BACHELOR OF SCIENCE IN ELECTRONIC AND TELECOMMUNICATION ENGINEERING

Tomato Feature Analysis and Detection in Horticulture using Deep Learning Algorithms.

Submitted To
Miskatur Rahman
T191071

Supervised By
Engr. Syed Zahidur Rashid
Assistant Professor
Dept. of ETE,IIUC

Department of Electronic and Telecommunication Engineering(ETE)
International Islamic University Chittagong
Kumira, Sitakunda, Chattogram-4318, Bangladesh.

February 2024

DECLARATION OF CANDIDATE

The work in this thesis, except for properly cited quotations and summaries, is my own, I hereby declare.

12 March 2024

_____

(Signature of Candidate)
Miskatur Rahman
ID:T191071
Program: BSc. In ETE
Academic Year: Spring 2019

CERTIFICATE OF APPROVAL

The thesis "Tomato Feature Analysis and Detection in Horticulture using Deep Learning Algorithms" was submitted by **Miskatur Rahman** bearing Matric ID **T191071** to the Department of Electronic & Telecommunication Engineering. I hereby state that after reading this thesis, I believe it to be of sufficient quality and scope to be awarded the degree of BSc. in Electronic & Telecommunication Engineering.

12 March 2024

Supervisor
Engr. Syed Zahidur Rashid
Assistant Professor & Chairman
Dept of ETE,IIUC

ACKNOWLEDGEMENT

I extend my sincere gratitude to Almighty Allah for granting me the fortitude and ability to successfully complete this thesis.

My heartfelt appreciation is reserved for my supervisor, Syed Zahidur Rashid, Assistant Professor in the Department of Electronic & Telecommunication Engineering at the International Islamic University Chittagong (IIUC). His unwavering support and guidance played an instrumental role at various stages of this academic endeavor. In moments of challenges, Professor Rashid provided invaluable assistance, ensuring the successful culmination of this thesis. I also express my gratitude to all those who offered guidance throughout this journey, aiding whenever needed. Their insightful comments and suggestions significantly contributed to the refinement of my work. My gratitude extends to everyone who played a role, no matter how small, in making this achievement possible.

I am truly thankful for the collective effort and support that has shaped the successful completion of this thesis.

ABSTRACT

Tomatoes fruits, a pivotal constituent of tomato plants, with a primary emphasis on elucidating the mechanisms governing their quality formation during the ripening process. Against the backdrop of heightened interest in the tomato industry, the research endeavours to augment the efficacy and success of automatic detection under greenhouse tomato conditions, a pivotal facet for the progression of contemporary agricultural practices. The paper introduces a ground-breaking method using convolutional neural networks to accurately classify tomato fruits based on their ripeness and overall condition. The study adopts a modified ResNet50V2 architecture as the underpinning framework for the CNN model, renowned for its effectiveness in image classification tasks. The outcomes demonstrate a commendable 95.36% accuracy in categorizing tomato fruits into four distinct classes: unripe, ripe, old, and damaged.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolution Neural Network
ADAM	Adaptive Moment Estimation
ADAMAX	Adaptive moment estimation with Maximum
ReLU	Rectified Linear Unit
RMSProp	Root Mean Square Propagation
ResNet	Residual Network

CHAPTER 1
Introduction

1.1 Introduction

An inventive approach to tomato fruit quality detection is being explored through machine learning or computer vision technology, which is capable of accurately assessing the ripeness and quality of fruits based on various factors such as fresh appearance, color, size, and texture. A tomato is a fruit that is ready to be picked after it reaches commercial maturity. Tomato, commonly used in various culinary dishes, is known for its rich flavor and nutritional value and is a staple ingredient in many cuisines around the world. Its vibrant red color and juicy texture make it visually appealing, while its high levels of vitamins and antioxidants contribute to its nutritional value. The use of machine learning and computer vision in tomato fruit quality detection can revolutionize the agricultural industry by providing efficient and accurate methods for assessing the freshness and quality of tomatoes, ensuring that only the best produce reaches consumers' plates. Since tomatoes are cultivated in a variety of temperatures and locales, it might be difficult to identify and locate them. To guarantee that only ripe and healthy tomatoes are harvested, hence enhancing the overall quality of the crop, it is crucial to precisely identify and localize tomatoes in images. Tomato localization and identification may be made more efficient and less prone to human error by automating the procedure. This may be accomplished by using Computer vision technology, as it can evaluate fruit imagery and classify them according to factors such freshness and quality.

In the realm of agriculture, the integration of deep learning technologies is steering a transformative era, with convolutional neural networks (CNN) leading the charge [1].These advanced technologies are not only revolutionizing the detection of tomato leaf diseases but also hold immense promise for the broader field of crop identification. In particular, plant factories, emblematic of vertical agriculture, have emerged as groundbreaking solutions for sustainable crop production, ensuring a consistent year-round supply of vegetables, with tomatoes taking center stage.

However, the dense foliage characteristic of tomato plants presents a significant challenge to detection accuracy, particularly for small-target varieties. Recent

initiatives have showcased the versatility and efficacy of diverse CNN architectures, including AlexNet, GoogLeNet, VGGNet, and ResNet, in addressing this challenge.

Researchers have leveraged these architectures to undertake tasks ranging from the creation of candidate structures to precise bounding box positioning through regression, specifically in the identification of tomato leaf diseases [2].

Recent strides in fruit identification have unveiled the considerable potential of CNNs. Pioneering initiatives have deployed rapid R-CNN and modified YOLOV3 for the detection of ripe and diseased tomatoes, offering glimpses into the future of automated crop monitoring [3]. Despite these advancements, the broader challenge of enhancing the accuracy of overall crop detection persists, necessitating the development of advanced CNN techniques tailored for agriculture [4]. A recent breakthrough involves the integration of semantic segmentation algorithms, promising to elevate the precision of identifying distinct crop parts.

In parallel, current detection models, often reliant on intricate and heavyweight architectures, not only impede accurate identification but also hinder the deployment of robots for essential operations within plant factories, thereby escalating manufacturing costs. Recognizing the imperative to overcome these limitations, a dedicated project is underway, seeking to propel tomato fruit detection to new heights by employing sophisticated CNN techniques[5]. The objective is clear: to enhance the efficiency and feasibility of automated agricultural operations within plant factories, heralding a new era of precision and sustainability in farming practices[6].

1.2 Deep Learning

Deep learning is a type of machine learning that uses neural networks with numerous layers (deep neural networks) to learn and represent complicated patterns in data. The term "deep" refers to the depth of the neural network, which is made up of interconnected layers of nodes, each of which learns and extracts hierarchical aspects from the input data[7]. Deep learning has demonstrated exceptional performance in domains like as picture and speech recognition, natural language processing, and decision-making [8].Artificial neural networks were first introduced in the 1940s, which is when deep learning first emerged. But deep learning didn't become popular

until the 2000s and 2010s, when enormous datasets became available, computational power increased, and creative techniques were employed[9].

Artificial intelligence is dominated by deep learning, which is driving advances in a number of industries, including finance, healthcare, agriculture and autonomous systems[10]. Deep learning's continual influence on the creation of intelligent systems is a result of its ongoing investigation and improvement [11].

1.3 Convolution Neural Network

Convolutional neural networks (CNNs) are a kind of deep neural network that are particularly useful for tasks like computer vision and image recognition since they are built for processing and evaluating visual data. Convolutional layers are a tool used by CNNs to extract complex features and patterns automatically and hierarchically from input images [12].

CNN is extremely valuable since it reduces human work by automatically detecting the features. For example, for apples and mangoes, it would automatically detect the various characteristics of each class on its own[13]. CNNs are a type of Deep Neural Network that can recognize and classify specific aspects in images and are commonly used to analyze visual images. Their applications include image and video recognition, classification, medical image analysis, computer vision, and natural language processing. CNN has great accuracy, which makes it ideal for picture recognition[14]. The name "Convolution" in CNN refers to the mathematical function of convolution, which is a special type of linear operation in which two functions are multiplied to form a third function that expresses how the shape of one function is altered by the other. Simply put, two images that can be represented as matrices are multiplied to provide an output that is then used to extract features from the image [15].

CNNs originated in the 1980s and 1990s, when Yann LeCun and others made substantial contributions to the creation of convolutional neural network designs.It wasn't until the 2010s that CNNs became widely recognized and adopted. In 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created the AlexNet, a deep CNN that won the ImageNet Large Scale Visual Recognition Challenge.Subsequent advances included GoogLeNet (2014) and ResNet (2015), which added innovative architectural aspects such as inception modules and residual connections, respectively, to improve

CNN efficiency and accuracy[16]. CNNs are the core of computer vision applications, enabling advances in facial recognition, object detection, medical picture analysis, and other area [17] . The ongoing evolution of CNN architectures and techniques continues to drive progress in the field of deep learning and visual data processing.

1.4 Pre-trained Models

A pre-trained model is one that has learnt parameters (weights and biases) on a big dataset for a particular job, like natural language processing or image classification, and that model has been saved after training. Large datasets, like Wikipedia for language modeling or ImageNet for image classification, are frequently used to train these pre-trained models. Pre-trained models are frequently employed in transfer learning, which involves adapting and transferring training data from one task to another that is related but different. Researchers and practitioners can use the learned representations to improve performance on the target task by initializing a new model using pre-trained weights[18].Pre-trained models are frequently used for natural language processing tasks like BERT, GPT, and Transformer as well as picture classification tasks like VGG, ResNet, and Inception. These pre-trained models have demonstrated state-of-the-art performance on a range of tasks when appropriately altered or changed, and they have been widely adopted across different domains[19].

Pre-trained models have the main benefit of being able to extract broad features and patterns from the training set, which may subsequently be adjusted or applied to different tasks using smaller datasets. When dealing with sparse data, this method can drastically cut down on the amount of computational power and time needed to train a model from start[20].

1.5 Residual Network

A particular kind of deep neural network architecture called ResNet, or Residual Network, was created to solve the difficulties involved in training extremely deep networks. When ResNet was first presented by Kaiming He et al. in 2015, it revolutionized the deep learning space[21].The usage of residual blocks, which include skip or shortcut connections, is the primary innovation in ResNet. By allowing the network to bypass one or more layers during training, these links help gradients move more easily throughout the network. This aids in reducing the vanishing gradient issue,

which can impede training in very deep networks [22].ResNet-50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer). 10]ResNet-50v2 is a variation of the ResNet (Residual Network) architecture, which is a deep convolutional neural network (CNN). ResNet was developed to solve the problem of training very deep neural networks by leveraging residual or skip connections[23]. The "50" in ResNet-50v2 stands for the network's depth, indicating that it includes 50 layers. The "v2" indicates that it is an upgraded or better version that incorporates improvements over the ResNet-50 architecture in its initial form. Similar to its forerunners, ResNet-50v2 is extensively employed for a range of computer vision applications, such as feature extraction, object identification, and image classification [24]. The architecture is well-liked in the deep learning community because of how well it handles deep networks and how well it can capture intricate hierarchical characteristics. As we discussed in the context of changing ResNet-50V2 for tomato detection in a prior session, it is normal practice when working with ResNet-50v2 or any other deep neural network to fine-tune or modify the architecture based on the specific requirements of the task at hand [25].

1.5.1 Frozen Layer

A frozen layer in neural networks refers to a layer whose weights and biases are not updated during training. This means that the frozen layer's parameters remain constant throughout training, effectively "freezing" them [26]. The notion of freezing layers is widely employed in transfer learning and fine-tuning scenarios, in which pre-trained models are applied to new tasks or domains. In such circumstances, the network's early layers, which often capture low-level data such as edges or textures, may already be well-suited to the new task and can be frozen to minimize overfitting while retaining previously learned knowledge[27]. Overall, freezing layers is a useful strategy in neural network training since it enables practitioners to use pre-trained models efficiently and effectively for new tasks while avoiding wasteful retraining of previously learnt features. This technique is often used in transfer learning, where the base model(trained on some other dataset)is frozen.

1.5.2 Fatten Layer

A flatten layer is a sort of layer that is often employed in deep learning neural networks,

notably convolutional neural networks (CNNs). Its principal role is to convert multidimensional array data into one-dimensional arrays or vectors.
In convolutional neural networks, prior layers usually execute operations such as convolution and pooling, which produce multidimensional outputs or feature maps. However, when feeding data into fully connected layers (also known as dense layers) for tasks such as classification or regression, the input is typically one dimension. Here's where the flatten layer comes in[28].The flatten layer essentially reshapes the previous layers' multidimensional output into a flat vector by collapsing all dimensions save the first (batch dimension) into one. This transformation preserves the spatial relationships acquired by the previous layers while preparing the input for further processing by the fully connected layers.For example, if the output of the previous layers is a 3D tensor with dimensions (batch_size, height, width, depth), the flatten layer will convert it to a 1D tensor with dimensions (batch_size, height * width * depth)[29].In summary, the flatten layer acts as an intermediary stage in neural network architecture, turning multidimensional feature maps into a format that can be processed further by fully connected layers, allowing for tasks such as classification and regression.

1.5.3 Dense Layer

A dense layer, also known as a fully linked layer, is a crucial component of artificial neural networks. It is made up of numerous neurons or units organized in a single layer, with each neuron linked to every neuron in the previous layer[30]. In a dense layer, each neuron receives input from all neurons in the previous layer and generates an output that contributes to the inputs of neurons in the next layer.

1.5.4 Rectified Linear Unit

The ReLU (Rectified Linear Unit) activation function is a common nonlinear activation function in neural networks. It adds non-linearity to the model by setting all negative values in the input to zero while keeping positive values constant[31].

Graphically, the ReLU function resembles a linear function with the negative portion of the input values clipped to zero. It is computationally efficient and helps to solve the vanishing gradient problem, making it a popular choice for deep learning models. ReLU activation is especially useful in convolutional neural networks (CNNs) and deep neural networks (DNNs), where it improves the learning of complicated representations

and allows for faster convergence during training than classic activation functions such as sigmoid or tanh[32].

1.5.5 Dropout Layer

A dropout layer is a regularization technique used in neural networks to avoid overfitting. Overfitting happens when a model learns to memorize training data too well, resulting in noise or irrelevant patterns that do not adapt well to new data. Overall, dropout layers are an effective strategy for boosting neural networks' generalization performance by decreasing overfitting and encouraging the development of more robust features [33].They are widely employed in a variety of neural network topologies, such as fully connected networks, convolutional neural networks (CNNs), and recurrent neural networks.

1.5.6 Output Layer

The output layer compares the network's predictions to the ground truth labels and calculates the loss function to determine the difference between expected and actual values [34].This loss is then utilized to adjust the network's parameters, such as weights and biases, via backpropagation and gradient descent optimization. For multiclass classification problems, neural networks typically use an output layer with a softmax activation function. SoftMax turns the raw output scores of each neuron in the output layer into probabilities that accumulate to one, allowing the network's predictions to be interpreted as class probabilities [35]. The output layer with SoftMax activation allows neural networks to yield class probabilities for multiclass classification problems, making it easier to evaluate the model's results.

1.5.7 Fine Tuning Model

Fine-tuning a model is the act of modifying a pre-trained neural network's parameters (often the weights of specific layers) to adapt it to a new job or dataset [36]. This technique is especially beneficial when you have access to a pre-trained model that has already been trained on a big dataset (often for a separate but related task) and want to apply its learned representations to a new task.Fine-tuning is an effective strategy that can save time and computational resources as compared to training a model from start, particularly when working with limited data or computer resources[37]. It enables you

to tap into the information embedded in pre-trained models and tailor it to your specific requirements, resulting in models that perform well on new tasks with less training data [38].

1.6 Research Background

An investigation into the classification of ripeness and quality based on tomato fruit condition images has recently been undertaken. However, a significant constraint arises when poor-quality images confine the capture of tomatoes to a singular angle, thereby restricting the comprehensive evaluation of overall fruit condition. The efficacy of such assessments is contingent upon the scale and diversity of the dataset, and a dataset lacking in these aspects may struggle to encompass the broad spectrum of fruit variations. Consequently, limited accuracy and generalizability in the assessment process may ensue, with challenges potentially encountered in recognizing rare or uncommon fruit types, introducing errors in classification. To mitigate these issues, it becomes imperative to curate a comprehensive and diverse dataset, enriched with a sufficient scale of data, to ensure the precision and reliability of fruit assessment. The task of extracting pixel-level object instance information from images becomes more challenging when dealing with complex fruit shapes and overlapping instances. This complexity can result in difficulties in accurately segmenting and identifying individual fruits, leading to potential errors in the analysis and classification of fruit types. This complication hinders the precise determination of individual fruit characteristics within a cluster, adding an additional layer of complexity to the tomato detection process. Addressing these challenges in dataset diversity and object occlusion is pivotal to advancing the accuracy and effectiveness of tomato ripeness and quality classification, ultimately contributing to the optimization of agricultural practices in tomato cultivation.

1.7 Problem Statement

Tomatoes highlight the various applications of deep learning in agriculture, as well as their culinary relevance. These algorithms aid in tasks such as plant health monitoring, growth optimization, and yield prediction. Deep learning improves crop management practices by analyzing large volumes of data, such as soil moisture levels and weather patterns, resulting in higher yield and resource efficiency. Tomatoes are also a common

element in many different cuisines around the world, valued for their variety and nutritional significance. Thus, their dual role emphasizes the junction of technology and tradition, demonstrating how advances like as deep learning can transform agriculture while still satisfying gastronomic preferences.Temperature, humidity, sun exposure, and soil quality are just a few of the variables that affect tomato growth, ripening, and quality. Analyzing these parameters traditionally entails labor-intensive, error-prone human observation and data collection. The advancement of deep learning algorithms has allowed researchers to create methods for the analysis and identification of tomatoes that are more effective and accurate.

1.8 Motivation

One of the significant benefits of fruit detection research is that it can help automate the harvesting process. With automated fruit detection systems, farmers can significantly lower their labor costs and increase productivity. Farmers can enhance harvesting efficiency by employing deep learning algorithms to pinpoint ripe fruits in fields instead of manual scanning. These algorithms, trained on extensive datasets, automate fruit detection, saving time and labor. Such technology ensures timely harvesting, minimizing waste and maximizing yield. By integrating these algorithms, farming becomes more sustainable, productive, and profitable, benefiting farmers across scales while advancing agricultural practices towards greater efficiency and innovation. Deep learning technology has allowed researchers to pinpoint areas of fruit that are not growing properly, which may help farmers increase crop yield and minimize wastage. One of the significant benefits of fruit detection research is that it can help automate the harvesting process. With automated fruit detection systems, farmers can significantly lower their labor costs and increase productivity. Instead of manually scanning the fields for ripe fruits, farmers can rely on deep learning algorithms to identify the exact location of the fruits, making the harvesting process faster and more efficient. Deep learning technology has allowed researchers to pinpoint areas of fruit that are not growing properly, which may help farmers increase crop yield and minimize wastage.

1.9 Objective of Research

This research objectives to:

1. Generate a dataset that incorporates publicly accessible datasets .

2. Modify a CNN model (ResNet50V2) to ascertain the quality of tomatoes.

1.10 Organization of the Thesis

The subsequent sections of this thesis delineate a structured progression of the inquiry into tomato fruit quality identification. Section 2, titled the Literature Review, provides a comprehensive overview of previous research on fruit quality identification, focusing on image analysis techniques. Moving forward, Section 3 delves into the datasets utilized, the features extracted, and the architectural framework implemented in the study. The outcomes of our experimentation, organized into three distinct facets of the case study, are consolidated and expounded upon in Section 4. Conclusively, Section 5 encapsulates the culmination of this study, providing a thorough examination of its findings, future avenues for research, and the inherent contributions to the broader domain of tomato fruit quality identification.

CHAPTER 2

Literature Review

2.1 Introduction

There are a ton of published research that look into various techniques and algorithms for tomato detection.This study's literatue review section provided an overview of the body of information while highlighting the various methods and approaches used in to mato detection.It highlighted any flaws or gaps in the current research and suggests su bjects for more investigation.

2.2 Scope of Research

There has been a lot of interest in the analysis of fruit and vegetable detection content in recent years. One of the most important sources of information, besides visual inspection, is the use of machine learning algorithms to analyze and interpret data from various sensors. These sensors can provide valuable information about the fruit's size, color, texture, and even its internal quality parameters, such as sugar content or firmness. Researchers have made it simple for farmers to fully comprehend the condition of their crop and make well-informed decisions about harvesting, sorting, and post-harvest treatments by merging these sensor data with the CNN model's analysis of tomato fruit images. The majority of the studies worked on short-scale images, but we created a dataset by collecting images from Google Images and existing datasets. This dataset allowed us to train our model on a larger and more diverse set of tomato fruit images. To improve accuracy, we modified the ResNet50V2 model.

2.3 Literature Review

Many researchers have contributed to the development of a system that can detect Fruit Detection from images. The following is a content analysis of this research study. An essential component of post-harvest procedures in agriculture is the evaluation of fruit quality. Conventional techniques, which mostly depend on manual examination, are laborious and subjective, which causes inefficiencies and irregularities in the assessment process. Research into automated solutions has been prompted by the need for a quick and accurate way to analyze the quality of fruit, with an emphasis on the

utilizing cutting-edge technology like deep learning[39].In the historical evolution of fruit detection methodologies, early endeavors were grounded in classical machine vision techniques, relying on hand-crafted features for the detection and classification of fruits. Various approaches, encompassing SVM binary classifiers, pixel-level segmentation based on color spaces, and blob-level processing, were explored for the identification of tomatoes[40]. However, these methodologies presented inherent challenges, such as susceptibility to false positives, imprecise object masks, and limitations in accurately pinpointing individual fruit centroids. While recent advances have marked a transformative shift in this landscape, the quest for precision and sustainability in agricultural practices remains an ongoing pursuit [41]. The research focused more on the fruit's look.

