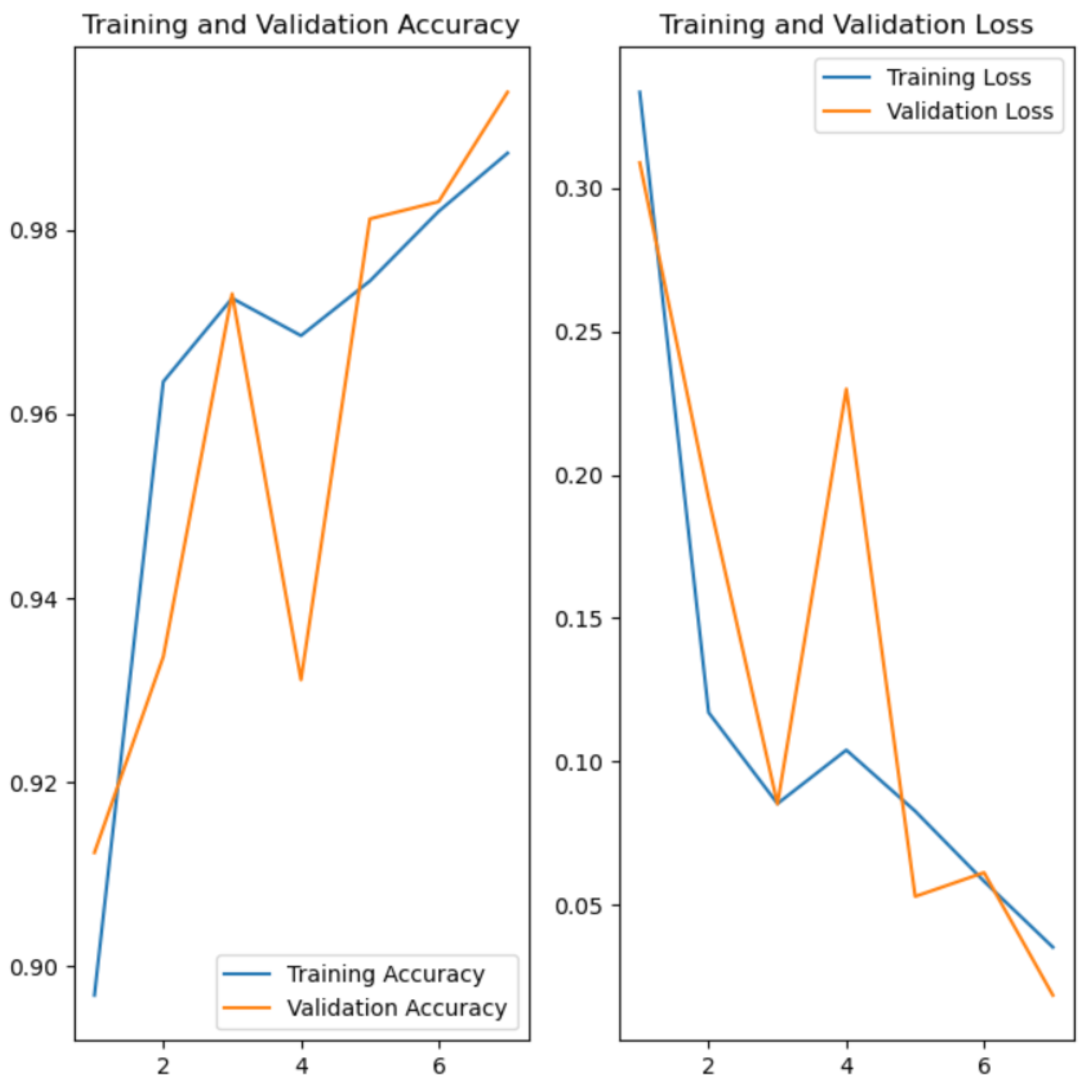


5.5.3 EfficientNet



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       214
           1       0.97      0.97      0.97       100
           2       0.99      0.98      0.98       192
           3       0.99      1.00      0.99        96
           4       1.00      1.00      1.00       178
           5       0.98      1.00      0.99       169
           6       0.99      0.99      0.99       141
           7       0.99      1.00      1.00       322
           8       1.00      1.00      1.00        38
           9       0.99      1.00      1.00       159

    accuracy                           0.99      1609
   macro avg       0.99      0.99      0.99      1609
weighted avg       0.99      0.99      0.99      1609


Confusion Matrix:
 [[211   0   0   0   0   0   1   2   0   0]
 [  0  97   2   1   0   0   0   0   0   0]
 [  0   3 188   0   0   0   0   0   0   1]
 [  0   0   0  96   0   0   0   0   0   0]
 [  0   0   0   0 178   0   0   0   0   0]
 [  0   0   0   0   0 169   0   0   0   0]
 [  0   0   0   0   0   2 139   0   0   0]
 [  0   0   0   0   0   1   0 321   0   0]
 [  0   0   0   0   0   0   0   0  38   0]
 [  0   0   0   0   0   0   0   0   0 159]]
```

Below are the line plots illustrating training validation accuracy and validation loss.