Researchers developed a fuzzy logic method for maturity grading by considering the color, size, and form of the tomato fruit [42]. Discussed a cost-effective maturity grading system for tomatoes that uses image processing algorithms to identify six key: green, breakers, turning, pink, light red, and red stages of ripening[43]. Regarding maturity grade detection, a 98% overall accuracy rate was attained. The paradigm shift in fruit detection has been notably catalyzed by the integration of deep learning technologies, particularly convolutional neural networks (CNN). Diverse CNN architectures, including AlexNet, GoogLeNet, VGGNet, and ResNet, have transcended various domains, with researchers leveraging these frameworks within plant pathology for tasks ranging from creating candidate structures to achieving precise bounding box positioning through regression [44].To find flaws in photos of 1200 tomatoes fruit, three deep learning models—VGG16, InceptionV3, and ResNet50—were employed. Performing better than the others, VGG16 achieved an accuracy of 95.75%–98.75%.Precision agriculture has emerged as a critical approach for enhancing crop yield and resource efficiency [45]. In this study, we investigate the application of Convolutional Neural Networks (CNNs) for plant image recognition and classification in the context of precision agriculture. The study utilizes a dataset comprising high-resolution images of various plant species commonly found in agricultural settings[46]. Preprocessing techniques are employed to standardize image quality and remove noise, followed by feature extraction using CNNs. The CNN model architecture is optimized to effectively capture spatial hierarchies and patterns in plant images, facilitating accurate classification of different plant species and conditions. We conduct

comprehensive experimentation to evaluate the performance of the CNN model in terms of accuracy, precision, recall, and F1-score. Additionally, we explore the robustness of the model across different environmental conditions, such as variations in lighting, weather, and plant growth stages [47].

Within the domain of fruit detection, object detection, a pivotal facet of deep learning, has garnered attention for its informativeness, albeit with an inherent dependency on complex training data. Noteworthy approaches in this context involve region-based convolutional neural networks (R-CNNs), such as Fast-RCNN and Faster-RCNN, utilizing the selective search method for region proposal detection. Conversely, the You Only Look Once (YOLO) detector partitions the image into regions, predicting bounding boxes and probabilities for each region, offering speed advantages over traditional methods[48]. However, these techniques often encounter challenges in conveying pixel-level object instance information, particularly in scenarios involving overlapping or occluding objects. As the trajectory of fruit detection advances, striking a balance between speed and precision remains a critical pursuit in optimizing agricultural practice [49].

Crop diseases have a significant impact on production, hence their detection and identification are critical. Deep learning and intelligent firming can be used to automatically identify damaged crops. As part of this research plan, we present highly efficient convolution neural network (CNN) architectures for detecting leaf illnesses. This project's training and testing steps need the creation of a potato leaf database. CNN was used to extract the illness's features from the input photos of the given training dataset, allowing the sickness to be categorized. 1700 images of potato leaves were used to train the model, followed by 600 photographs for testing. Citrus ailments were identified using Convolutional Neural Networks, Deep Learning, base learning, and transfer learning [50]. The suggested architecture beats other existing ResNet models in terms of accuracy, obtaining a score of 99.62% based on training, testing, and experimental results.Current gene regulatory network (GRN) inference approaches are renowned for concealing a large number of indirect interactions inside predictions. Filtering out indirect connections from direct ones continues to be a significant difficulty in GRN reconstruction. To overcome this issue, we devised a redundancy silencing and network enhancement technique (RSNET) to infer GRNs. the redundant interactions including weak and indirect connections are silenced by recursive

optimization adaptively, and the highly dependent nodes are constrained in the model to keep the real interactions. This study provides a useful tool for inferring clean networks[51].The most crucial crop for both socioeconomic stability and global food security is rice. A portion of the global population considers rice to be a staple food, however the issue is that all types of rice are plagued by various illnesses and pests. As a result, identifying and treating rice plant illnesses is essential to guaranteeing the quality of healthy and appropriate rice growth. In order to get the best accuracy, the Convolutional Neural Network (CNN) algorithm was used in this study to classify diseases on the leaves of rice plants [52]. Several parameters and architecture were tested. This study was conducted image classification of rice plant disease using CNN architecture ResNet-50V2 with data using preprocessing Augmentation. The test was conducted with three optimizers such as SGD, Adam, and RMSprop by combining various parameters, namely epoch, batch size, learning rate, and SGD and RMSprop optimizers. Division of image data with 70:30 ratio of training data and test data; 80:20; 90:10. From these results, it was found that Adam was the best optimizer in the 80:20 data division in this study with an accuracy level of 0.9992, followed by the SGD optimizer with an accuracy level of 0.9983, while the RMSProp optimizer was ranked third with an accuracy level of 0.9978 [53].There are two fully linked layers and four convolution layers in the FASNet model. Our self-development FASNet CNN model made use of the pre-trained deep learning models. To reduce the number of parameters in the model and the channel depth, the first convolutional layer uses a 1x1 kernel size on each pixel as a fully linked connection. The purpose of the dropout layer and early-stopping class is to restrict the amount of neural connections and avoid overfitting. An open-source collection of 6,432 training and testing photos served as the source of the dataset for this investigation. As a consequence, our method identified healthy people, those with pneumonia, and COVID-19 infected people with 98.48% accuracy. We anticipate that the FASNet model can be utilized in future development research to aid with COVID-19 diagnosis, based on these encouraging preliminary results. When compared to other well-known models like ResNet50V2 and MobileNetV2, the FASNet model's output shows a strong correlation[54].In this research, two simple yet very effective methods for facial attribute estimation in various image circumstances are presented. The suggested methods make use of a quick and simple face alignment process for preprocessing, after which they employ the lightweight Convolutional Neural Network (CNN) architectures MobileNetV2 and Nasnet-Mobile to predict facial

features. When it comes to accuracy and processing speed, both models perform similarly.Our suggested technique outperforms the top state-of-the-art model in processing speed and outperforms the fastest existing model in accuracy, according to a rigorous comparative evaluation against state-of-the-art methodologies concerning both processing time and accuracy. Moreover, our approach is intuitive and well-suited for mobile device implementation. To summarize, this study introduces two novel techniques for estimating facial attributes in images, showcasing their simplicity, effectiveness, and efficiency. By leveraging fast face alignment techniques and lightweight CNN architectures, we achieve competitive performance in accuracy and speed. Our method outperforms existing models in both metrics, making it particularly suitable for real-time applications and mobile deployment[55].

To improve the precision of domestic industrial defect detection, this research suggests a Two-Stage Industrial Defect Detection Framework. The Optimized-Inception-ResnetV2 and Improved-YOLOv5 models, each of which serves a distinct purpose of defect classification and location, form the foundation of the system [56]. There are improvements made to YOLOv5's multiscale detection layer, feature scales, and backbone network in order to increase the first-stage recognition's efficacy in detecting small flaws with high similarity on steel surfaces. Furthermore, the convolutional block attention module (CBAM) is integrated into the Inception-ResnetV2 model in the second stage of recognition, which is succeeded by network architecture and loss function optimizations. Several datasets, such as Pascal VOC2007, NEU-DET, and Enriched-NEU-DET, are used in comparative tests, which show notable gains in testing outcomes. With an AUBO-i5 robot with an Intel RealSense D435 camera, the two-stage framework achieves a mean average precision (mAP) of 91.0% in real industrial situations and 83.3% on the Enriched-NEU-DET dataset. This demonstrates the framework's superiority and versatility in the identification of industrial defects[57].Computer vision-based scene classification technology is widely used in many different fields. Nevertheless, many current computers vision classification models find it difficult to keep up with the demands of modern scene categorization jobs as images get more complex. These responsibilities require considering several interrelationships inside the image in addition to objects, backdrops, and spatial layouts [58]. In order to handle complicated scene classification difficulties, this study proposes an approach that improves dataset processing by analyzing existing scene classification

algorithms in conjunction with the Xception model. A serialized image improvement technique is used to increase the size of the dataset and magnify picture features by using image enhancement technologies.By utilizing the Xception model, the method collects information from images and produces a more robust scene categorization model. Results from experiments show how well the Xception model performs scene categorization tasks, making up for the shortcomings of classic Convolutional Neural Networks(CNN) models in terms of feature extraction and generalization capacity [59]. This method presents a pre-trained Convolutional Neural Network (CNN) model specifically intended to identify benign or malignant pre-segmented breast cancer masses in mammography pictures. Through meticulous investigation and analysis, the system employs various methods to address the challenge of limited training data. These methods include data augmentation, which artificially expands the dataset by generating modified versions of existing data points. Targeted pre-processing techniques are also utilized to enhance the quality and relevance of the input data [60]. Additionally, transfer learning is leveraged to transfer knowledge from a pre-trained model to the current task, enabling the network to benefit from features learned on a larger dataset. By integrating these approaches, the system effectively mitigates the limitations posed by small training datasets, improving the model's robustness and performance [61]. The system is based on a modified DENSENET201 architecture that has undergone extensive training and testing to overcome categorization difficulties. Data from the RGB color model, which includes 2480 benign and 5429 malignant instances, is used to train the CNN model. The findings show a 97% accuracy rate, with 99% precision rates for benign cases and 83% precision rates for malignant cases. For benign instances, the recall rates are 83%, while for malignant situations, they are 99%. In general, the accuracy of the provided DENSENET201 model is better than that of earlier approaches for this system[62].In comparison to earlier models, this research presents EfficientNetV2, a unique family of convolutional networks that is intended to increase training time and parameter efficiency. These models are created by combining scaling and training-aware neural architecture search to concurrently maximize parameter efficiency and training speed. New techniques like Fused-MBConv broaden the search space for these models [63]. Results from experiments show that EfficientNetV2 models can be up to 6.8 times smaller and train substantially faster than state-of-the-art models. An improved progressive learning strategy that dynamically modifies regularization methods like data augmentation in tandem with picture size is

suggested as a solution to this accuracy decline. EfficientNetV2 models perform better with progressive learning than earlier models on datasets including ImageNet, CIFAR, Cars, and Flowers [64]. EfficientNetV2 outperforms more contemporary models like ViT by 2.0% accuracy on the ImageNet ILSVRC2012 dataset after pretraining on the ImageNet21k dataset. This is achieved while training 5 to 11 times quicker with the same computational power [65].The purpose of this study is to introduce a deep learning approach to traffic sign recognition, with a focus on identification and classification. Using the ResNet-50, VGG16, RegNetX002, and DenseNet121 models, the study uses transfer learning. Transfer learning is the process of extracting characteristics from images using pre-trained models and then training a new model for detection. Each transfer learning model's accuracy rates are calculated using the Traffic Signs dataset, which contains 162 more special classes in addition to the GTSRB dataset [65].Diagnosing defects in electric motors is critical for the industrial sector because of the potential losses caused by equipment downtime. This study addresses the issues associated with the quality and amount of motor data by offering an intelligent failure detection model based on Deep Transfer Learning (DTL) and InfraRed Thermal (IRT) pictures. A In addition to the modified ResNet architecture, a cropping layer is incorporated into the network to identify regions of interest [67]. This layer facilitates focusing on specific areas within the input data. Subsequently, hyperparameter optimization is conducted using Random Search (RS). RS systematically explores a predefined hyperparameter space to identify the combination that yields optimal performance [68]. By integrating these components, the network becomes adept at identifying and processing relevant regions within the input, while the hyperparameter optimization ensures that the model's parameters are fine-tuned for improved performance on the given task. cropping layer is also added to the network to designate regions of interest, followed by hyperparameter optimization via Random Search (RS) . The results show that the RegNetX002 model performs well in fault classification, with an accuracy of 98.18% in 3245 seconds. Overall, tests applying deep transfer learning demonstrate significant potential for adapting to machinery failure diagnosis, particularly when leveraging Infrared Thermal data [69] .

Action recognition plays a pivotal role in computer vision, yet current models come with significant computational overhead, limiting their deployment on mobile devices for real-world applications. of real-time action recognition, distinct from traditional

inference settings. Results show that models, which have a 6x speed gain while preserving comparable accuracy to cutting-edge techniques, can efficiently meet real-time requirements on mobile devices. Notably, this research is the first attempt to implement current deep learning action recognition models on mobile devices, paving the path for widespread use in real-world applications [70].

TABLE 2.1 KEY FINDINGS OF THE PREVIOUS LITERATURE

Author and year	**Title**	**Methods/Algorithms**	**Findings**
Ibrahim, Nehad M., Dalia Goda Ibrahim Gabr, Atta-ur Rahman, Sujata Dash, and Anand Nayyar [71]. 81, no. 19 (2022): pp.27783-27798.	A deep learning approach to intelligent fruit identification and family classification	CNN	This study developed a deep learning model using fruit photos from 52 species across four families for fruit identification. The model achieved a 93% prediction success rate and 99.82% accuracy for testing and training.
Ünal, Haci Bayram, Ebru Vural, Burcu Kir Savaş, and Yaşar Becerikli [72]. pp. 1-5, IEEE ,2020.	Fruit recognition and classification with deep learning support on embedded system (fruitnet).	ConNN, Image Processing Method	The proposed study uses image processing methods to classify fruits, reducing time, cost, and labor losses. A Convolutional Neural Networks (nNN) deep learning model is developed on the Keras platform. The model is tested on 20 different fruits in two data sets, and finally on a Jetson Nano card in real time
Gill, Harmandeep Singh, Osamah Ibrahim Khalaf, Youseef Alotaibi, Saleh Alghamdi, and Fawaz Alassery [73]. 33, no. 1 2022	Multi-Model CNN-RNN-LSTM Based Fruit Recognition and Classification	CNN, RNN, LSTM	The deep learning algorithm ensemble for fruit categorization is proposed in this research. The suggested strategy performs better than the current methods in terms of accuracy analysis and F-measure, according to experiments conducted on ten photos of fruit.

Septiarini, Anindita, Hamdani Hamdani, Sri Ulan Sari, Heliza Rahmania Hatta, Novianti Puspitasari, and Wiwien Hadikurniawati [74]. pp. 92-96. IEEE, 2022.	Image Processing Techniques For Tomato Segmentation Applying K-Means Clustering and Edge Detection Approach.	K-means clustering , HSV, Canny operator.	With an emphasis on region of interest identification, pre-processing, segmentation, and post-processing, this work presents a tomato segmentation approach for plantation fields. The performance evaluation revealed segmentation accuracy averages of 2.74%, 4.77%, and 91.43%.
Sudharshan, Duth P., and T. N. Jhansy [75]. pp. 1-6. IEEE, 2022.	Tomato Fruits Disease Detection Using Image Processing.	Enhanced SVM	In farming, computer learning techniques reduce labor costs and optimize harvest activities by improving fruit recognition and classification. Experimental data indicates that 94.036 percent of diseases can be diagnosed accurately using these techniques.
Mureşan, Horea-Bogdan [76]. pp. 103-107. IEEE, 2022.	An Automated Algorithm for Fruit Image Dataset Building.	single shot multibox detector	The paper presents an algorithm that generates annotation files for bounding boxes around fruits using pictures from the Fruits-360 collection. The method minimizes the necessity for gathering data in the actual world by considering differences in illumination and occlusion in outdoor settings. The trained model outperformed other innovative models, with a mean average accuracy of 0.750.
Legaspi, Jericho, John Raphael Pangilinan, and Noel Linsangan [77]. pp. 613-618. IEEE, 2022.	Tomato Ripeness and Size Classification Using Image Processing.	Raspberry Pi, Raspberry Camera v1.3, Ultrasonic sensor,	The study created an image processing system to categorize the size and maturity of tomatoes. The system's accuracy score for classifying ripeness was 92.86%, while its accuracy score for classifying size was 96%.

Tunio, Muhammad Hanif, Li Jianping, Muhammad Hassaan Farooq Butt, Imran Memon, and Yumna Magsi[78]. pp. 1-5. IEEE, 2022.	Fruit Detection and Segmentation UsingCustomized Deep Learning Techniques	U-Net architecture	With the contraction path encoding characteristics U-Net architecture and the expansion path decoding resolution, the approach employs segmentation to discover and locate objects. For an enhanced crop, the model projects accuracy and test image loss of 98.66% and 0.0268%, respectively.
Nagesh, A. Seetharam, and G. N. Balaji [79]. vol. 1, pp. 1-6. IEEE, 2022.	Deep Learning Approach for Recognition and Classification of Tomato Fruit Diseases.	VGG16	This research suggests a convolutional neural network-based technique for tomato disease detection in color photos. The technique demonstrates effective prediction and early detection of tomato illnesses by using an augmentation strategy to build a dataset with big samples.
Azman, Nur Fitrah, Nor Ashikin Mohamad Kamal, and Norizan Mat Diah [80]. pp. 119-124. IEEE, 2023.	Tomato Fruit Ripening Classification Using Wavelet-Based Feature Extraction and Multilayer Perceptron.	DWT, MLP	This study aimed to accurately classify a diverse range of data samples using the combination of discrete wavelet transform and multilayer perceptron classifiers. The achieved accuracy of 81% demonstrates the effectiveness of this approach in accurately categorizing the data.
Hong, Suk-Ju, Seongmin Park, Chang-Hyup Lee, Sungjay Kim, Seung-Woo Roh, Nandita Irsaulul Nurhisna, and Ghiseok Kim [81]. IEEE Access (2023).	Application of X-ray imaging and convolutional neural networks in the prediction of tomato seed viability.	CNN	Based on X-ray scans, models were constructed in this work to evaluate tomato seed viability. The models were evaluated for structural integrity once they were in the seedling stage.The CNN-based model had a greater accuracy of 86.01% in comparison to the image-processing-based model, indicating its possible application in determining the viability of tomato seeds.

Kushwaha, Arvinda [82]. pp. 1-5. IEEE, 2023.	Fruit Classification Using Optimized CNN.	CNN	The study demonstrated the effectiveness of CNN technology in fruit classification, with a TensorFlow backend model achieving 96.88% accuracy after 40 training epochs, demonstrating the practical viability of such approaches in real-world applications.
Mehta, Shiva, Vinay Kukreja, and Rishika Yadav[83]. pp. 309-314. IEEE, 2023.	A Federated Learning CNN Approach for Tomato Leaf Disease with Severity Analysis.	CNN	The study classifies and detects tomato leaf infections into five severity categories using a CNN model with federated learning. With an accuracy range of 96% to 98%, the model regularly produces excellent results. The model has the highest recall when it comes to class 5 (illness 4), indicating that agricultural settings might benefit from its use.
Saini, Archana, Kalpna Guleria, and Shagun Sharma[84] . pp. 01-06. IEEE, 2023.	Tomato Leaf Disease Classification using Convolutional Neural Network Model.	CNN	Tomato leaf diseases were identified and categorized using Convolutional Neural Networks (CNNs), a deep learning technique. The model was developed using Adam and the SGD optimizer, and the dataset was sourced from Kaggle. With a loss value of 0.0044 and an accuracy of 0.9966, the CNN model performed well.
Singh, Utpal Kant, Rajnish Kumar, Saurabh Kumar, Shibasish Kar, and Santos Kumar Baliarsingh. [85]. pp. 1-6. IEEE, 2023.	Detection of Diseases in Tomato Plants using Convolutional Neural Network.	CNN	Their study, produced an average classification accuracy of 82.4%, demonstrating how these cutting-edge approaches may perform better than conventional ones.

Roy, Kyamelia, Sheli Sinha Chaudhuri, Jaroslav Frnda, Srijita Bandopadhyay, Ishan Jyoti Ray, Soumen Banerjee, and Jan Nedoma. [86]. IEEE Access 11 (2023): 14983-15001.	Detection of tomato leaf diseases for agro-based industries using novel PCA DeepNet.	F-RCNN, GAN, PCA DeepNet	The system integrates Generative Adversarial Network (GAN) and a customized Deep Neural Network (PCA DeepNet) with Principal Component Analysis (PCA).The findings indicate an average precision of 98.55% and a classification accuracy of 99.60%.
Hsieh TH, Kiang JF [87]. Sensors 20, no. 6 (2020): 1734.	Comparison of CNN algorithms on hyperspectral image classification in agricultural lands.	CNN	The HSI data of a crop agriculture in Salinas Valley and a mixed vegetation agriculture in Indian Pines were used to compare the performance of these CNN algorithms. The highest overall accuracy on these two cases are 99.8% and 98.1%,
Yalcin H, Razavi S [88]. In 2016 Fifth International Conference on Agro-Geoinformatics 2016	Plant classification using convolutional neural networks.	CNN	Convolutional Neural Networks (CNNs) have shown remarkable success in image classification tasks, including plant species identification. In this study, we explore the effectiveness of CNNs for plant classification using a comprehensive dataset of plant images. We propose a CNN architecture tailored for plant classification, leveraging transfer learning and data augmentation techniques.
Kayabasi A, Toktas A[89]. Neural Network World 28.3 (2018)	Automatic classification of agricultural grains: Comparison of neural networks	ANN	it would be exciting if the models can accumulate knowledge to handle continual tasks. Towards this goal, we propose an ANN-based continual classification method via memory storage and retrieval, with two clear advantages: Few data and high flexibility. This

			proposed ANN-based model combines a convolutional neural network (CNN) and generative adversarial network (GAN).
Kujawa S, Niedbała G [90]. PP-497	Artificial neural networks in agriculture	ANN	Artificial neural networks are one of the most important elements of machine learning and artificial intelligence. They are inspired by the human brain structure and function as if they are based on interconnected nodes in which simple processing operations take place.
Gupta A, Nahar P [91]. PP10235-10244 Journal of Ambient Intelligence and Humanized Computing	Classification and yield prediction in smart agriculture system using IoT	ML	Machine learning (ML) methods achieve the requirement of scaling the learning performance of the model. This paper introduces a hybrid ML model with IoT for yield prediction. This work involves three phases : preprocessing, feature selection(FS) and classification. Initially, the dataset is preprocessed, and FS is done on the basis of Correlation based FS (CBFS) and the Variance Inflation Factor algorithm (VIF). Finally, a two-tier ML model is proposed for IoT based smart agriculture system.
Saad AM, Abu-Naser SS[92]. 2023	Rice Classification using ANN	ANN	Rice classification plays a vital role in ensuring food security and quality control. In this study, we propose an approach utilizing Artificial Neural Networks (ANN) for the automated classification of rice grains based on their varieties. The dataset comprises high-resolution images of different rice varieties obtained from various sources.

Panthakkan A, Anzar SM, Jamal S [93].	Concatenated Xception-ResNet50—A novel hybrid approach for accurate skin cancer prediction.	ResNet 50	The suggested approach's performance is contrasted with that of a Deep CNN and other cutting-edge transfer learning models. The performance of the recommended technique is evaluated using the Human Against Machine (HAM10000) dataset. 10,500 skin photos were used in this investigation. The sliding window method is used to test and train the model. With a 97.8% prediction accuracy, the concatenated X-R50 model that has been suggested is state-of-the-art.

Table 2.1 provides an overview of the major conclusions drawn from the prior research. There is always room for more research in every new piece. After evaluating all the information, we were able to determine the viability of our system and research.

2.4 SUMMARY

The literature study identifies a new trend in the assessment of fruit quality in agricultural contexts through the application of deep learning techniques. There is a significant gap in the application of these technologies to tomatoes, despite the fact that a great deal of study has been done on the general detection of fruit problems. By providing a focused examination of deep learning-based methods for identifying and evaluating tomato fruit condition, this paper seeks to close this gap.Tomatoes present unique challenges in terms of their diverse shapes, sizes, and susceptibility to various conditions affecting their quality. The paper underscores the need for specialized methodologies tailored to address these challenges, emphasizing the importance of a dedicated exploration into the realm of tomato fruit condition detection.

By exploring the intricacies of deep learning applications for tomatoes, the study makes a substantial literary contribution. It not only recognizes the wider interest in applying deep learning to agriculture, but it also focuses specifically on the particular field of tomato fruit quality evaluation. The objective is to improve the comprehension and usefulness of deep learning methods in handling the complexities related to tomato fruits, which are not only abundant but also multipurpose. It not only recognizes the

wider interest in applying deep learning to agriculture, but it also focuses specifically on the field of tomato fruit quality evaluation. The objective is to improve the comprehension and usefulness of deep learning methods in handling the complexities related to tomato fruits, which are not only abundant but also multipurpose. This dedicated exploration is poised to advance the field by providing insights into the nuances of tomato fruit condition detection, paving the way for more accurate and efficient methods. As agriculture increasingly adopts technological advancements, the paper seeks to bridge the gap between general fruit quality assessment and the unique characteristics of tomatoes.

CHAPTER 3

Methodology

3.1 Introduction

I provided short reports on the identification and functioning of Tomato detection at various stages of the procedure in this part.

3.2 Methodology

Recent investigations have specifically delved into the efficacy of employing deep learning methodologies for the tracking and analysis of fruit. The utilization of computer vision or deep learning in scrutinizing images or videos of fruit quality introduces a novel dimension to the assessment of quality severity. The CNN transfer learning technique facilitates the acceleration of training and the utilization of pre-trained weights. This approach optimizes the use of available data by reutilizing weights acquired from prior tasks. Moreover, the performance of CNNs can be enhanced through adjustments to the learning rate and the adoption of various optimization algorithms, such as stochastic gradient descent or momentum-based techniques. Addressing this challenge, we employ an adapted CNN transfer learning method tailored for the precise identification of tomato events in images depicting tomato quality. This method amalgamates the advantages of transfer learning with specific modifications customized for the task of tomato event identification. Not only does this modified CNN transfer learning method enhance training efficiency, but it also elevates the overall performance of tomato event classification, establishing it as a valuable tool for agricultural applications and crop management. To further refine the accuracy of the deep learning-based approach, this paper advocates for a set of strategies, including the amalgamation of data from diverse sources, the utilization of multi-resolution input, and the selection of optimal hyperparameters.

The proposed methodology focuses on employing a deep learning-based modified ResNet50V2 Convolutional Neural Network (CNN) with transfer learning to conduct comprehensive analysis of diverse imagery, indicating a robust approach for extracting meaningful insights from varied visual data sources. The approach enhances

comprehensive image analysis capabilities, showcasing versatility in understanding and interpreting visual data across various domains.

3.3 Research Activity

In this section, we detail the comprehensive process undertaken in my study, encompassing the collection and annotation of data, the architecture of our deep learning model, and the subsequent training and testing phases. The primary contribution of our work lies in the meticulous compilation of a diverse dataset featuring tomato imagery, systematically labeled with corresponding classes, fostering an extensive range of tomato conditions The dataset creates a solid basis for training and testing our suggested deep learning model by combining photos from various sources. This varied compilation adds to the overall robustness of the model by improving its performance and adaptability in various settings. Notably, our manual labeling process enhances the precision of classification. I chosen approach involves the utilization of a modified ResNet50V2 classifier, a decision substantiated by its ability to yield notably more accurate predictions. To elucidate the entire process, we provide a comprehensive workflow diagram (Figure 3.1) and a Flow chart (Figure 3.2) of our model that offers both a detailed explanation and a visual representation of my tomato detection system.

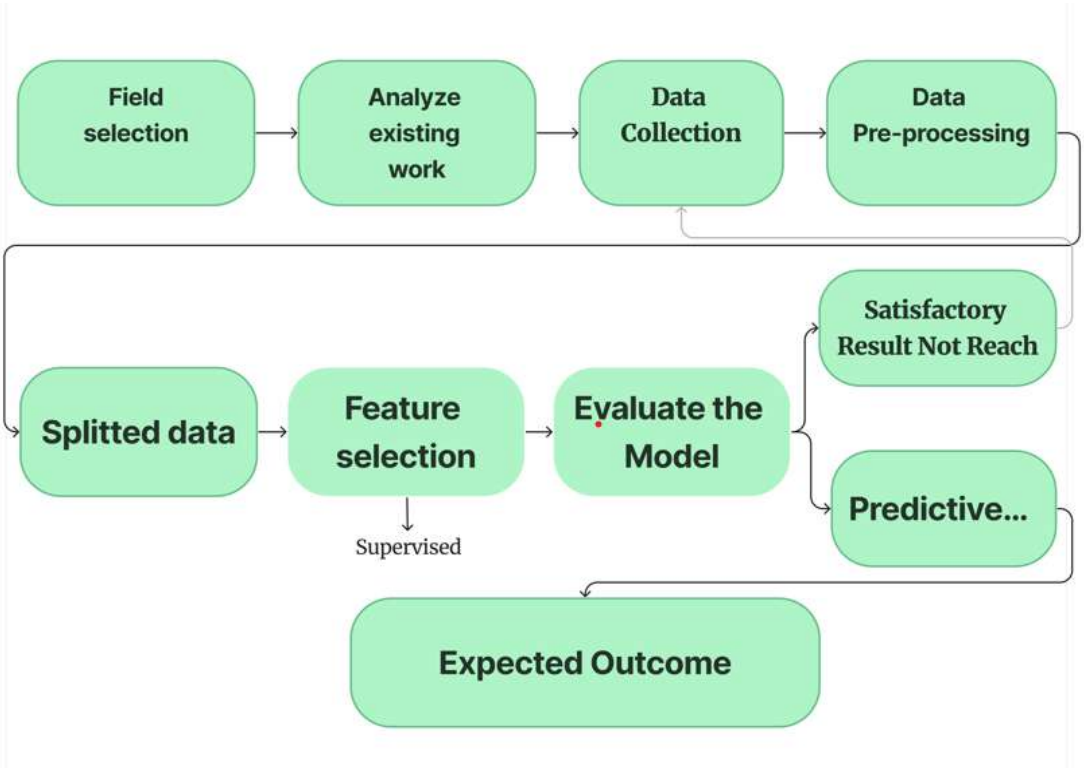


Figure 3.1:Workflow Diagram of our Research

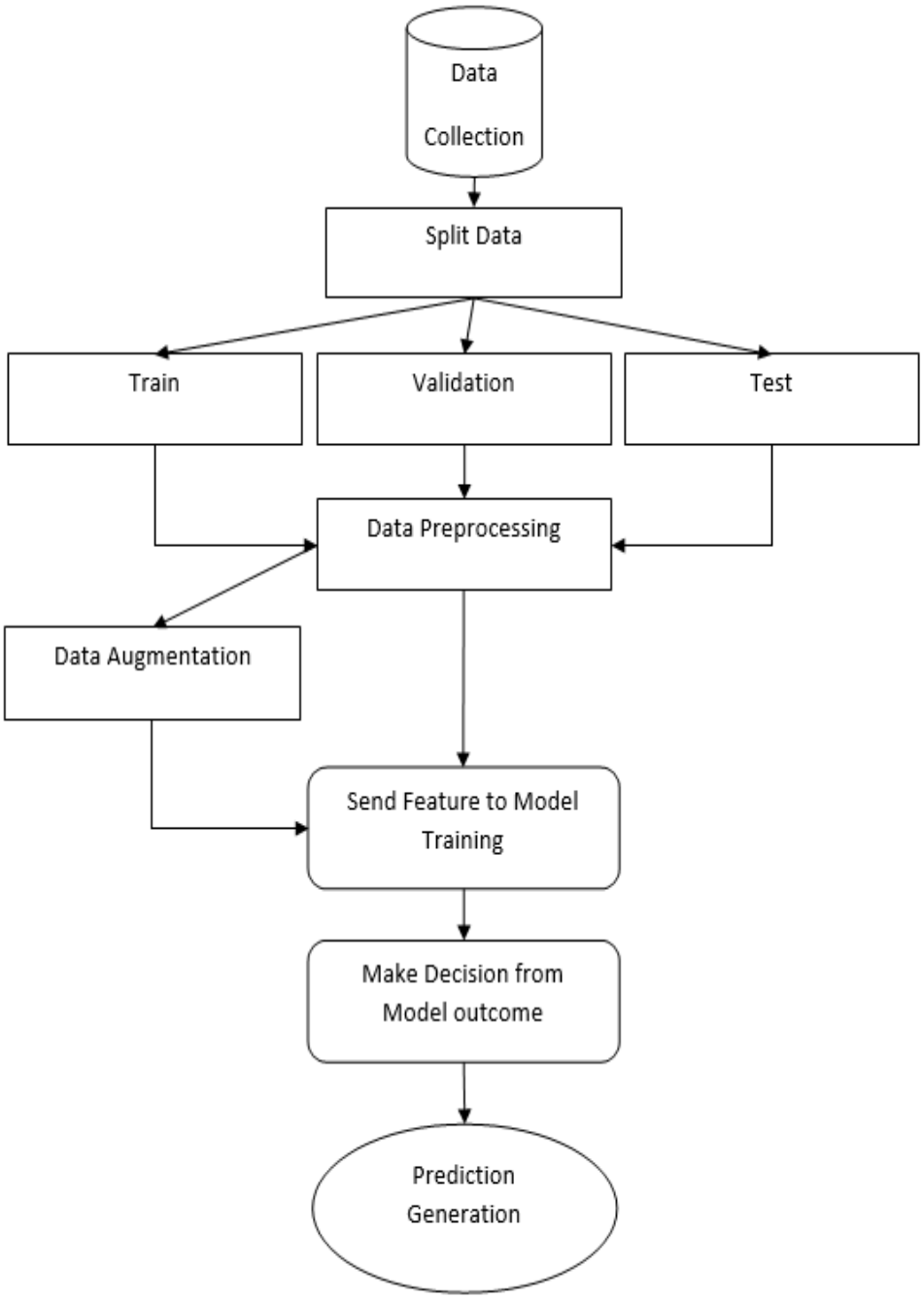


Figure 3.2: Flow chart of proposed CNN model

3.3.1 Dataset generation

To train and assess the efficacy of our tomato condition detection model, I curated a comprehensive dataset sourced from multiple channels to ensure diversity and inclusivity. This dataset comprises a total of 3345 images gathered from reputable

platforms such as Kaggle, Mendeley Data[94], and other organizations specializing in agricultural imagery. By amalgamating datasets from various sources, my goal was to encompass a broad spectrum of tomato conditions, thereby facilitating a robust training process. The largest contribution to our dataset comes from a collection of 2036 images sourced from Kaggle. To maintain a balanced representation of tomato classifications and avoid redundancy[95], some images were selectively omitted. The labeling process involved categorizing tomatoes into four distinct classes, representing conditions such as damages, maturity levels, ripeness, and unripens. This meticulous labeling ensures that the model can discern nuanced differences in tomato conditions accurately. Additionally, Figure 3.3 provides a visual representation of the labeled images within the Tomatoes Dataset, aiding in the comprehension of my dataset's composition and organization.

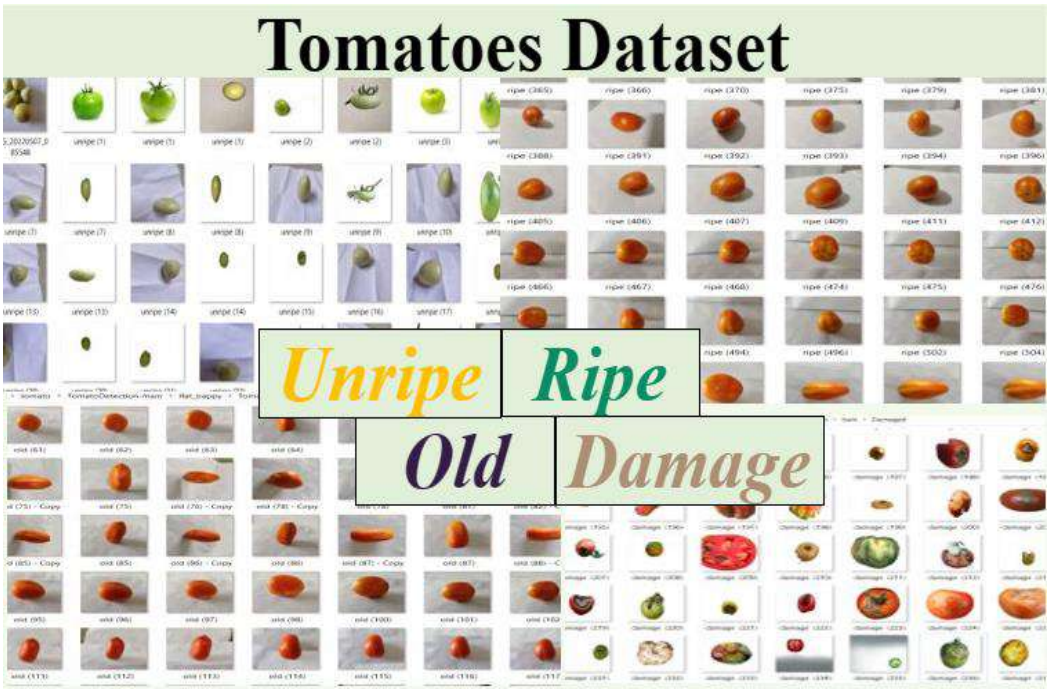


Figure 3.3: Example of Tomatoes Dataset.

3.3.2 Data Pre-processing

To evaluate the performance of our model,I utilized a dataset partitioned into training (comprising over 62.3%), validation (20.6%), and testing (less than 17.1%) sets. Prior to training, preprocessing steps were implemented, which included resizing the images to dimensions of (224, 224) and normalizing the RGB channels using mean and standard deviations obtained from the ImageNet dataset. This normalization technique

transformed pixel values from the original range of 0 to 255 to a normalized range of 0 to 1, ensuring uniformity in the input data and promoting convergence during the training phase. Additionally, the use of the ImageNet dataset for normalization serves to enhance the model's generalization capabilities by aligning the data distribution with a widely used benchmark dataset. Facilitating convergence during the training process.

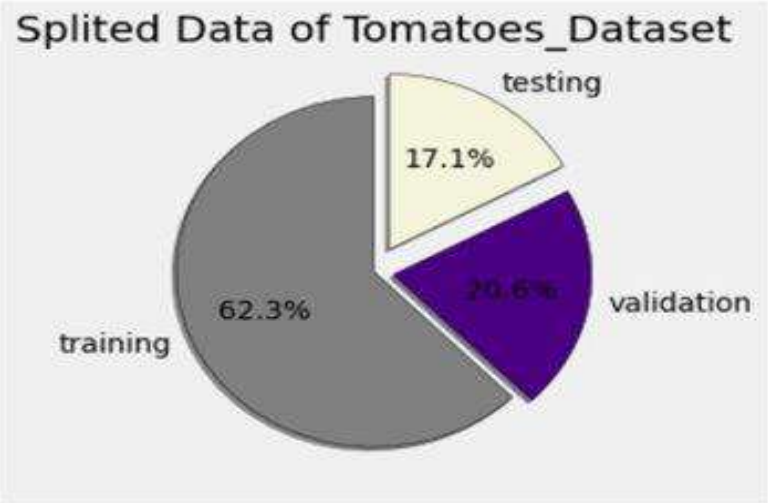


Figure 3.4: Split Data of Tomatoes Dataset

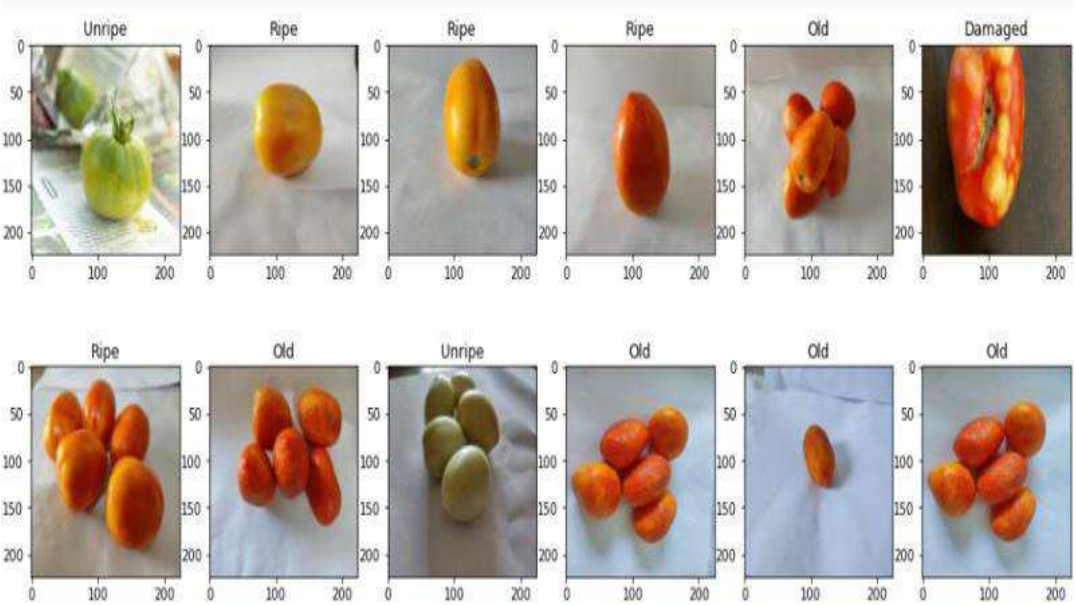


Figure 3.5: Sample Images in Tomatoes Dataset

3.3.3 Data Augmentation

In addition to utilizing various picture augmentation methods like random flipping,

contrasting, and rotating, we implemented strategies to ensure a well-balanced dataset, crucial for robust model training. These techniques not only diversified the dataset but also promoted resilience in the face of varied real-world scenarios. Following the augmentation process, I employed a modified convolutional neural network (CNN) to classify the images. This holistic approach aimed to bolster the model's ability to generalize and recognize patterns effectively, thereby improving its overall efficacy in object categorization tasks. Moreover, by integrating augmentation techniques into the training pipeline, I aimed to mitigate overfitting and enhance the model's adaptability to unseen data, ultimately fostering greater performance and reliability in practical applications. Here are some augmented images showing below:

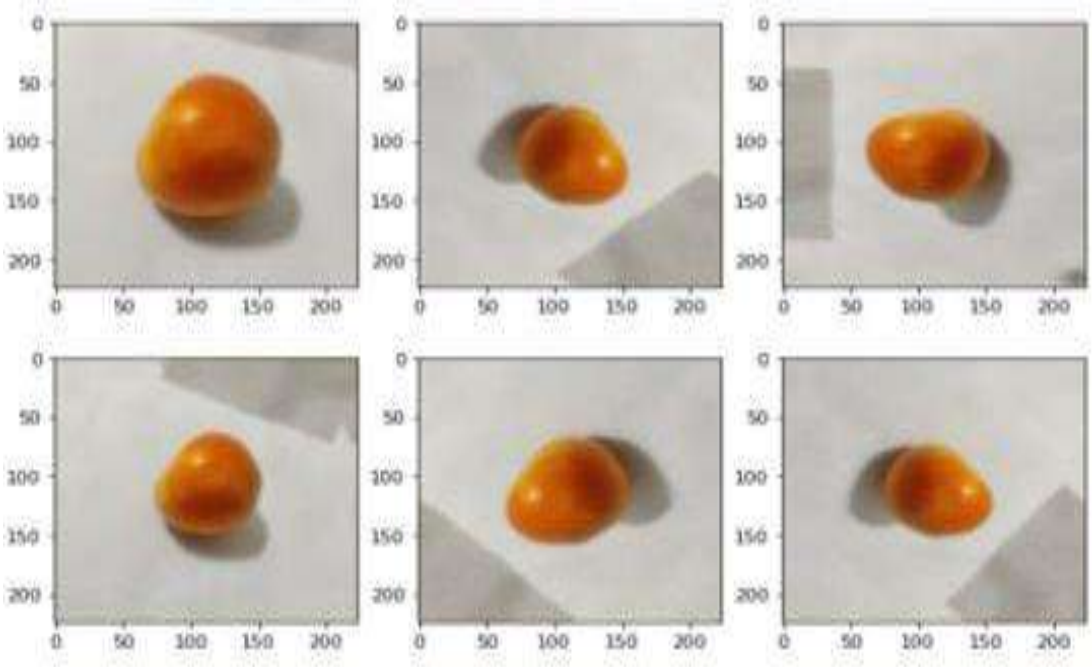


Figure 3.6: Augmented Image in Tomatoes Dataset

3.3.4 Modified ResNet Architecture

These modifications might include variations in the number of layers, introduction of new layers or modules, changes in activation functions, or the addition of attention mechanisms, among others.