5.6 Model Selection

After a comprehensive evaluation of various deep learning models for tomato disease classification, the choice of EfficientNet emerges as the optimal solution. With a precision and recall of 0.99, EfficientNet demonstrates superior performance compared to other models such as the Vision Transformer and CNN. Despite achieving competitive accuracy scores, both the Vision Transformer and CNN fall short in precision and recall, highlighting the efficiency and effectiveness of EfficientNet in accurately identifying and classifying tomato diseases. Moreover, the remarkable performance of EfficientNet is achieved within a remarkably short training period of just seven epochs, underscoring its ability to quickly converge to a highly accurate solution. Thus, based on its exceptional precision, recall, and efficiency, EfficientNet stands out as the model of choice for tomato disease classification in this study.

5.7 Model Integration

Integrating the EfficientNet model into FastAPI facilitates seamless deployment and utilization of the high-performing tomato disease classification system. Leveraging the FastAPI framework's asynchronous capabilities and robust interface, EfficientNet's predictive power can be harnessed in real-time applications with minimal latency. By encapsulating the model within FastAPI endpoints, users can easily interact with the classification system through simple HTTP requests, enabling scalable and efficient deployment across various platforms. Overall, integrating EfficientNet into FastAPI empowers users to leverage state-of-the-art deep learning capabilities for tomato disease classification in a fast, reliable, and user-friendly manner.

5.8 Web Application Development

ReactJS in conjunction with FastAPI, offers a seamless user experience for image classification tasks. In this setup, users can upload images directly through the React.js frontend interface. Upon image submission, a POST request is triggered, sending the image data to the FastAPI backend endpoint. Here, the image undergoes classification using the EfficientNet model.

Once the classification process is complete, the result is sent back to the React.js frontend, where it is dynamically displayed to the user. This end-to-end workflow ensures smooth interaction and real-time feedback, enhancing user engagement and satisfaction.

CHAPTER-6: RESULT AND ANALYSIS

In our tomato disease classification project, we compared the performance of three different deep learning models: Convolutional Neural Networks (CNN), Vision Transformers, and EfficientNet. After training and evaluation, we obtained the following classification results:

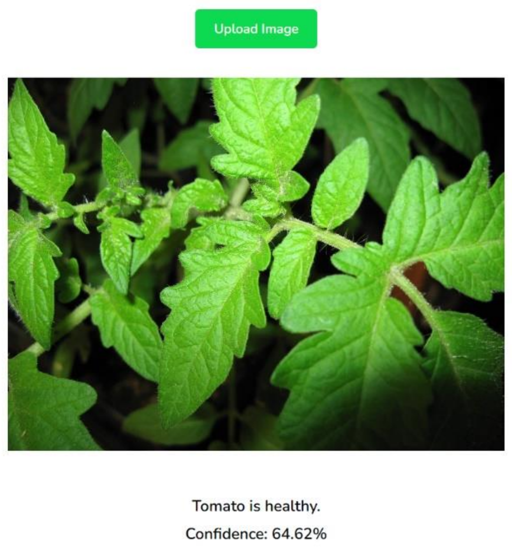
- EfficientNet: Precision = 0.99, Recall = 0.99, Accuracy = 0.99
- Vision Transformer: Precision = 0.92, Recall = 0.93, Accuracy = 0.93
- CNN: Precision = 0.91, Recall = 0.91, Accuracy = 0.91

The superior performance of EfficientNet can be attributed to its efficient architecture design, which balances model complexity and computational efficiency. EfficientNet achieves state-of-the-art performance by scaling the model's depth, width, and resolution simultaneously, allowing it to adapt to various classification tasks effectively.

In contrast, while Vision Transformers and CNN also achieved respectable performance, they fell short compared to EfficientNet. Vision Transformers, despite being a promising architecture for image classification tasks, exhibited slightly lower precision, recall, and accuracy scores than EfficientNet. Similarly, CNN, a traditional and widely used architecture for image classification, demonstrated good performance but was outperformed by EfficientNet in terms of classification accuracy.

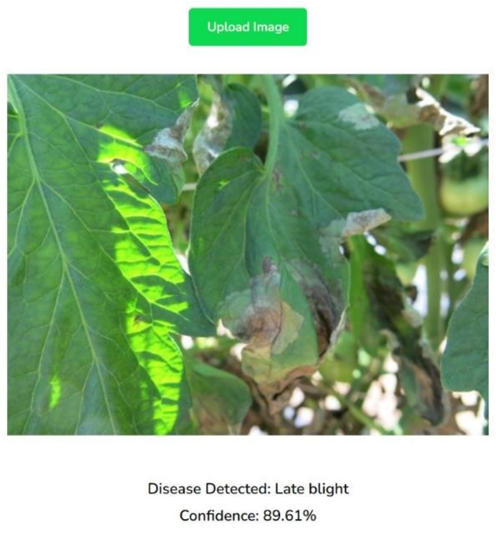
These results highlight the importance of choosing the right model architecture for the task at hand. In our case, EfficientNet proved to be the most suitable choice for tomato disease classification, offering superior performance and accuracy. The findings from this study can guide future research and application development in agricultural management and crop health monitoring, where accurate and efficient disease classification is crucial for maintaining crop yield and quality.