3.3.5 ResNet Architecture Overview

The ResNet architecture is renowned for its distinctive structure, featuring multiple residual stages, each composed of several residual blocks. Within these blocks, two

convolutional layers are stacked, each followed by ReLU activation functions and batch normalization layers. However, what truly sets ResNet apart is its ingenious solution to the vanishing gradient problem. By incorporating identity shortcut connections, also known as skip connections, ResNet facilitates the smooth flow of gradients during training. These shortcut links enable the network to bypass one or more layers, allowing gradients to propagate more effectively through the network. This innovative design not only addresses the issue of vanishing gradients but also contributes to the model's ability to train deeper architectures without degradation in performance.

3.3.6 Modification for Improved Performance

In my quest to refine the original ResNet architecture, my primary focus was on fortifying its capabilities through the strategic introduction of skip connections. This augmentation entails integrating shortcut links within the architecture, which play a pivotal role in smoothing the flow of gradients during the training process. By enhancing gradient propagation, this modification fundamentally contributes to an overarching enhancement in performance and training efficiency, particularly within the ResNet50V2 framework. The incorporation of skip connections addresses a fundamental challenge in deep neural network training: the vanishing gradient problem. By enabling the direct flow of gradients through the network, skip connections effectively mitigate the issue of diminishing gradient magnitudes, thereby fostering more effective and stable training dynamics. This not only expedites convergence during training but also enhances the model's ability to capture intricate patterns and nuances within the data. Moreover, the introduction of skip connections serves to augment the model's depth without exacerbating the optimization difficulties typically associated with deeper networks. This enables ResNet50V2 to leverage its increased capacity for feature representation, leading to superior performance in tasks such as image classification, object detection, and semantic segmentation.

Furthermore, the strategic incorporation of skip connections aligns with broader trends in deep learning research, which increasingly emphasize the importance of architectural design in optimizing model performance. By carefully engineering the connectivity patterns within the network, we aim to harness the full potential of ResNet50V2, unlocking new levels of performance and efficiency in complex computational tasks. This approach reflects a shift towards more thoughtful and deliberate architectural

choices, recognizing that model design plays a crucial role in determining overall effectiveness. The deliberate integration of skip connections represents a pivotal advancement in refining the ResNet architecture, marking a significant stride towards achieving state-of-the-art performance in deep learning-based image analysis and classification tasks. This architectural enhancement underscores our commitment to pushing the boundaries of model efficacy and advancing the field of computer vision. As we continue to explore novel architectural innovations and optimization techniques, we are poised to further enhance the capabilities of ResNet and contribute to the ongoing evolution of deep learning methodologies.

In the development of my customized ResNet50V2 variant, I implemented specific alterations to the architecture to tailor it to our specific task. Firstly, I removed the top layer and replaced it with a sequential model comprising three new layers. This restructuring allowed us to take a more focused approach to feature extraction, ensuring that the model learns relevant patterns from the input data more effectively. Subsequently, we introduced a dense layer equipped with 800 neurons, accompanied by a ReLU activation layer. This adjustment was carefully designed to enhance the model's capability to extract meaningful features from the input data, enabling it to capture and represent essential information more accurately. To mitigate the risk of overfitting and ensure the model's generalization ability, I incorporated a dropout layer. This regularization technique helps prevent the model from memorizing noise in the training data, promoting better performance on unseen instances during inference.

The final layer in my modified architecture is an output layer featuring a softmax activation function. This crucial component plays a pivotal role in generating probability distributions over the target classes, enabling the model to make informed predictions with confidence. Furthermore, to provide a comprehensive understanding of these architectural modifications and their impact on model performance, we include a detailed illustration in the form of a figure. This visual aid serves to elucidate the intricacies of my customized ResNet50V2 architecture, allowing for easier comprehension and assessment of the model's design and functionality. The following Figure 3.7 depicts the architecture of the Frozen Layer of ResNet50V2, highlighting the structural changes implemented to optimize the model for my specific task. These changes are tailored to optimize the model's functionality and efficiency in addressing

the specified objective, indicating a targeted approach to adapting the ResNet50V2 architecture for improved task-specific performance.

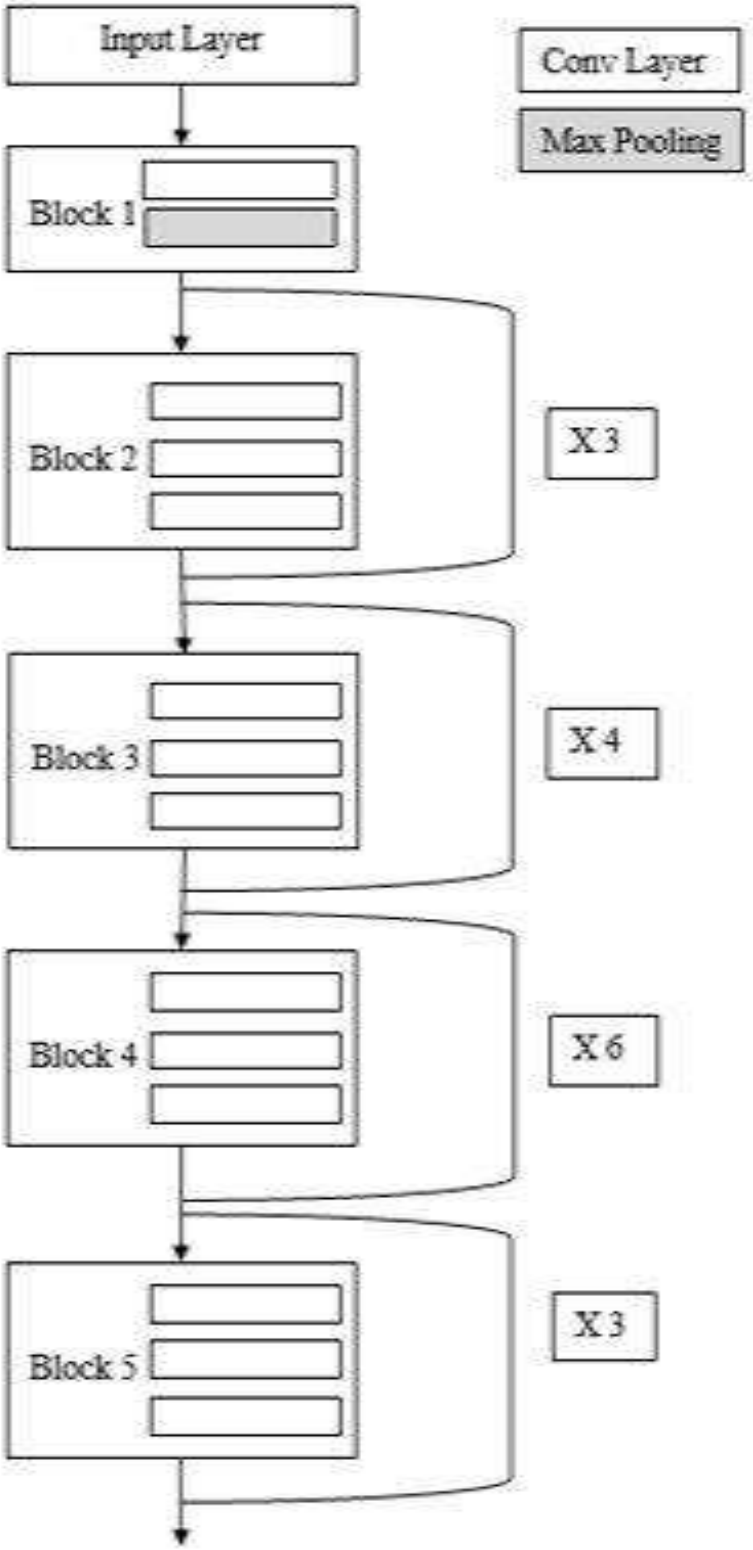


Figure 3.7: Frozen Layer of ResNet50V2 Architecture

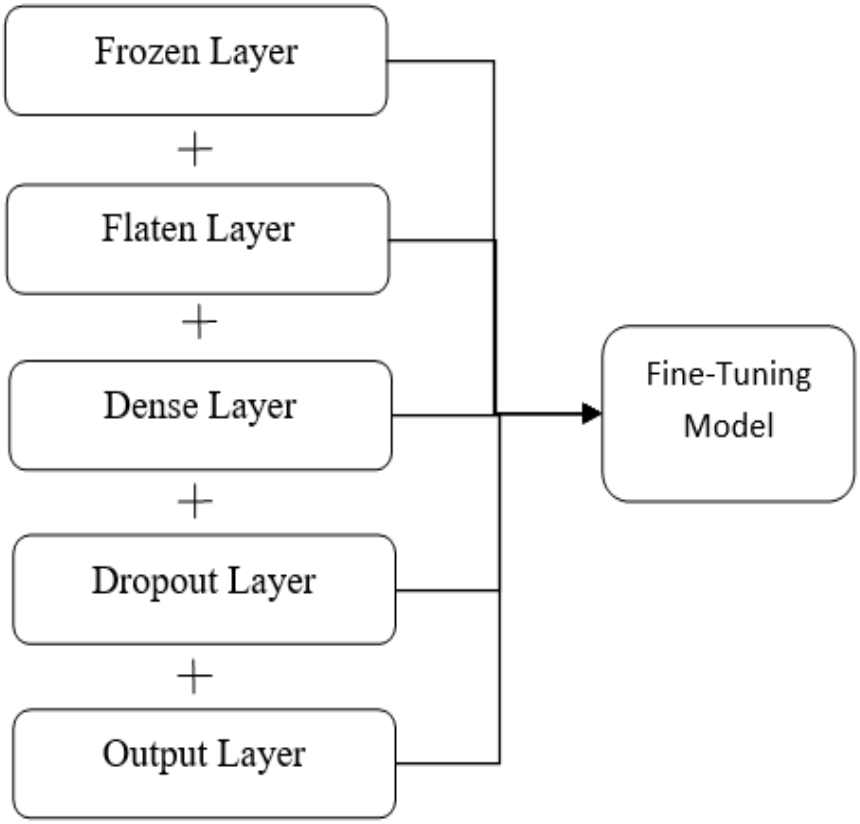


Figure3.8: Modified ResNet50V2 Architecture

In Figures 3.8 and 3.9, I present the upgraded ResNet50V2 model, which incorporates a newly integrated flattening layer aimed at expediting data processing. This architectural refinement is strategically designed to enhance information propagation within the network by optimizing data transfer and processing efficiency. By implementing this modification, my goal is to elevate the model's performance and augment its capacity for comprehending intricate patterns embedded within the neural network's layers. The introduction of the flattening layer serves a crucial role in streamlining the transformation of multidimensional data into a one-dimensional format. This process facilitates smoother data flow throughout the network, thereby enhancing the model's efficacy in extracting significant features from the input data. By simplifying the data representation process, the flattening layer enables the model to more efficiently capture and analyze complex information, ultimately leading to improved performance in various tasks, especially those demanding intricate pattern recognition, such as image classification. The integration of the flattening layer underscores our commitment to optimizing the ResNet50V2 model for real-world applications. By enhancing the model's ability to discern subtle variations and intricate

details within the data, I aim to ensure more accurate classification outcomes across a wide range of tasks and scenarios. This enhancement not only bolsters the model's performance but also enhances its versatility and applicability in diverse domains, further solidifying its position as a powerful tool in the realm of deep learning-based image analysis and classification.

Moreover, the inclusion of the flattening layer contributes significantly to the model's versatility and adaptability across diverse datasets and applications. By simplifying the data representation process, the model becomes more proficient at handling varying input formats and extracting relevant features, thereby enhancing its performance in real-world scenarios. This architectural refinement reflects a strategic endeavor to optimize the ResNet50V2 model for enhanced performance and efficacy in tackling complex computational tasks, ultimately driving advancements in the state-of-the-art of deep learning-based image analysis and classification. The redesigned architecture is anticipated to enhance the model's ability to capture and leverage complex data patterns, resulting in improved performance across tasks such as picture categorization and pattern recognition. The additional layers depicted in Figure 3.9 contribute to processing efficiency. Specifically, the flattening layer serves as a conduit, converting multidimensional arrays into one-dimensional representations, thereby facilitating seamless data integration into subsequent layers. This transformation optimizes the network's capacity to extract relevant features from input data, thereby bolstering overall performance. These modifications are aimed at enhancing the network's ability to extract pertinent features from input data, ultimately aimed at improving the model's performance. The overarching objective of these alterations is to empower the model to effectively analyze and learn intricate patterns within the dataset, thereby enhancing its ability to make accurate predictions or classifications. This deliberate approach is intended to refine the model's understanding and representation of complex data structures, thus enhancing its proficiency across a wide range of image analysis tasks and contributing to advancements in the field of deep learning-based image processing.

The improved ResNet50V2 model stands as a noteworthy breakthrough in the realm of deep learning architecture, characterized by a meticulously crafted balance between complexity and computing efficiency. Through deliberate modifications to its architecture, researchers aim to not only enhance the model's performance but also ensure the optimal utilization of computational resources. This iterative refinement

process plays a crucial role in empowering the model to effectively capture and leverage subtle patterns inherent in the data, thereby enhancing its predictive capabilities across a myriad of applications. Beyond mere performance enhancement, the architectural enhancements introduced in the ResNet50V2 model are poised to elevate its overall functionality and versatility. By bolstering the model's capability to handle diverse and intricate data structures, researchers aspire to expand its applicability across a broad spectrum of tasks and datasets. This upgrade is geared towards enhancing the model's adaptability to various tasks and datasets by fine-tuning its design to better accommodate complex data patterns. The strategic modifications made to the model's architecture are strategically aimed at augmenting its capacity to discern and leverage complex data features, thereby improving its performance and versatility across diverse applications. These adjustments empower the model to thrive in varied settings by adeptly adapting to the nuances present in different datasets and scenarios. Through the refinement of both the architecture and training procedures, researchers seek to optimize the model's efficacy in recognizing intricate patterns within the data. By meticulously adjusting the architecture and fine-tuning the training procedures, researchers aim to enhance the model's ability to capture subtle variations and complex relationships present in the data. These enhancements enable the model to consistently deliver robust performance across a wide spectrum of tasks and scenarios, ranging from image classification and object detection to natural language processing and beyond. The iterative refinement process ensures that the model remains adaptable and capable of handling the diverse challenges posed by real-world applications, thereby reinforcing its relevance and impact in various domains.

The enhanced ability to handle complex data features not only ensures dependable results but also broadens the model's utility and impact across various domains such as image classification, object detection, and pattern recognition. This underscores the pivotal role of ongoing research and development endeavors in advancing deep learning architectures to meet the ever-evolving demands of real-world applications. As technology continues to evolve and datasets become increasingly diverse and complex, the need for sophisticated deep learning models capable of effectively handling such challenges becomes more pronounced. By continuously refining and optimizing deep learning architectures like ResNet50V2, researchers can stay at the forefront of innovation, driving progress in fields ranging from healthcare and autonomous driving

to natural language processing and beyond. This commitment to advancement ensures that deep learning continues to push the boundaries of what is possible, enabling transformative solutions to some of the most pressing challenges facing society today.

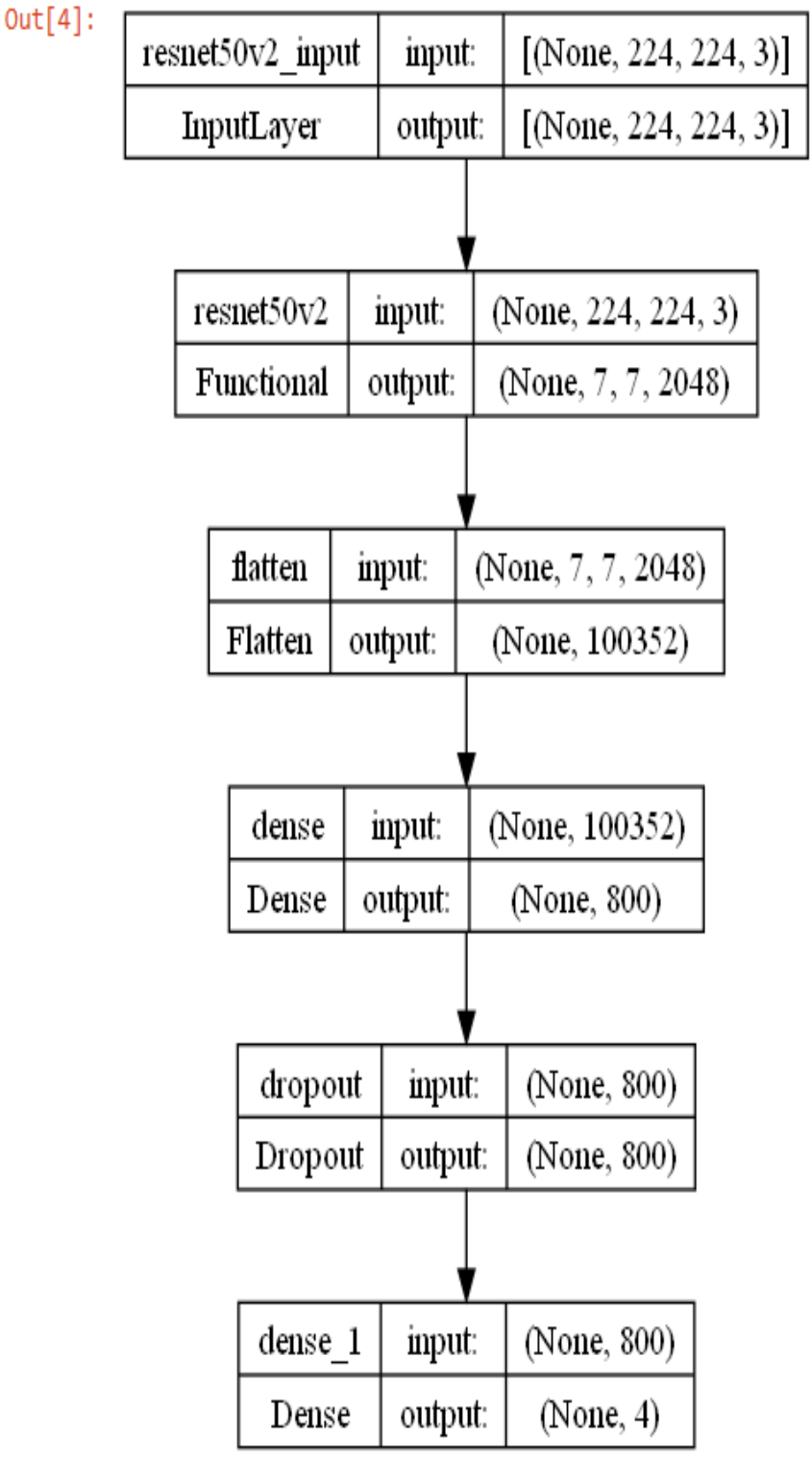


Figure 3.9: Modified ResNet50V2 Architecture

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50v2 (Functional)     (None, 7, 7, 2048)        23564800

 flatten (Flatten)           (None, 100352)            0

 dense (Dense)               (None, 800)               80282400

 dropout (Dropout)           (None, 800)               0

 dense_1 (Dense)             (None, 4)                 3204

=================================================================
Total params: 103,850,404
Trainable params: 80,285,604
Non-trainable params: 23,564,800
_________________________________________________________________
```

Figure 3.10: Modified ResNet50V2 model summary

3.3.5 Model Training

In addition to fine-tuning hyperparameters for accuracy optimization, my efforts to enhance the model's performance on tomato detection tasks involved rigorous training on a specialized Tomatoes Dataset. Before training commenced, preprocessing steps were undertaken to clean the data and eliminate any irrelevant information. Subsequently, the dataset was divided into distinct training and testing sets to facilitate the evaluation of the model's performance. During the training phase, the model was exposed to the training set while employing various techniques such as data augmentation and regularization to combat overfitting. These strategies played a pivotal role in enhancing the model's ability to generalize to unseen data and achieve superior performance in tomato detection tasks. By systematically incorporating these methodologies, I ensured that the model was not only effectively trained but also fine-tuned to its maximum potential in accurately identifying and classifying tomatoes across diverse scenarios. This comprehensive and systematic approach underscores our commitment to developing a robust and reliable model for tomato detection. By meticulously adjusting hyperparameters and employing advanced training techniques, I aimed to optimize the model's performance and bolster its capability to tackle real-world challenges in agricultural and food processing industries.

Two optimizers, Adam and Adamax, were employed to assess their efficacy in training the updated model. Adamax, an extension of the Adam optimizer, introduces additional functionality to handle sparse gradients more effectively. The objective of this comparative analysis was to discern which optimizer yielded superior results for the updated model. Through meticulous evaluation of their performance, researchers aimed to pinpoint the optimizer that optimized convergence speed, stability, and overall efficacy in adjusting the model's parameters. This comparative research furnished valuable insights into the relative strengths and limitations of each optimizer, facilitating the selection of the optimal optimization strategy to enhance the model's performance in tomato detection tasks. By systematically comparing the performance of Adam and Adamax, researchers could make informed decisions regarding the choice of optimizer, thereby refining the training process to maximize the model's accuracy and efficiency.

The training input size of 224×224 pixels was selected to ensure the model could capture adequate details and features from the images. A batch size of 32 was utilized to strike a balance between computational efficiency and model convergence during training. This configuration enabled efficient processing of data in batches, facilitating effective parameter updates while minimizing computational overhead. These considerations are crucial for achieving high-performance results while effectively managing computational resources in training deep learning models.

3.3.6 Model Testing

In the evaluation phase, the model's performance is rigorously tested on tomato imagery from the dataset. Initially, a random sample test image is selected, and the model generates a prediction based on its learned parameters. These predictions are then validated against existing datasets to ensure accuracy and reliability. Subsequently, the model's performance is assessed on hypothetical data by computing metrics such as accuracy, recall, and F1 score. These metrics provide valuable insights into the model's ability to correctly classify tomatoes across various conditions and scenarios. Further evaluation is conducted on unseen Kaggle datasets to gauge the model's generalization ability and its effectiveness in recognizing tomatoes in diverse photographs. A precision vs. recall curve is plotted to assess the accuracy of the categorization process, offering a comprehensive view of the model's performance. To refine and improve the model,

adjustments to hyperparameters such as learning rate and number of epochs are made. The model's performance is continuously monitored, and modifications are implemented as necessary based on its confidence levels and overall efficacy. Finally, the model's performance is evaluated using assessment metrics such as accuracy or precision, providing a comprehensive understanding of its effectiveness in tomato detection tasks. Comparative analysis with other models further validates the model's efficacy and highlights areas for improvement, ultimately guiding future iterations and advancements in deep learning-based image analysis.

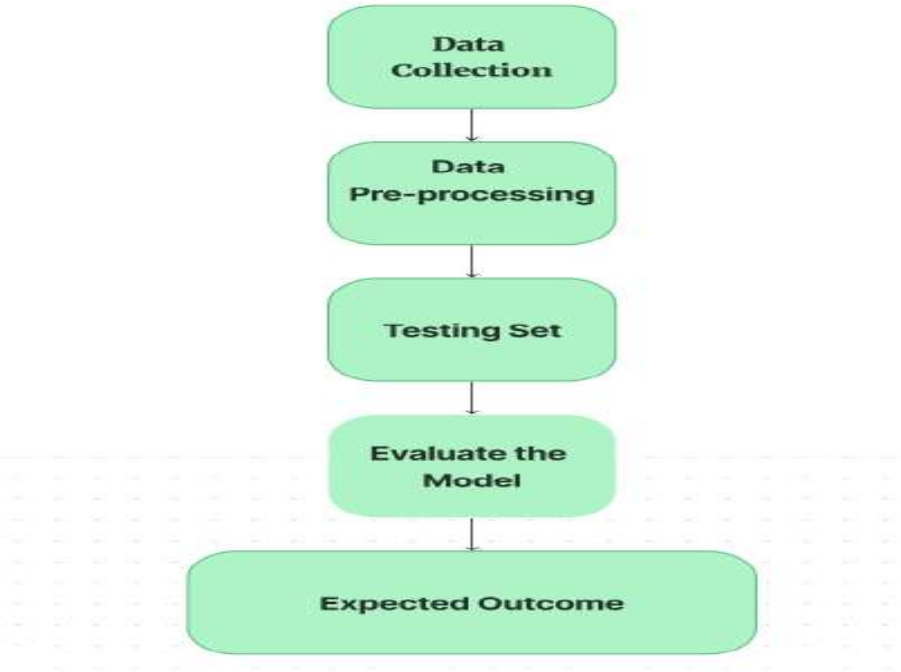
3.4 Summary

The ResNet50V2, a deep convolutional neural network (CNN), is renowned for its pre-trained layers that possess a comprehensive understanding of image characteristics such as shape, color, and structure. One of the main factors contributing to the ResNet50V2's outstanding image analysis performance is its extraordinary depth. This depth enables the model to delve deeply into the minute features included in pictures, revealing minute differences and nuances that could otherwise go missed. Consequently, the model displays an increased susceptibility to the many attributes seen in various kinds of images and environments. From a practical standpoint, this depth means that images of any complexity or diversity can be uniquely analyzed to extract useful features. The amazing precision with which the ResNet50V2 captures the core of visual input is demonstrated in its ability to distinguish fine textures, subtle color gradients, and complicated patterns. When working with different datasets that include a wide range of image types—from natural scenes to industrial surroundings, and everything in between—this functionality is especially helpful. Furthermore, because of its depth, the model can learn hierarchical feature representations, in which more complex ideas are constructed from simpler primitives. This hierarchical method improves the model's resilience and adaptability while also improving its comprehension of images. The ResNet50V2 gains proficiency in processing a wide range of picture data by learning to identify both simple and complex visual components. This ensures dependable performance in a variety of real-world applications. Fundamentally, the ResNet50V2's depth is a key component in its capacity to efficiently traverse the complex and varied terrain of visual data, which makes it an excellent tool for image analysis jobs in domains like computer vision, medical imaging, and more. Thanks to its extensive

training on a vast repository of images, the ResNet50V2 demonstrates remarkable adaptability and generalization capabilities when presented with new datasets. Moreover, it exhibits robustness and versatility in effectively managing a wide range of image features, highlighting its prowess in complex image analysis tasks.

3.5 Methodology (For Testing)

For the sake of testing, this is our operational procedure.



Working Flowchart for Testing

3.5.1 Initial Data Gathering

Start by collecting primary data from relevant sources. Ensure the collected data covers the necessary aspects of your project and is of good quality. Organize the data in a structured format for further analysis.

3.5.2 Data Preparation

Clean and preprocess the collected data by addressing any missing values, outliers, or inconsistencies. Standardize or normalize the data to ensure consistency across different features. Convert categorical data into numerical formats if required. Split the preprocessed data into testing sets for model development and evaluation.

3.5.3 Creation of Testing Set

Allocate a portion of the preprocessed data specifically for testing purposes.

This segregated dataset will be used to assess the model's performance independently.

3.5.4 Model Evaluation

Choose an appropriate machine learning algorithm based on your problem's nature and data characteristics. Train the selected model using the training dataset. Fine-tune model parameters to optimize its performance. Evaluate the model's effectiveness using various metrics such as accuracy, precision, recall, or mean squared error. Utilize techniques like cross-validation to validate the model's generalization capabilities.

3.5.5 Analysis of Expected Results

Analyze the performance of the trained model and derive insights from the obtained results. Identify areas for potential enhancement based on the model's performance metrics. Adjust the model architecture or data preprocessing methods as needed. Iterate through the process as required until the desired level of performance is achieved. Anticipate deploying the trained model for making predictions on new, unseen data.

CHAPTER 4

Results and Discussion

4.1 Introduction

The full experimental assessment of the model being offered is based on tomato and its category Prediction for diverse images monitoring using modified model described in this chapter.

4.2 Performance parameters

A numerical declaration of the representational work and its outcomes might serve as a performance measure. Measures of performance are sponsored data that show clearly if representation or action is accomplishing its objectives and whether policy or organizational goals are being promoted. I used an assessment matrix consisting of precision, recall, F1-score, true negative rate, and false-positive rate accuracy to determine how well my proposed model performed. In classification evaluation, precision and recall are important measures that evaluate how well a model identifies affirmative class instances. Recall assesses the fraction of true positives that are correctly detected, whereas precision counts the percentage of correctly classified positives. Both metrics provide information about how well a model performs in classification tasks. The precision of positive predictions is highlighted by measuring the percentage of accurately predicted positive cases among all positive instances that are labeled as such. Conversely, recall measures the percentage of accurately predicted positive occurrences among all true positive instances in the dataset, emphasizing the model's capacity to include all positive examples. The tradeoff between precisely recognizing positive examples (precision) and thoroughly capturing all positive instances (recall) is balanced by these measurements, which offer complementary insights into the model's performance. Assessing accuracy and recall facilitates the assessment of the model's efficacy in accurately classifying positive events and provides guidance for optimization tactics to improve its performance. Recall, calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN), quantifies the model's sensitivity to positive instances.

Recall = TP / (TP + FN) ……..(4.1)

Precision, on the other hand, is computed as the ratio of true positives to the sum of true positives and false positives (FP), characterizing the accuracy of positive predictions.

Precision=TP / (TP + FP)…….(4.2)

The F1-score, representing the weighted average of precision and recall, encapsulates both false positives and false negatives in its estimation. Mathematically, the F1-score is expressed as

F1 = 2 * (Recall * Precision) / (Recall + Precision) ……...…….(4.3)

True Negative Rate (TNR), denoting the proportion of samples correctly identified as negative among those tested negative, is computed as TNR = TN / (FP + TN). Conversely, False Positive Rate (FPR), an accuracy metric applicable to specific machine learning models, is calculated as

F1 = FP / (FP + TN)……..……...(4.4)

Accuracy, as an overall measure of correctly classified samples, is determined by the ratio of the sum of true positives and true negatives to the total number of samples.

Accuracy = (TP + TN) / (TP + TN + FP + FN)……...…….(4.5)

In comparison to the classification matrix, True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) represent the counts of properly and erroneously categorized cases. These metrics are crucial for evaluating the model's performance. True Positive indicates successfully recognized positive instances; False Positive indicates negative instances misclassified as positive; True Negative indicates correctly identified negative instances; and False Negative implies positive instances incorrectly classified as negative. By analyzing these quantities, researchers gain insights into the model's accuracy, precision, recall, and other performance metrics, allowing for a thorough evaluation of its effectiveness in correctly classifying instances and informing potential improvements to improve its predictive capabilities.

4.3 Experimental Analysis

The modified CNN architecture, initialized with pre-trained ImageNet weights for classification, underwent rigorous evaluation on the Tomato Dataset. This assessment aimed to gauge the model's performance in accurately identifying tomatoes, crucial for

agricultural applications and optimizing yield. The evaluation process for the modified CNN included robust validation set support, where the model's performance was thoroughly assessed using classification accuracy metrics and a confusion matrix. This comprehensive analysis enabled a detailed understanding of the model's ability to accurately classify tomatoes and provided insights into potential areas for improvement, ensuring its effectiveness for agricultural applications and yield optimization. The evaluation formed the foundation for comparing and analyzing the results against architectures presented in existing literature, indicating a systematic approach to benchmarking the performance of the proposed methodology against established methods documented in prior research.

TABLE4.1: THE HYPER PARAMETER AND PARAMETER FOR OUR MODEL

Hyper parameter	**Parameter value**
Number of Epoch	25
Batch Size	32
Optimizer	Adam,Adamx
Learning rate	0.001
Objective function	Sparse Categorical Crossentropy
Hidden layer	ReLu
Output layer	Softmax

In the experimental analysis, two optimization techniques, Adam and Adamax, were employed with a shared learning rate of 0.001 and a consistent batch size of 32. The model's performance was evaluated using both optimizers, with Adamax serving as an alternative. Notably, both optimization strategies exhibited improved accuracy after 25 epochs of training. This suggests that the model benefited from the optimization techniques implemented, regardless of the specific algorithm utilized. The choice to include Adamax as an alternative optimizer indicates a thorough exploration of optimization methods to identify the most effective strategy for the given task. The consistent learning rate and batch size across experiments ensure a fair comparison between optimization techniques. The observed enhancement in accuracy ensure underscores the effectiveness of the optimization process in refining the model's performance, contributing valuable insights into the impact of optimization algorithms on the overall efficacy of the deep learning model. All experiments employed

optimizers with adaptive learning rates and utilized the sparse cross-entropy function as the loss as function. This approach ensures efficient optimization by adjusting learning rates dynamically. The sparse cross-entropy loss function is suitable for scenarios with sparse target labels, making it ideal for classification tasks with many classes. The meticulous optimization and evaluation process are essential for refining the performance of the convolutional neural network (CNN) model to effectively detect tomatoes in the provided dataset. Through rigorous experimentation with various parameters, such as optimizers with adaptive learning rates and the sparse cross-entropy loss function, the model's accuracy and generalization capabilities can be enhanced. The Convolutional Neural Network (CNN) model is refined iteratively until it achieves optimal performance in precisely identifying tomatoes, which is important for agricultural applications and yield optimization. These iterative improvements entail ongoing modifications and improvements to the training protocols, data preprocessing methods, and model architecture. Through iteratively fine-tuning the model parameters and optimizing its performance measures, the CNN learns to accurately detect tomatoes in farming environments. To maximize crop yields, farmers must be able to manage their produce effectively and make well-informed decisions. This leads to increased agricultural production and profitability.

4.3.1 Performance Analysis of Adam optimizer

By looking at the Figure 4.1 training plot, I can observe that the validation loss tracks its training loss, suggesting that the dataset itself does have less overfitting:

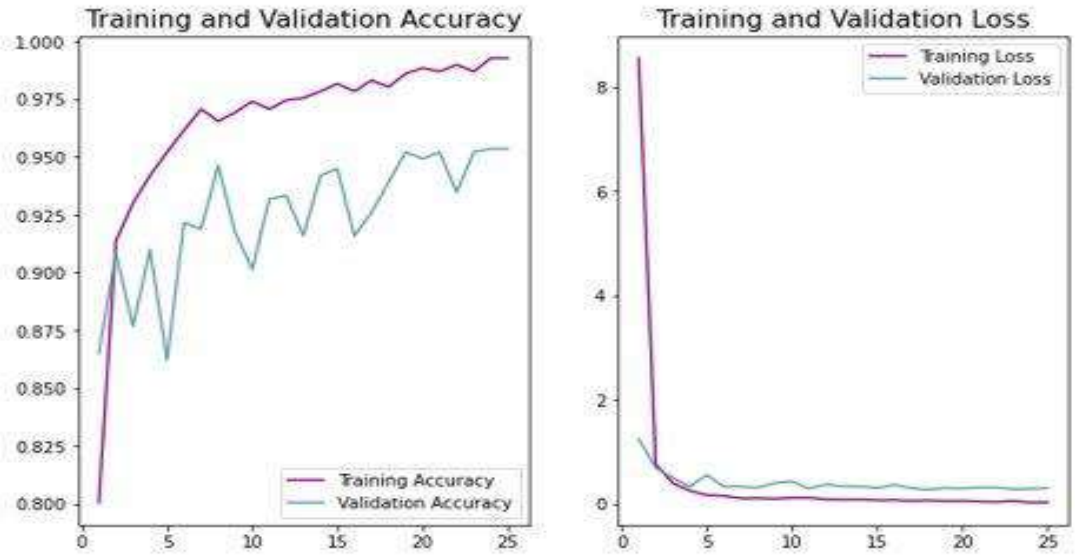
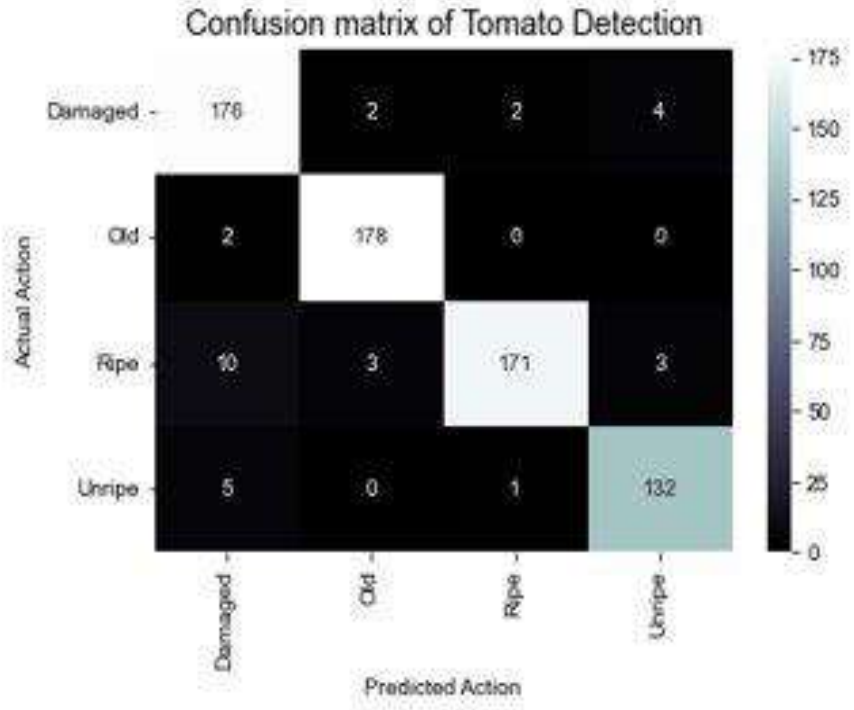


Figure 4.1: Accuracy and Loss curve in Adam Optimizer in 25 epoch



Figur4.2: Confusion matrix of Adam Optimizer for our model

```
22/22 [==============================] - 98s 4s/step
None
              precision    recall  f1-score   support

     Damaged       0.91      0.96      0.93       184
         Old       0.97      0.99      0.98       180
        Ripe       0.98      0.91      0.95       187
      Unripe       0.95      0.96      0.95       138

    accuracy                           0.95       689
   macro avg       0.95      0.95      0.95       689
weighted avg       0.95      0.95      0.95       689
```

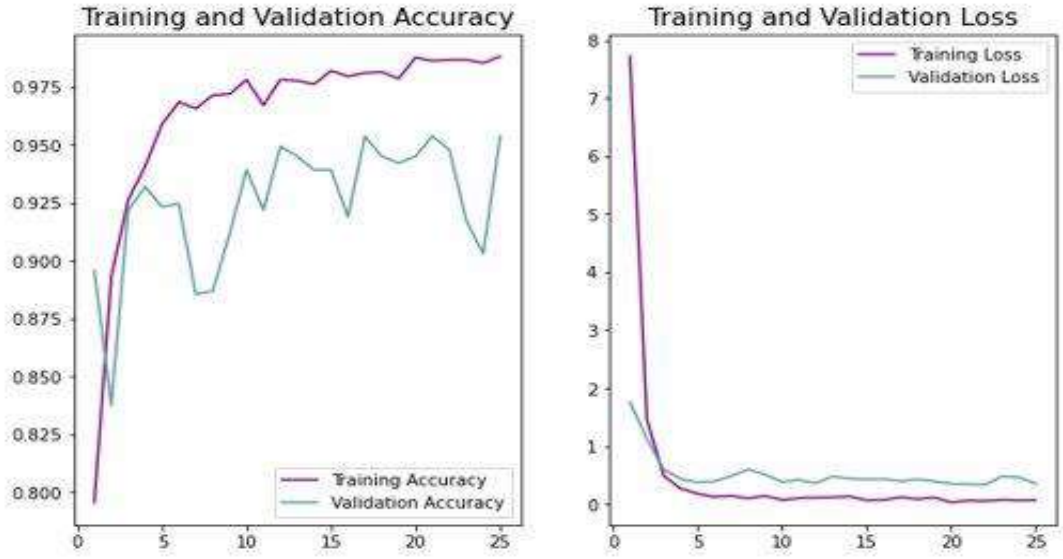
Figure 4.3: Classification Report of Adam Optimizer for our model

As the number of epochs rose, it is clear from the Figure 4.3 confusion matrix and Figure 4.6 classification report that the precision, f1-score, recall, support, and accuracy all increased. On my 3345 data for four classes, I also employed 25 epochs of the Adam

optimizer in this respect and achieved 95.36% testing accuracy. Adam performed marginally better when dealing with my model, which has a learning rate of 0.001.

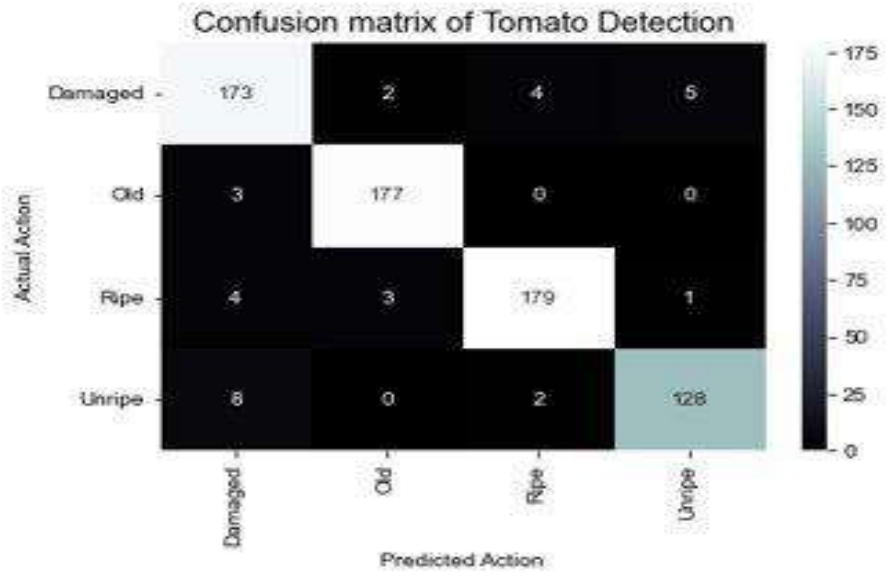
4.3.2 Performance Analysis of Adamax optimizer

This Following figure 4.4 shows how training accuracy rises and training loss declines as the frequency of epochs rises.



Fgure 4.4: Accuracy and Loss curve in Adamax Optimizer in 25 epoch

Using Adamax optimizer, I trained for a total of 25 epochs, and my test results showed that the training and validation sets had an accuracy of 95.36% and the testing set had an accuracy of about 95%. Our model's performance is seen in the following Figure 4.5 confusion metric and Figure 4.6 classification report.



Figur4.5: Confusion matrix of Adamax Optimizer for our model

```
22/22 [==============================] - 64s 3s/step
None
              precision    recall  f1-score   support