Observations on external dataset



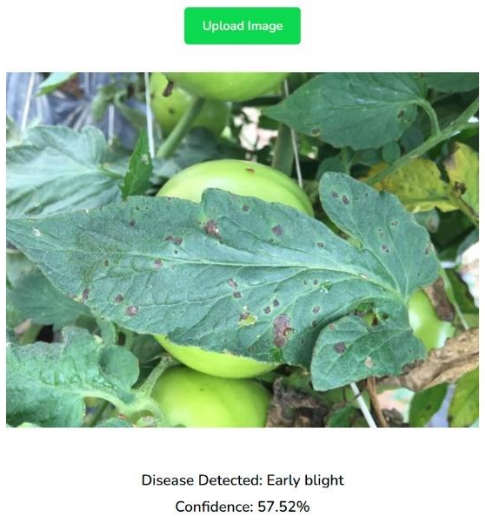
Tomato is healthy.
Confidence: 64.62%

The tomato leaf is diagnosed as Healthy with a 64% confidence level, aligning with its labeled condition as Healthy.



Disease Detected: Late blight
Confidence: 89.61%

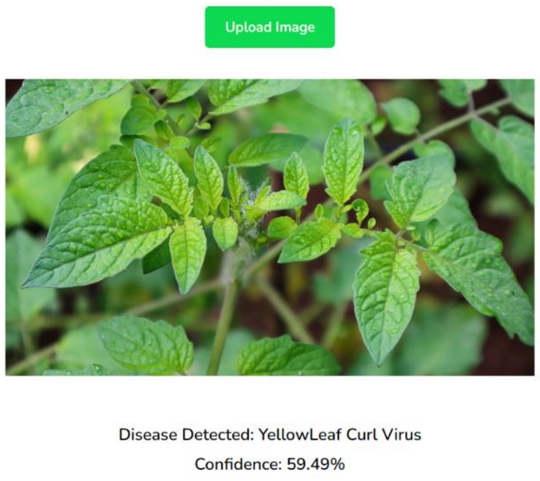
The tomato leaf is diagnosed as Late Blight with a 89% confidence level, aligning with its labeled condition as Late Blight.



Disease Detected: Early blight

Confidence: 57.52%

The tomato leaf is diagnosed as Early Blight with a 57% confidence level, aligning with its labeled condition as Early Blight.



Disease Detected: YellowLeaf Curl Virus

Confidence: 59.49%

The tomato leaf is diagnosed as Yellow Leaf Curl Virus with a 59% confidence level, opposing with its labeled condition as Healthy.

CHAPTER-7: CONCLUSION

We trained three different deep learning models: CNN, Vision Transformers and EfficientNet on same dataset. Next, evaluation of models was performed on test dataset. Henceforth, classification report and confusion matrix were analyzed. All these stages concluded that EfficientNet was the best deep learning model for tomato disease classification, comparatively. Hence, we conclude the report stating the objective accomplished.

CHAPTER-8: LIMITATIONS

Below are the limitations of our project:

- Generalization of findings may be limited by the specific characteristics of the dataset used.
- Evaluation metrics used in this study may not fully capture real-world complexities of tomato disease classification.
- The performance of the model may vary based on the quality of the input image.
- The task performed in this project was classification, not detection. As a result, the trained model generates prediction for images that do not contain tomato leaves, leading to inevitable misclassifications.

CHAPTER-9: FUTURE ENHANCEMENTS

Utilizing the following methods could significantly enhance the model's performance:

- Collection of additional data from diverse sources would improve the decision-making ability of the model.
- Implementing more advanced image augmentation techniques to further diversify the training dataset could improve model generalization.
- Exploring other hyperparameter configurations could improve the generalization ability of the model for tomato disease classification.
- Extend the project scope to include detection tasks, enabling the identification and localization of tomato diseases within images.
- Evaluating alternative optimizers, in addition to Adam, could lead to improved performance and convergence for the model.

REFERENCES

1. K. Shaheed, I. Qureshi, et al., "EfficientRMT-Net—An Efficient ResNet-50 and Vision Transformers Approach for Classifying Plant Leaf Diseases," Nov. 30, 2023.

2. L. Li, S. Zhang, et al., "Plant Disease Detection and Classification by Deep Learning—A Review," IEEE, Apr. 08, 2021.

3. N. K. Trivedi et al., "Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network," Nov. 30, 2021.

4. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, 2017.

5. D. Gandhi, V. Shah, and P. M. Chawan, "A Vision Transformer Approach for Classification on a Small-Sized Image Dataset," Dec. 2022.

6. M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," 2020.

7. A. D. Saputra et al., "Comparison of Accuracy in Detecting Tomato Leaf Disease with GoogleNet VS EfficientNetB3," vol. 8, no. 2, April 2023.