     Damaged       0.92      0.94      0.93       184
         Old       0.97      0.98      0.98       180
        Ripe       0.97      0.96      0.96       187
      Unripe       0.96      0.93      0.94       138

    accuracy                           0.95       689
   macro avg       0.95      0.95      0.95       689
weighted avg       0.95      0.95      0.95       689
```

Figure 4.6: Classification Report in Adamax Optimizer in our model

4.4 Experimental Analysis Existing model in our dataset

The experimental examination of an existing model inside our dataset included careful evaluation and comparison to predetermined performance indicators. The results were most likely presented in a structured table format, which allowed for a clear depiction of the model's performance across multiple tests and conditions. The rigorous evaluation of these results revealed insights into the model's efficacy, strengths, and opportunities for improvement. This study most certainly influenced later decisions on model refinement, optimization tactics, and prospective enhancements, adding to the iterative process of model development and validation within the context of the dataset and research objectives.

Table4.2: THE HYPER PARAMETR AND PARAMETR FOR EXISTING MODEL

Hyper parameter	**Parameter value**
Number of Epoch	25
Batch Size	32
Optimizer	Adam, Adam , Adamx
Learning rate	0.0001,0.001,0.001
Objective function	Sparse Categorical Crossentropy
Hidden layer	ReLu
Output layer	Softmax

According to [96],I trained their model with an adaptive 0.0001 learning rate for Adam. Then I demonstrated the both Adam and Adamax optimization, when 0.001 learning rate was used, and the batch size was set to 32 with 25 epochs. I did it because our model learning rate was 0.001for both of them. The optimization process was particularly focused on models utilizing adaptive learning rates with sparse cross-entropy as the chosen loss function. The use of adaptive learning rates is notable for dynamically modifying during training, resulting in optimal convergence for a variety of input patterns. This dynamic modification improves the model's ability to respond to various complexities in the data, hence increasing training efficiency. Meanwhile, the loss function used is sparse cross-entropy, which is geared to scenarios with integer target values and meets the study's special needs. Sparse cross-entropy is an effective measure of the disparity between predicted and true labels, making it ideal for classification jobs with many classes. Together, these strategies add to the model's robustness, allowing for effective learning and accurate predictions in settings with a variety of data patterns and integer target values. To ascertain their effect on the model's performance and training, the study carefully examined the Adam and Adamax optimization methods in a number of scenarios. Researchers sought to determine the relative effects of different optimization techniques on training stability, convergence speed, and overall performance by methodically comparing them. Popular adaptive learning rate algorithms Adam and Adamax take different ways to calculate the changes to the learning rates. The overarching goal of the investigation was to identify the observable benefits and drawbacks associated with each optimization method within the context of the scenarios explored, employing thorough testing and assessment. The comparative study provided meaningful insights into the most suitable optimization strategy to select based on the task parameters and the characteristics of the dataset under,scrutiny.

The central objective of the study was to determine the discernible advantages and disadvantages of each optimization approach within the confines of the situations considered, utilizing meticulous testing and evaluation. The comparative analysis yielded substantial insights into the preferred optimization strategy to pursue, contingent upon the specifications of the task and the attributes of the dataset examined. The comparative research yielded significant insights into the best optimization strategy to choose depending on the task specifications and dataset attributes.

4.4.1 Performance Analysis of Adam optimizer in existing model

This Following figure 4.7 graph shows that as the quantity of epochs grows, loss drops and accuracy rises.

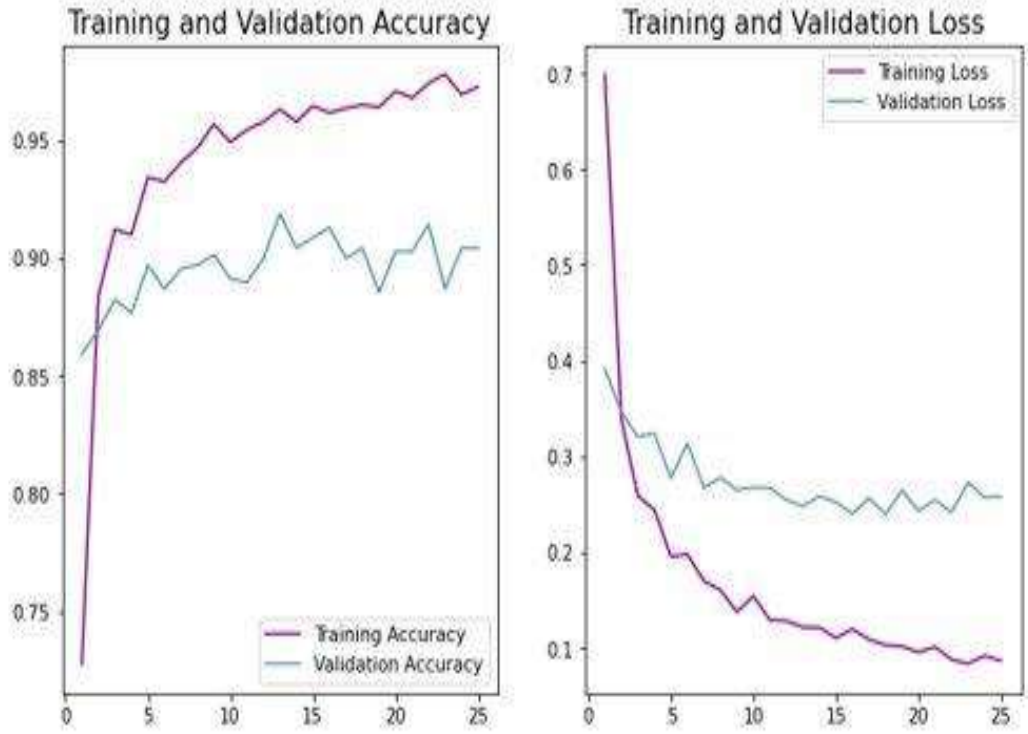


Figure 4.7: Accuracy & loss curve in Adam Optimizer for existing model

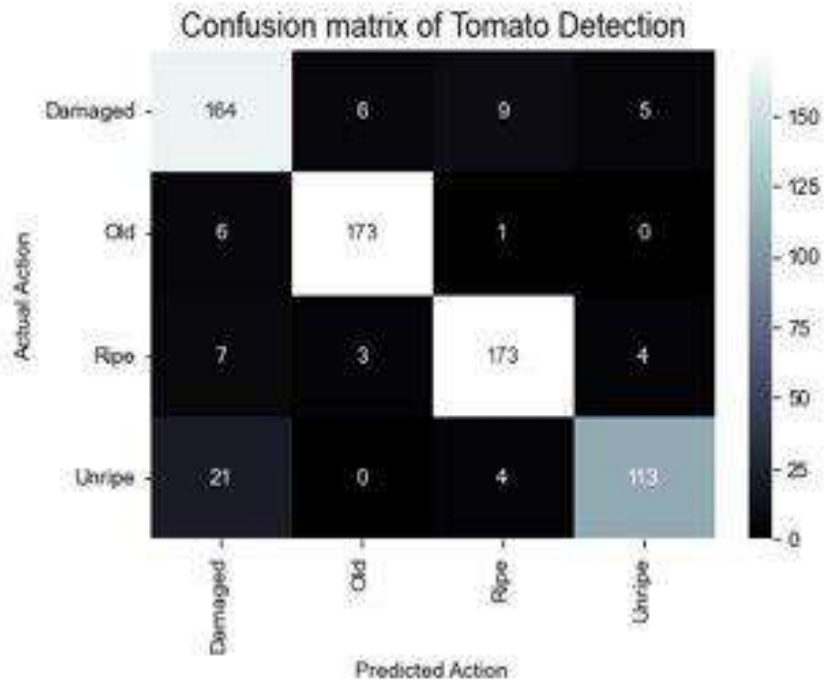


Figure 4.8: Confusion Matrix of Adam optimizer on existing model

```
22/22 [==============================] - 71s 3s/step
None
              precision    recall  f1-score   support

     Damaged       0.83      0.89      0.86       184
         Old       0.95      0.96      0.96       180
        Ripe       0.93      0.93      0.93       187
      Unripe       0.93      0.82      0.87       138

    accuracy                           0.90       689
   macro avg       0.91      0.90      0.90       689
weighted avg       0.91      0.90      0.90       689
```

Figure 4.9: Classification Report of Adam optimizer on existing model

4.4.2 Performance Analysis of Adam optimizer in existing model

This Following figure 4.10 graph shows that as the quantity of epochs grows, loss drops, and accuracy rises.

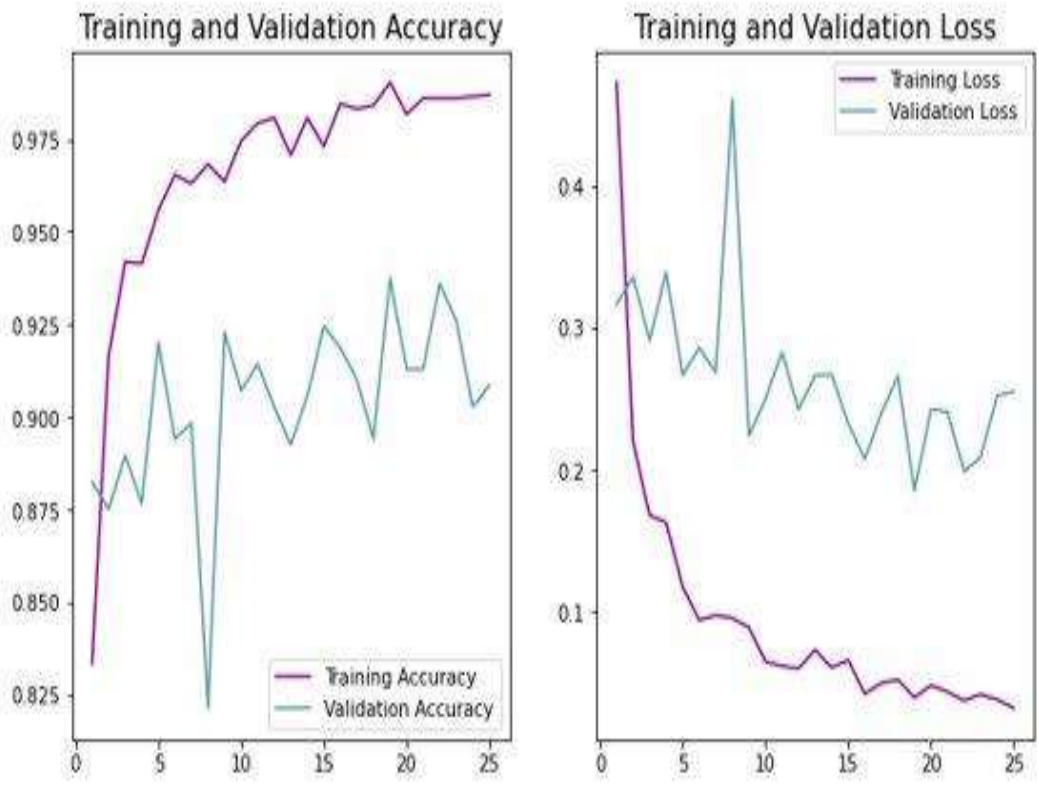


Figure 4.10: Accuracy & loss curve when Adam Optimizer for existing model with 0.001 learning rate

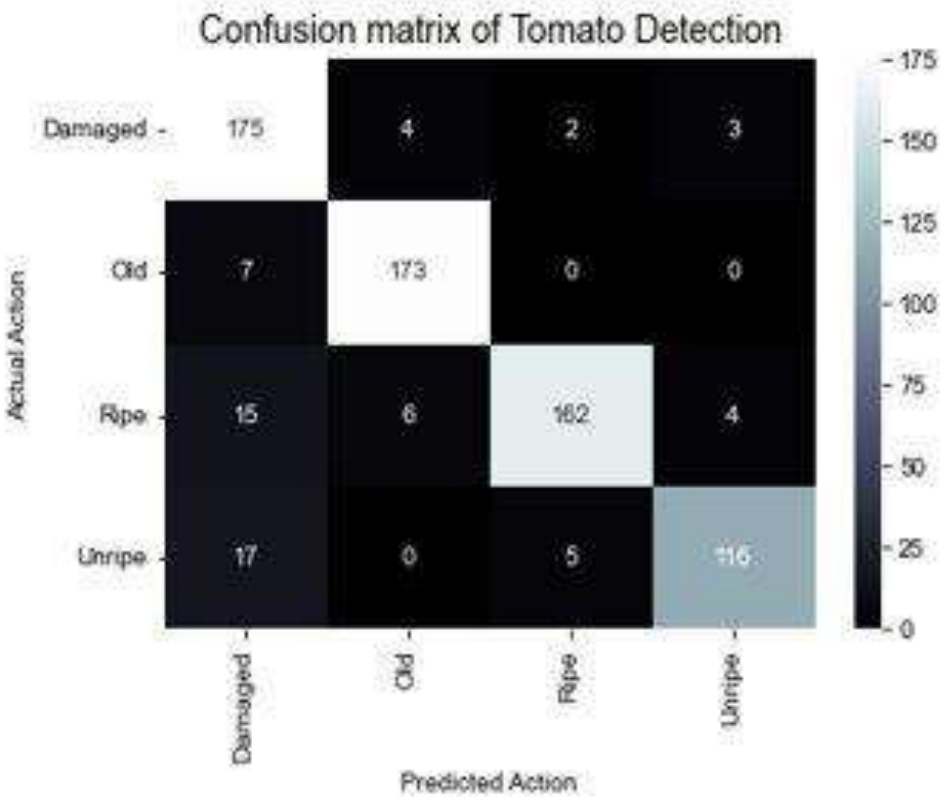


Figure 4.11: Confusion Matrix in Adam optimizer existing model with 0.001 learning rate

```
22/22 [==============================] - 69s 3s/step
None
              precision    recall  f1-score   support

     Damaged       0.82      0.95      0.88       184
         Old       0.95      0.96      0.95       180
        Ripe       0.96      0.87      0.91       187
      Unripe       0.94      0.84      0.89       138

    accuracy                           0.91       689
   macro avg       0.92      0.90      0.91       689
weighted avg       0.91      0.91      0.91       689
```

Figure 4.12: Classification Report in Adam Optimizer for existing model with 0.001 learning rate

4.4.3 Performance Analysis of Adamax optimizer in existing model

This following figure 4.13 graph shows that as the quantity of epochs grows, loss drops and accuracy rises.

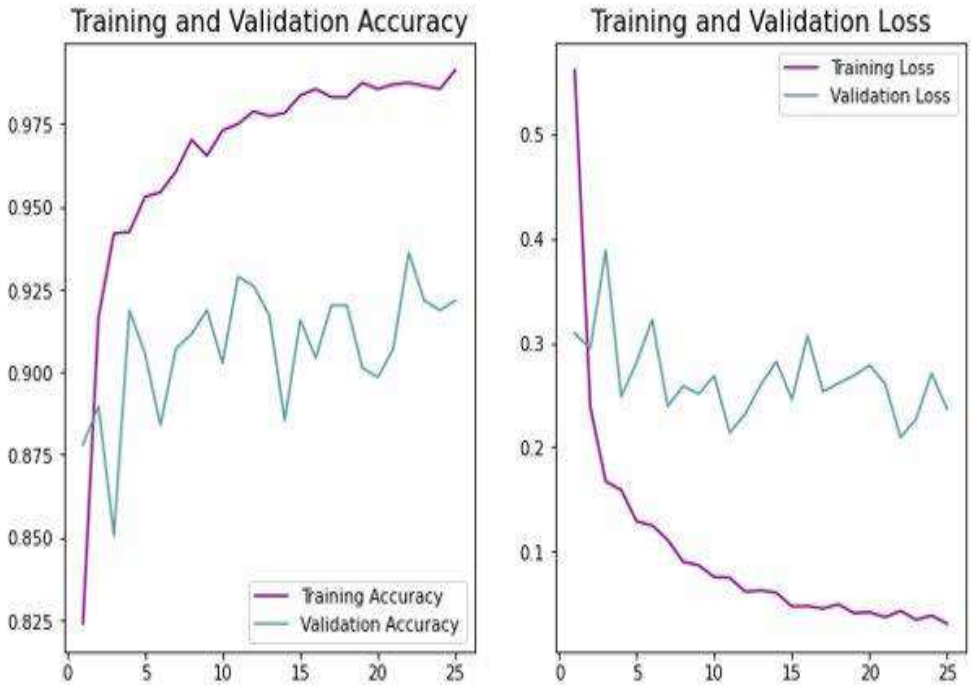


Figure 4.13: Accuracy & loss curve when Adamax Optimizer for existing model with 0.001 learning rate

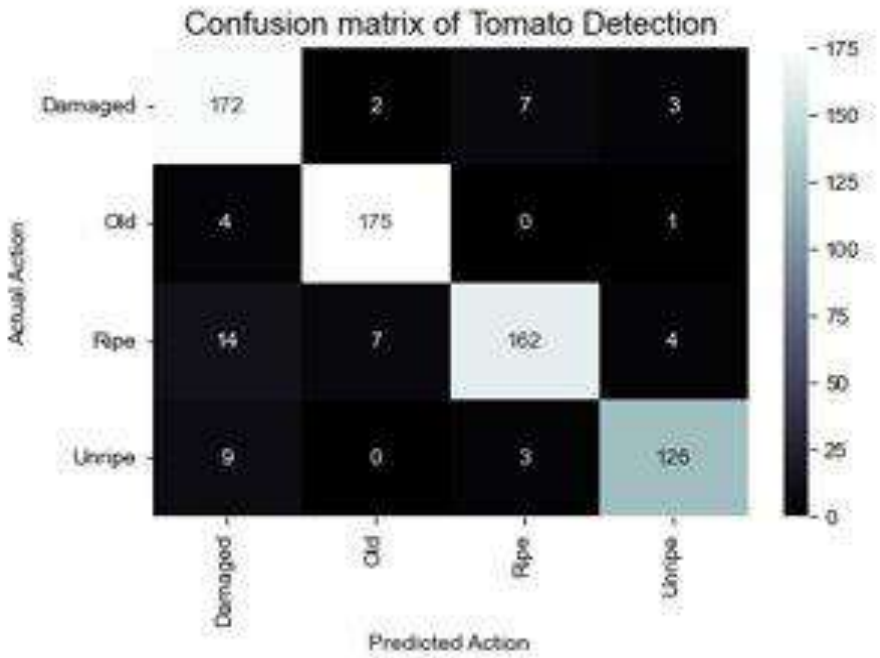


Figure 4.14: Confusion Matrix of Adamax Optimizer for existing model with 0.001 learning rate

```
22/22 [==============================] - 58s 3s/step
None
```

	precision	recall	f1-score	support
Damaged	0.86	0.93	0.90	184
Old	0.95	0.97	0.96	180
Ripe	0.94	0.87	0.90	187
Unripe	0.94	0.91	0.93	138
accuracy			0.92	689
macro avg	0.92	0.92	0.92	689
weighted avg	0.92	0.92	0.92	689

Figure 4.15: Classification Report of Adamax Optimizer for existing model with 0.001 learning rate

4.5 Predicted sample

Figures 4.13–4.16 show the predicted picture visualizations produced by my model. These graphic representations demonstrate the model's ability to reliably identify and forecast outcomes for each label. These figures, which visually demonstrate the model's performance, provide vital insights into its ability to correctly recognize and categorize different classes. Visualizations play a crucial role in aiding researchers and stakeholders in understanding the capabilities and limitations of a model, thereby enhancing comprehension of its predictions, and facilitating potential modifications or optimizations. By providing clear and intuitive representations of the data and the model's behavior, visualizations offer insights into how the model processes information and makes predictions. This transparency fosters trust in the model's capabilities, as stakeholders can directly observe its performance and understand the factors influencing its predictions. Furthermore, visualizations enable the identification of potential areas for improvement, guiding the development of more accurate and reliable models. Overall, visualizations serve as an effective tool for evaluating model performance, increasing understanding among stakeholders, and enhancing confidence in the model's prediction capabilities. Through visual representations, researchers and stakeholders can collaborate more effectively to refine models and ensure their suitability for various applications.

Figure 4.16: Predicted outcomes for Damaged Class

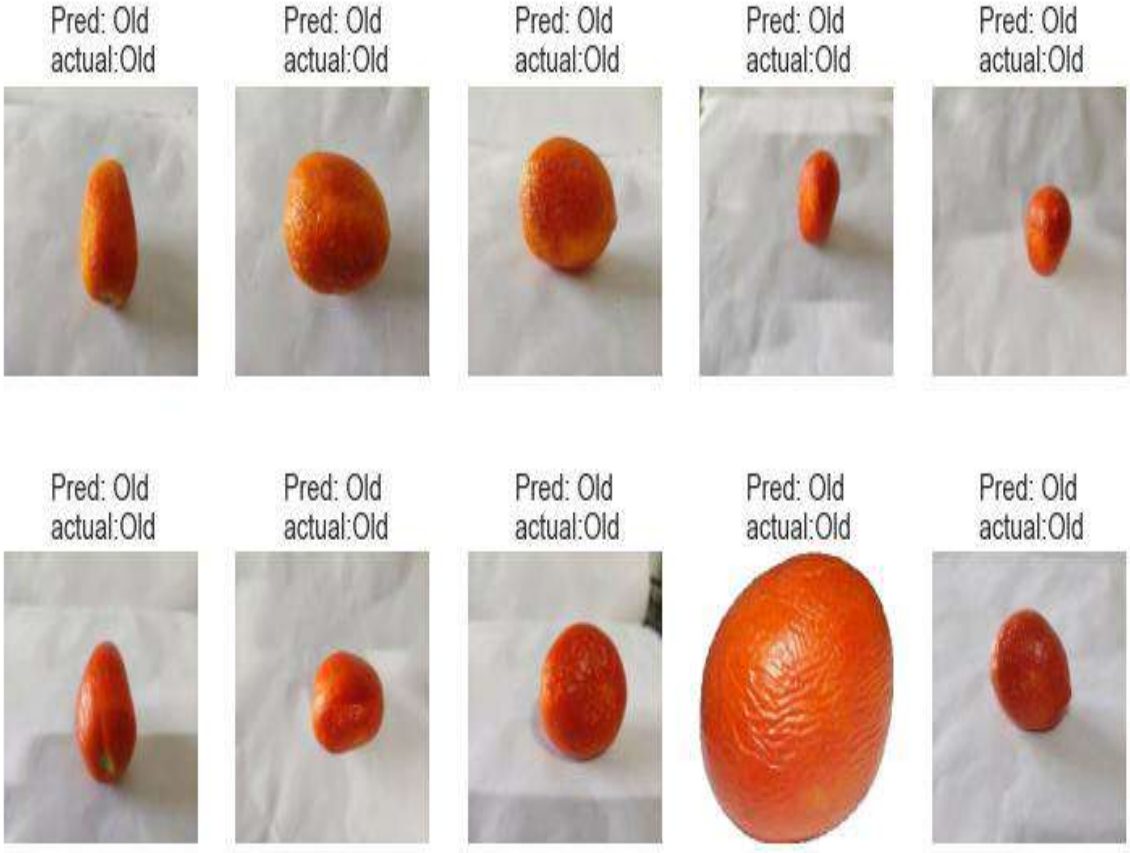


Figure 4.17: Predicted outcomes for Old Class

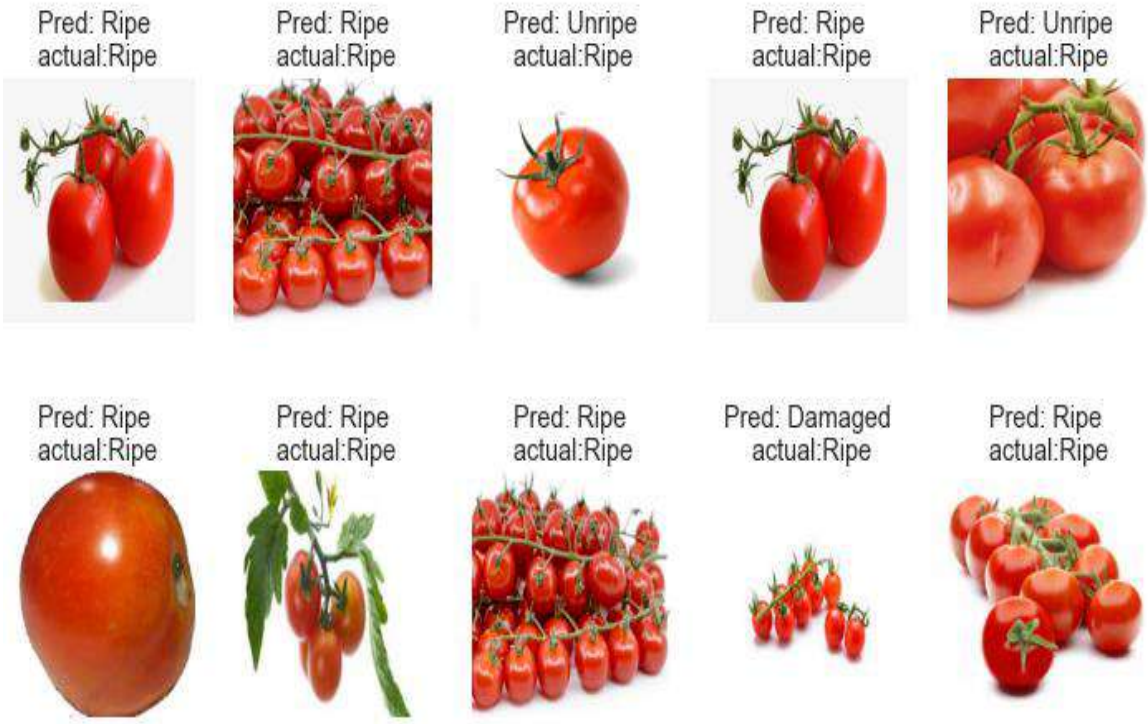


Figure 4.18: Predicted outcomes for Ripe Class

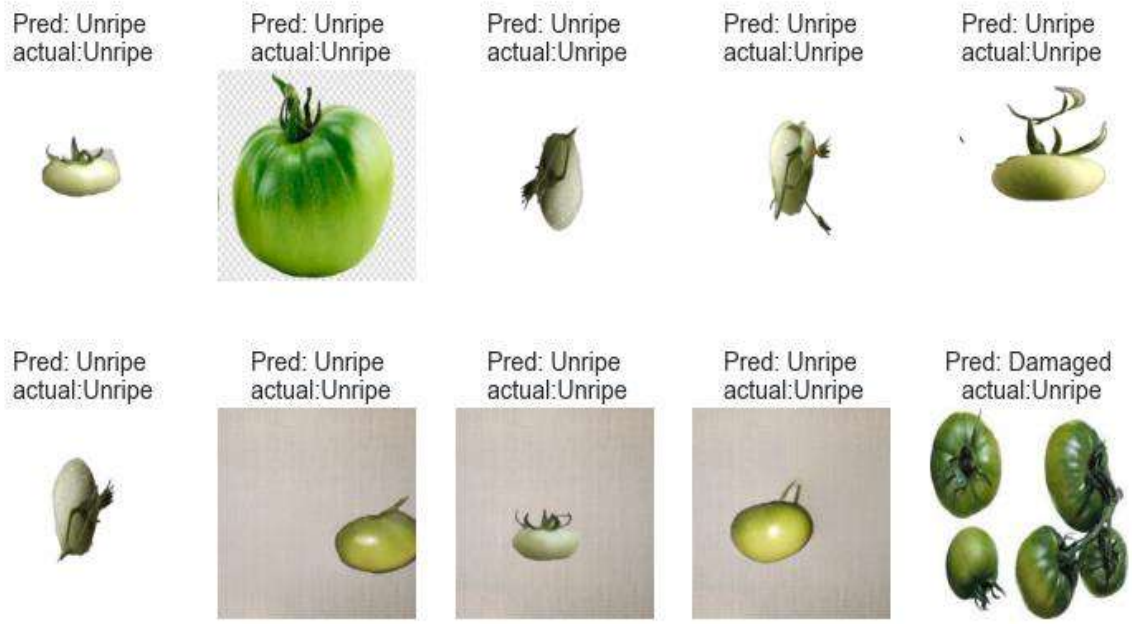


Figure 4.19: Predicted outcomes for Unripe Class

4.6 Result Analysis of existing model

This part describes how well my model performed using Adam, Adamax, Each optimizer's performance was thoroughly compared to determine its effectiveness. This

comparison analysis seeks to determine and comprehend the relative strengths and limitations of each optimizer in the context of the given task or model. A comparison table visually presents the performance of the model and evaluates the effectiveness of each optimizer, simplifying the assessment process based on performance criteria. This table likely includes metrics such as accuracy, precision, recall, and F1 score for each optimizer, allowing for a comprehensive comparison of their performance across various evaluation metrics. The comparison table showcases the performance of the two optimizers, Adam and Adamax, within our model. By visually summarizing the results in a structured format, stakeholders can easily compare the effectiveness of different optimizers and make informed decisions regarding their selection based on the desired performance criteria and objectives. In my model

Table 4.3: COMPARISON TABLE OF ABOVE TWO OPTIMIZERS IN OUR MODEL:

Optimizer	**Learning Rate**	**Epoch**	**Training Accuracy**	**Training Loss**	**Validation Accuracy**	**Validation Loss**
Adam	0.001	25	0.9928	0.0222	0.9536	0.2965
Adamax	0.001	25	0.9880	0.0700	0.9536	0.3546

I examined these optimizers using my dataset in an existing model to see how they affected accuracy and loss. Where the Adam optimizer was run twice at progressively higher learning rates of 0.0001 and 0.001, as well as a study of adamax with a learning rate of 0.001. It was evident from the comparison table that Adam and Adamax scored 90.16%, 90.86%, and 92.16%.

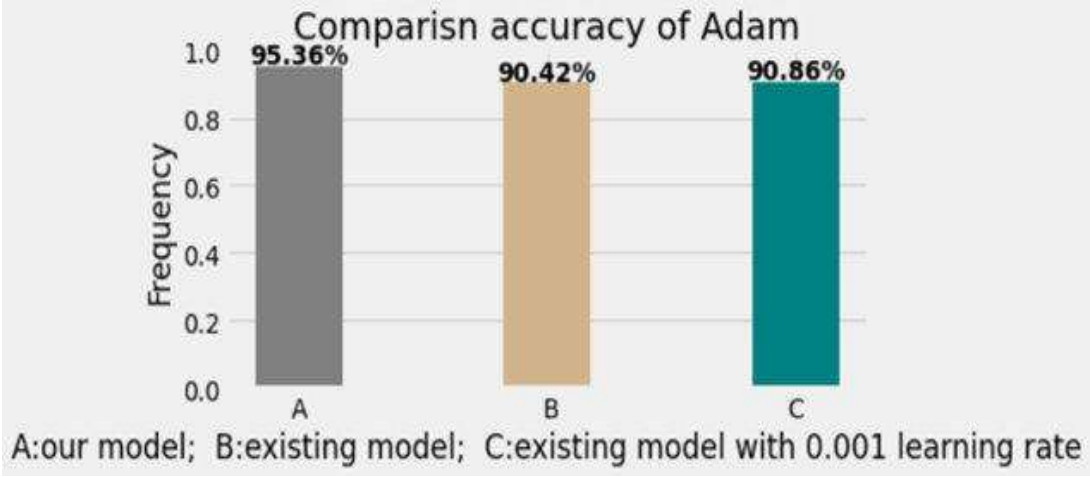
Table 4.4: COMPARISON TABLE OF ABOVE THREE OPTIMIZERS

Optimizer	**Learning Rate**	**Epoch**	**Training Accuracy**	**Training Loss**	**Validation Accuracy**	**Validation Loss**
Adam	0.0001	25	0.9726	0.0865	0.9042	0.2579
Adam	0.001	25	0.9870	0.0318	0.9086	0.2552
Adamax	0.001	25	0.9914	0.0308	0.9216	0.2366

4.7 Comparison ADAM With existence work

I compared our model with the existing paper [35] with our collected Tomatoes Dataset. I examined their model carefully. I also tested both Adam and Adamax optimizers on their model with 0.001 learning rate where their learning rate was 0.0001. My proposed model gave better accuracy than them, I used Adam optimizer for comparison. The Comparison accuracies in the optimizer- Adam respectively 95.36%, 90.42%, 90.86%.

Figure 4.20: Comparison of validation accuracies of 3 adam optimizers with our model & their proposed system.



4.8 Discussions

I made a comparatively wide dataset and built a model, which I modified from ResNet%0V2. I utilized existing models to test the validity of our model with every optimizer employed in this research. The results from which we got the better than their model gave on my dataset. With these promising results, I am confident that my model can be applied to other datasets of tomato dataset and, hopefully, will lead to improved accuracy and performance.

4.9 Summary

In this chapter, I have gone over the dataset and experimental settings, as well as the environment I utilized to apply my suggested models into practice. In my experiments, ResNet50V2's depth improves picture analysis. Shows robust performance across varied datasets. Fine-tuned hyperparameters improve accuracy. Comparative study evaluates efficacy and informs further developments.

4.10 Performance Analysis of Adam optimizer in our model

This Following figure 4.21 graph shows that as the quantity of epochs grows, loss drops, and accuracy rises.

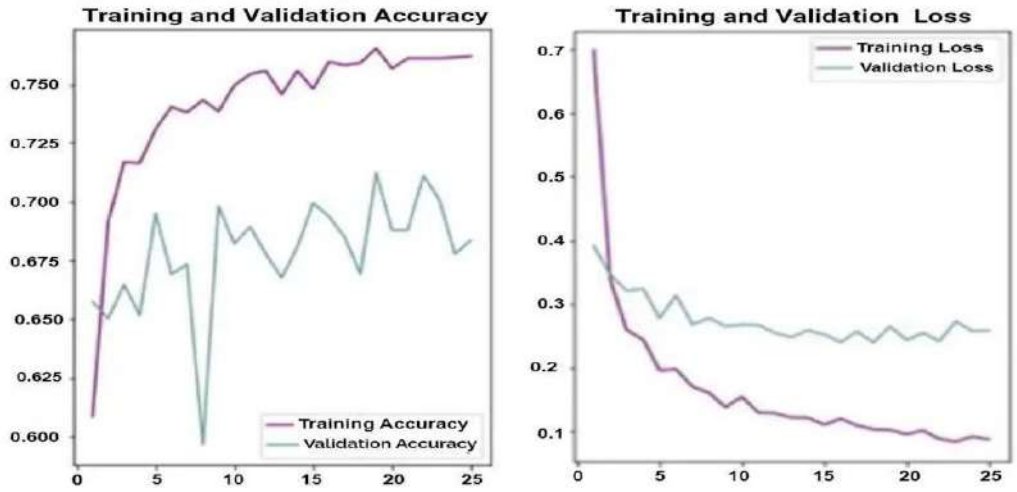


Figure 4.21: Accuracy and Loss curve in Adam Optimizer in 25 epoch

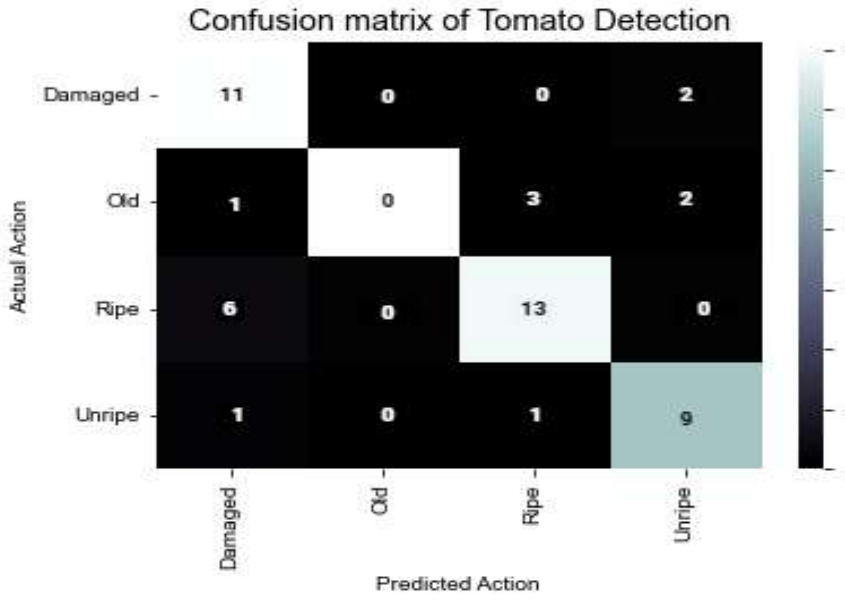


Figure 4.22: Confusion matrix of Adam Optimizer for our model

A confusion matrix is a tool used to visualize the performance of a classification model, where the model's predictions are compared against the actual labels. From the information provided, it appears that as the number of epochs increased, the model's performance improved, as evidenced by the confusion matrix (Figure 4.22) showing better results. The Adam optimizer, a popular optimization algorithm commonly used in training neural networks, contributed to achieving a testing accuracy of 68.00%. complexity of the task and the quality of the dataset.

4.11 Result Analysis of model

Adam performed marginally better when dealing with my model, which has a learning rate of 0.001.

Table 4.5: COMPARISON TABLE OF ABOVE ONE OPTIMIZERS IN OUR MODEL

Optimizer	**Learning Rate**	**Epoch**	**Validation Accuracy** Secondary	**Validation Loss** Secondary	**Validation Accuracy** Primary	**Validation Loss** Primary
Adam	0.001	25	0.9536	0.2965	0.6835	0.3165

4.12 Predicted Sample

Examine an image of tomatoes.ResNet50V2 predicts with 68% accuracy that the image contains tomatoes. Here's an illustration of a predicted sample.

Figure 4.23: Present a visual representation demonstrating the examination of predicted tomato images.

4.13 Primary Data Outcomes

Firstly, I collected primary data(raw data) for testing purpose. My model was able to predict the primary data. I separated into four classes ripe, unripe, damage, old. I collected 49 photos, and our accuracy is 68%.Due to time constraints, I couldn't gather enough old data, impacting accuracy. I plan to address this by working with larger datasets in the future. To compensate for the lack of data, we supplemented with data from Kaggle to complete the thesis.

CAPTER 5

Conclusion and Future Works

5.1 Conclusion

In this paper we have proposed a new modified CNN model ResNet50V2 and collected images for tomato condition detection. When tomato images were used to predict tomato quality like Damaged, old, ripe and unripe, the findings were better and accurate. I have gathered image data from google images and different organization dataset images. Overall, the research reveals that suggested CNN model could be employed for disaster detection and it can predict disasters from images.

5.2 Contribution of this Thesis

The researcher introduced a fresh dataset in this thesis, making major contributions to the field of tomato detection. This dataset is the outcome of combining existing datasets to provide a more diversified and thorough training set for the neural network. The goal of this project is to improve the effectiveness of tomato identification systems by giving a more diverse and representative set of samples for the model to learn from. Secondly, the ResNet50V2 model is modified to specifically tackle challenges related to tomato characteristics. The specific changes made to the ResNet50V2 model greatly improve its accuracy in tomato detection in a variety of scenarios. The effect of adding a flattening layer to the neural network on processing one-dimensional arrays is particularly noteworthy. This well-thought-out inclusion simplifies data representation, enabling learning and optimization that is more efficient. Through particular architecture adaptation to tomato properties, the modified model demonstrates its effectiveness in enhancing tomato detection performance in a variety of settings.

The combined effect of these contributions yields a more robust and adaptable tomato detection method. The improved model can adjust to changing tomato characteristics and environmental conditions, making it useful for precision agriculture applications. The implications extend beyond yield optimization, where precise and flexible tomato detection might be critical for successful crop management. Overall, this study bridges the gap between dataset diversity and model modification by providing practical answers to real-world agricultural difficulties.

5.3 Future works

Some areas of the current research can be examined and improved. The following recommendations are made based on the literature reviews and studies undertaken in this thesis.

This thesis opens avenues for future research, including the exploration of transfer learning techniques and the integration of real-time data augmentation strategies to further improve model generalization. Additionally, collaborative efforts to create and share annotated datasets specific to tomato detection can contribute to advancing the field. In the future, a new classifier with a different optimizer can be introduced for improved accuracy and outcomes. In addition, video recognition will be the primary focus of future study. Besides detecting natural disasters will do image segmentation.

In conclusion, this thesis presents a holistic approach to advancing tomato detection through dataset fusion and modifications to the ResNet50V2 model. The results showcase the potential for enhanced accuracy in tomato detection, contributing to the broader goal of optimizing agricultural practices through technology-driven solutions.

References

[1] G. Zampokas and I. Mariolis, ""Residual Cascade CNN for Detection of Spatially Relevant Objects in Agriculture: The Grape-Stem Paradigm.," International Conference on Computer Vision Systems. Cham: Springer Nature Switzerland, pp. 159-168, 2023.

[2] S. Hassan and A. Maji , "Identification of plant-leaf diseases using CNN and transfer-learning approach," Electronics 10, vol. 12, p. 1388, 2021.

[3] "Fruit Detection for Classification by Type with YNOv3-Based CNN Algorithm."," Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), vol. 4, no. 3, pp. 476-481., 2020.

[4] L. Boukhris and . J. Abderrazak, "Tailored deep learning based architecture for smart agriculture," 2020 International Wireless Communications and Mobile Computing (IWCMC). IEEE, pp. 964-969, 2020.

[5] Alam, Mohammed Jaber, Abdul Gafur, Syed Zahidur Rashid, Md Golam Sadeque, Diponkor Kundu, and Rosni Syed. "Permutation based load balancing technique for long term evolution advanced heterogeneous networks." International Journal of Electrical and Computer Engineering 12, no. 6 (2022): 6311.

[6] Rashid, Syed Zahidur, Abdul Gafur, Aasim Ullah, Md Akbar Hossain, Sultan Shah Mamun, and Md Qudrat-E-Alahi Majumder. "Maximum Power Design and Simulation for a Low Return Loss Wearable Microstrip Patch Antenna." In Towards a Wireless Connected World: Achievements and New Technologies, pp. 161-175. Cham: Springer International Publishing, 2022.

[7] Soumen, Abu Zafar Md Imran, Md Emdadul Hoque Bhuiyan, Subrata Barua, Syed Zahidur Rashid, Abdul Gafur, and Md Razu Ahmed. "Highly efficient microstrip patch antenna for wireless gigabit alliance applications." Indonesian Journal of Electrical Engineering and Computer Science 26, no. 3 (2022): 1451-1459.

[8] "https://www.dataversity.com,"[Online].Available:https://www.dataversity.net/briefhistorydeeplearning/#:~:text=The%20history%20of%20deep%20learning,to%20mimic%20the%20thought%20process..

[9] Chowdhury, Mehedi Hasan, Radowanul Haque, Fahad Uddin, Syed Zahidur Rashid, Md Mohiuddin Soliman, and Abdul Gafur. "A Simple UWB Antenna with WiMAX/WLAN/X Triple Band Rejection Features." In 2022 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp. 56-60. IEEE, 2022.

[10] Islam, Md Tohedul, Abu Zafar Md Imran, Fahim Chowdhury, Md Nur Nobe Hridoy, Md Humayun Kabir, Abdul Gafur, and Syed Zahidur Rashid. "Design and Analysis of Multiband Microstrip Patch Antenna Array for 5G Communications." In 2022 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp. 29-32. IEEE, 2022.

[11] "https://medium.com,"[Online].Available:https://medium.com/@sreyan806/history-of-deep-learning-c176e2d3cddf.

[12] L. Xie and . A. ,Yuille, "Genetic cnn," Proceedings of the IEEE international conference on computer vision, pp. 1379-1388, 2017.

[13] Hossain, Tanvir, Masud Parvez, Abu Zafar Md Imran, Mohammed Jashim Uddin, Abdul Gafur, and Syed Zahidur Rashid. "TeraHertz Antenna for Biomedical Application." In 2022 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp. 42-45. IEEE, 2022.

[14] Samad, Md Abdus, Md Razu Ahmed, and Syed Zahidur Rashid. "An overview of rain attenuation research in Bangladesh." Indonesian Journal of Electrical Engineering and Computer Science 23, no. 2 (2021): 902-909.

[15] D. Bhatt, and C. Patel, , "CNN variants for computer vision: History, architecture, application, challenges and future scope," Electronics 10, vol. 20, p. 2470, 2021.

[16] Syed Zahidur Rashid. “Performance Analysis of DWDM Based Advanced Optical Network System for Different Optical Amplifiers and Channel Configurations.” Daffodil International University (DIU), 2021

[17] "towardsdatascience,"[Online].Available:https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[18] "encord,"[Online].Available:https://encord.com/glossary/pre-trained-model definition/#:~:text=A%20pre%2Dtrained%20model%20is,tuned%20for%20a%20specific%20task.

[19] "nvida," [Online]. Available: https://blogs.nvidia.com/blog/what-is-a-pretrained-aimodel/#:~:text=A%20pretrained%20AI%20model%20is,8%2C%202022%20by%20Angie%20Lee.

[20] Billah, Mohammad Masum, Niaz Ahmed Khan, Mohammad Woli Ullah, Faisal Shahriar, Syed Zahidur Rashid, and Md Razu Ahmed. "Developing a secured and reliable vehicular communication system and its performance evaluation." In 2020 IEEE Region 10 Symposium (TENSYMP), pp. 60-65. IEEE, 2020.

[21] Gafur, Abdul, M. S. Islam, and Syed Zahidur Rashid. "Comparison of coherent optical transmission systems performance by DP-QAM levels." International Journal of Electrical and Computer Engineering 10, no. 3 (2020): 2513.

[22] "wikipedia,"[Online].Available:https://en.wikipedia.org/wiki/Residual_neural_network.

[23] Huque, Saad Mazhurul, Imam Muhammad Amirul Maula, and Syed Zahidur Rashid. "An Advanced Distribution Layer Solution to Improve Bandwidth Utilization and Media Quality for Multi-Access Network Management in Wide Area Network." In 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), pp. 1-4. IEEE, 2019.

[24] "paperswithcode,"[Online].Available:https://paperswithcode.com/method/inceptionresnetv2#:~:text=Inception%2DResNet%2Dv2%20is%20a,of%20Residual%20Connections%20on%20Learning.

[25] Rashid, Syed Zahidur, Abdul Gafur, Atikur Rahaman, Yaheya Solaiman, and Erfanul Hoque Bahadur. "Traffic Load Based Efficient Energy Management Technique for 5G Small Cell Network." In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1-5. IEEE, 2019.

[26] Faisal, Mohammad, Abdul Gafur, Syed Zahidur Rashid, Md OmarFaruk Shawon, Kazi Ishtiak Hasan, and Md Bakey Billah. "Return loss and gain improvement for 5g wireless communication based on single band microstrip square patch antenna." In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1-5. IEEE.

[27] "stackoverflow,"[Online].Available:https://stackoverflow.com/questions/62228981/what-is-freezing-unfreezing-a-layer-in-neural-networks.

[28] ".educative.io," [Online]. Available: https://www.educative.io/answers/what-is-a-neural-network-flatten-layer.

[29] "simplilearn,"[Online].Available:https://www.simplilearn.com/tutorials/deeplearningtutorial/convolutionalneuralnetwork#:~:text=Flattening%20is%20used%20to%20convert,layer%20to%20classify%20the%20image..

[30] "Medium," [Online]. Available: https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnnde73f69c5b83#:~:text=Dense%20Layer%20is%20simple%20layer,on%20output%20from%20convolutional%20layers..

[31] "Kaggle,"[Online].Available:https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook.

[32] "builtin,"[Online].Available:https://builtin.com/machine-learning/reluactivation-function.

[33] "Dotnettutorial," [Online]. Available: https://dotnettutorials.net/lesson/dropout-layer-in-cnn/.

[34] "enjoyalgorithms,"[Online].Available:https://www.enjoyalgorithms.com/blog/how-to-choose-activation-function-for-output-layer.

[35] "enthought,"[Online].Available:https://www.enthought.com/blog/neuralnetwork-output-layer/.

[36] Kabir, Md Humayun, Syed Zahidur Rashid, Abdul Gafur, Muhammad Nurul Islam, and MD Jiabul Hoque. "Maximum energy efficiency of three precoding methods for massive MIMO technique in wireless communication system." In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 1-5. IEEE, 2019.

[37] "d2l.ai,"[Online].Available:https://d2l.ai/chapter_computer-vision/finetuning .html.

[38] Sabbir, Md Maruful Hasan, Md Toukidul Islam, Syed Zahidur Rashid, Abdul Gafur, and Md Humayun Kabir. "An Approach to Performance and Qualitative Analysis of Routing Protocols on IPv6." In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 1-6. IEEE, 2019.

[39] N. G. Raju, A. Therala, V. Yalla, R. R. Ch and K. Rajiv, "Natural Disaster Discernment and Vigilance," in E3S Web of Conferences, 2021.

[40] Mutha, Sahil Amol, Akshat Mayur Shah, and Mohammed Zakee Ahmed. "Maturity detection of tomatoes using deep learning." SN Computer Science 2 (2021): 1-7.

[41] Quach, Luyl-Da, KHANG NGUYEN Quoc, Anh Nguyen Quynh, Nguyen Thai-Nghe, and Tri Gia Nguyen. "Explainable deep learning models with gradient-weighted class activation mapping for smart agriculture." IEEE Access 11, no. August (2023): 83752-83762.

[42] G. Ciocca , and P. Napoletano, "CNN-based features for retrieval and classification of food images.," Computer Vision and Image Understanding, vol. 176, pp. 70-77, 2018

[43] Gafur, Abdul, M. S. Islam, and Syed Zahidur Rashid. "Quality Optimization Based Trend Line for Hybrid Optical Amplifier Configurations in DWDM Transmission Systems." International Journal of Microwave & Optical Technology 14, no. 5 (2019).

[44] He,K.,Zhang,X.,Ren,S.,&Sun,J.(2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision & pattern recognition (pp. 770-778).

[45] MW Ullah, MA Rahman, F Shahriar, ZU Ahmed, MMA Faisal, MJ Uddin and Syed Zahidur Rashid. "Enlightenment of Saint Martin Island: Underwater Submarine Cable and its Reliability." International Conference on Computer and Information Technology (ICCIT), 2018

[46] Kamrul, MD Imtiaz, AZ M. Imran, Abdul Gafur, and Syed Zahidur Rashid. "Rain Attenuation Estimation of Vertical and Horizontal Polarizations for Bangabandhu-1 Satellite." In 2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE), pp. 1-3. IEEE, 2018.

[47] Y. Lee and Y. Kim , "Comparison of CNN and YOLO for Object Detection," Journal of the semiconductor & display technology, vol. 19, no. 1, pp. 85-92, 2019.

[48] R. Girshick , "Fast r-cnn," Proceedings of the IEEE international conference on computer vision, 2015.

[49] Huque, Saad Mazhurul, Imam Muhammad Amirul Maula, Razu Ahmed, Syed Zahidur Rashid, and Abdul Gafur. "An Adaptive Routing Protocol for the Performance of Real-Time Applications." In 2018 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp. 284-289. IEEE, 2018.

[50] J. Wang, and L. Chen, "CNN transfer learning for automatic image-based classification of crop disease," mage and Graphics Technologies and Applications: 13th Conference on Image and Graphics Technologies and Applications, IGTA 2018, Beijing, Chin, vol. 13, pp. 319-329, 2018.

[51] Y. Zhang and Z. Zhao , "Construction of green tea recognition model based on ResNet convolutional neural networ," Journal of Tea Science, pp. 261-271, 2022.

[52] Alam, Towhidul, Chowdhury Mohammad Masum Refat, Abu Zafar Md Imran, Syed Zahidur Rashid, Md Humayun Kabir, Rabiul Hasan Tarek, and Abdul Gafur. "Design and implementation of a secured enterprise network using dynamic multipoint VPN with HSRP protocol." In 2018 International conference on innovations in science, engineering and technology (ICISET), pp. 367-371. IEEE, 2018.

[53] M. Razavi and S. Mavaddati, "ResNet deep models and transfer learning technique for classification and quality detection of rice cultivars," Expert Systems with Applications, vol. 247, p. 123276, 2024.

[54] H. Huynh and Q. Diep , "Analysis and detection of COVID-19 cases on chest X-ray images using a novel architecture self-development deep-learning," 2021 IEEE International Biomedical Instrumentation and Technology Conference (IBITeC), 2021.

[55] F. Saxen and P. Werner , "Face attribute detection with mobilenetv2 and nasnet-mobile.," 2019 11th international symposium on image and signal processing and analysis (ISPA). IEEE,, pp. 176-180, 2019.

[56] Alam, Md Jamshed, MD Imtiaz Kamrul, SM Zia Ur Rashid, and Syed Zahidur Rashid. "An Expert System Based on Belief Rule to Assess Bank Surveillance

Security." In 2018 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp. 451-454. IEEE, 2018.

[57] "A two-stage industrial defect detection framework based on improved-yolov5 and optimized-inception-resnetv2 models.," Applied Sciences, vol. 12, no. 2, p. 834, 2022.

[58] Shahriar, Faisal, S. Newaz, Syed Zahidur Rashid, Mohammad Azazur Rahman, and Muhammad Foyazur Rahman. "Designing a reliable and redundant network for multiple VLANs with Spanning Tree Protocol (STP) and Fast Hop Redundancy Protocol (FHRP)." In Proc. Int. Conf. Ind. Eng. Oper. Manag, vol. 2018, pp. 534-540. 2018.

[59] A. Madhwani and . P. Kumar, "Detection of Breast Cancer from Histopathology Images Using Deep Learning".

[60] Refat, Chowdhury Mohammad Masum, Rabiul Hasan Tarek, Syed Zahidur Rashid, and Abdul Gafur. "Design and Performance Investigation of Campus Area Network (CAN) Based on Different Routing Protocols." 5th INTERNATIONAL CONFERENCE ON NATURAL SCIENCES AND TECHNOLOGY (ICNST'18), Asian University for Women Bangladesh

[61] Syed Zahidur Rashid. “Analysis of Handover Schemes: A Virtual Connection Tree (VCT) Based Wireless Asynchronous Transfer Mode (WATM) Network Approach.”,LAP Publishing

[62] M. Tan and Q. Le , "Efficientnetv2: Smaller models and faster training.," International conference on machine learning,PMLR, pp. 10096-10106, 2021.

[63] MJ Uddin, and Syed Zahidur Rashid. “Performance Analysis of High Frequency BJT and LDMOS Current Mode Class-D Power Amplifier”, International Journal of Scientific & Engineering Research, Volume 4, Issue 6, June-2013

[64] Barua, S., Imran, A. Z. M., Bhuiyan, M. E. H., Barua, S., Rashid, S. Z., Gafur, A., & Ahmed, M. R. (2022). Highly efficient microstrip patch antenna for wireless gigabit alliance applications. Indonesian Journal of Electrical Engineering and Computer Science, 26(3), 1451-1459.

[65] dos Santos LF and dos Santos Canuto JL, "Thermographic image-based diagnosis of failures in electrical motors using deep transfer learning," Engineering Applications of Artificial Intelligence, vol. 126, p. 107106, 2023.

[66] B. Singh and S. Rajak , "Deep Neural Networks for Traffic Sign Recognition Systems," In 2023 6th International Conference on Contemporary Computing and Informatics (IC3I),IEEE., vol. 6, pp. 472-477, 2023.

[67] Hossen, S., Uddin, M. K., Gafur, A., Rashid, S. Z., Hannan, S., & Imran, A. Z. M. (2023, June). Design and analysis of an antenna using ring slotted rectangular reflector with double substrate for 5G mm-wave application. In 2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM) (pp. 1-6). IEEE.

[68] Hossain, T., Chakraborty, S., Uddin, M. J., Gafur, A., Rashid, S. Z., Rahman, M. A., & Ibrahim, M. (2023, June). Performance Enhancement of UWB Antenna Using FSS Layer for millimeter-wave 5G. In 2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM) (pp. 1-6). IEEE.

[69] "Towards real-time action recognition on mobile devices using deep models," arXiv preprint arXiv, p. 1906.07052, 2017.

[70] R. Kanagaraj and E. Elakiya , "Predictive Classification Model of Crop Yield Data Using Artificial Neural Network," 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE., pp. 747-751, 2023.

[71] Ibrahim, Nehad M., Dalia Goda Ibrahim Gabr, Atta-ur Rahman, Sujata Dash, and Anand Nayyar. "A deep learning approach to intelligent fruit identification and family classification." Multimedia Tools and Applications 81, no. 19 (2022): 27783-27798

[72] Ünal, Haci Bayram, Ebru Vural, Burcu Kir Savaş, and Yaşar Becerikli. "Fruit recognition and classification with deep learning support on embedded system (fruitnet)." In 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 1-5. IEEE, 2020.

[73] Gill, Harmandeep Singh, Osamah Ibrahim Khalaf, Youseef Alotaibi, Saleh Alghamdi, and Fawaz Alassery. "Multi-Model CNN-RNN-LSTM Based Fruit Recognition and Classification." Intelligent Automation & Soft Computing 33, no. 1 (2022).

[74] Septiarini, Anindita, Hamdani Hamdani, Sri Ulan Sari, Heliza Rahmania Hatta, Novianti Puspitasari, and Wiwien Hadikurniawati. "Image processing techniques for tomato segmentation applying k-means clustering and edge detection approach." In 2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE), pp. 92-96. IEEE, 2022.

[75] Sudharshan, Duth P., and T. N. Jhansy. "Tomato Fruits Disease Detection Using Image Processing." In 2022 International Conference on Futuristic Technologies (INCOFT), pp. 1-6. IEEE, 2022.

[76] Mureşan, Horea-Bogdan. "An Automated Algorithm for Fruit Image Dataset Building." In 2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS), pp. 103-107. IEEE, 2022.

[77] Legaspi, Jericho, John Raphael Pangilinan, and Noel Linsangan. "Tomato Ripeness and Size Classification Using Image Processing." In 2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pp. 613-618. IEEE, 2022.

[78] Tunio, Muhammad Hanif, Li Jianping, Muhammad Hassaan Farooq Butt, Imran Memon, and Yumna Magsi. "Fruit Detection and Segmentation Using Customized Deep Learning Techniques." In 2022 19th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 1-5. IEEE, 2022.

[79] Nagesh, A. Seetharam, and G. N. Balaji. "Deep Learning Approach for Recognition and Classification of Tomato Fruit Diseases." In 2022 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI), vol. 1, pp. 1-6. IEEE, 2022

[80] Azman, Nur Fitrah, Nor Ashikin Mohamad Kamal, and Norizan Mat Diah. "Tomato Fruit Ripening Classification Using Wavelet-Based Feature Extraction and Multilayer Perceptron." In 2023 IEEE International Conference on Agrosystem Engineering, Technology & Applications (AGRETA), pp. 119- 124. IEEE, 2023.

[81] Hong, Suk-Ju, Seongmin Park, Chang-Hyup Lee, Sungjay Kim, Seung-Woo Roh, Nandita Irsaulul Nurhisna, and Ghiseok Kim. "Application of X-ray 45

[82] Kushwaha, Arvinda. "Fruit Classification Using Optimized CNN." In 2023 International Conference on IoT, Communication and Automation Technology (ICICAT), pp. 1-5. IEEE, 2023.

[83] Mehta, Shiva, Vinay Kukreja, and Rishika Yadav. "A Federated Learning CNN Approach for Tomato Leaf Disease with Severity Analysis." In 2023 Second

International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), pp. 309-314. IEEE, 2023.

[84] Saini, Archana, Kalpna Guleria, and Shagun Sharma. "Tomato Leaf Disease Classification using Convolutional Neural Network Model." In 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), pp. 01-06. IEEE, 2023.

[85] Singh, Utpal Kant, Rajnish Kumar, Saurabh Kumar, Shibasish Kar, and Santos Kumar Baliarsingh. "Detection of Diseases in Tomato Plants using Convolutional Neural Network." In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-6. IEEE, 2023.

[86] Roy, Kyamelia, Sheli Sinha Chaudhuri, Jaroslav Frnda, Srijita Bandopadhyay, Ishan Jyoti Ray, Soumen Banerjee, and Jan Nedoma. "Detection of tomato leaf diseases for agro-based industries using novel PCA DeepNet." IEEE Access 11 (2023): 14983-15001.

[87] H. Yalcin and S. Razavi , "Plant classification using convolutional neural networks," 2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics). IEEE, pp. 1-5, 2016.

[88] A. Kayabasi and A. Toktas , "Automatic classification of agricultural grains: Comparison of neural networks," Neural Network World, vol. 28, no. 3, 2018.

[89] S. Kujawa and G. Niedbała , "Artificial neural networks in agriculture.," Agriculture, vol. 11, no. 6, p. 497, 2021.

[90] A. Gupta and P. Nahar , "Classification and yield prediction in smart agriculture system using IoT," Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 8, pp. 10235-10244, (2023).

[91] A. Hamzah and A. Mohamed , "Classification of white rice grain quality using ANN: a review," IAES International Journal of Artificial Intelligence, vol. 9, no. 4, 2020.

[92] A. W. Whitney, “A direct method of nonparametric measurement selection,” IEEE Transactions on Computers, vol. 100, no. 9, pp. 1100–1103, 1971.

[93] A. Panthakkan and S. Anzar , "Concatenated Xception-ResNet50—A novel hybrid approach for accurate skin cancer prediction," Computers in Biology and Medicine, vol. 150, p. 106170, 2022.

[94] "MendeleyData,"[Online].Available:https://data.mendeley.com/datasets/9zyvdgp83m/1.

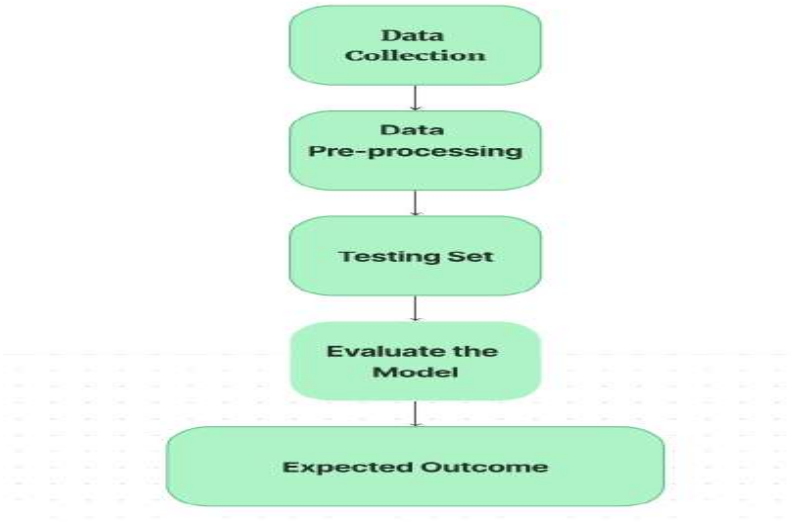
[95] "kaggle," [Online]. Available:https://www.kaggle.com/datasets/enalis/tomatoes-dataset.

[96] L. Quach and K. Quoc , "Explainable deep learning models with gradient-weighted class activation mapping for smart agriculture," IEEE Access 11, pp. 83752-83762., 2023.

APPENDIX

Methodology (For Testing)

For the sake of testing, this is my operational procedure.



Working Flowchart for Testing

Initial Data Gathering

Start by collecting primary data from relevant sources. Ensure the collected data covers the necessary aspects of your project and is of good quality. Organize the data in a structured format for further analysis.

Data Preparation

Clean and preprocess the collected data by addressing any missing values, outliers, or inconsistencies. Standardize or normalize the data to ensure consistency across different features. Convert categorical data into numerical formats if required. Split the preprocessed data into testing sets for model development and evaluation.

Creation of Testing Set

Allocate a portion of the preprocessed data specifically for testing purposes.This segregated dataset will be used to assess the model's performance independently.

Model Evaluation

Choose an appropriate machine learning algorithm based on your problem's nature and data characteristics. Train the selected model using the training dataset. Fine-tune model parameters to optimize its performance. Evaluate the model's effectiveness using various metrics such as accuracy, precision, recall, or mean squared error. Utilize

techniques like cross-validation to validate the model's generalization capabilities.

Analysis of Expected Results

Analyze the performance of the trained model and derive insights from the obtained results. Identify areas for potential enhancement based on the model's performance metrics. Adjust the model architecture or data preprocessing methods as needed. Iterate through the process as required until the desired level of performance is achieved. Anticipate deploying the trained model for making predictions on new, unseen data.

Performance Analysis of Adam optimizer in our model

This Following graph shows that as the quantity of epochs grows, loss drops, and accuracy rises.

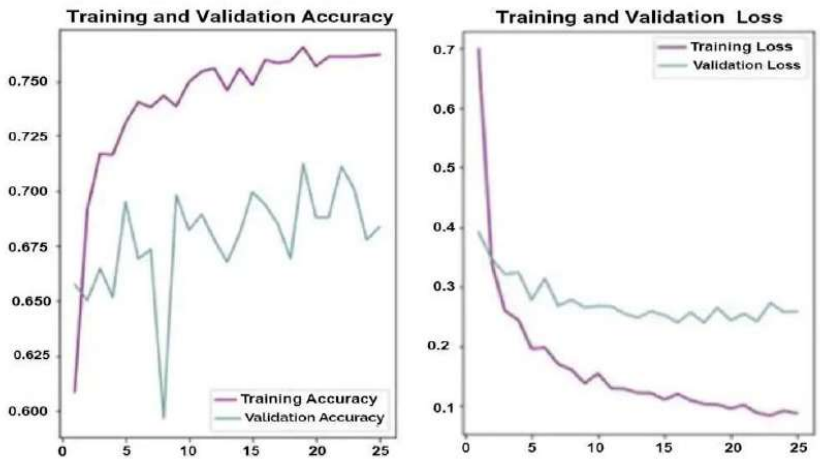


Figure: Accuracy and Loss curve in Adam Optimizer in 25 epoch

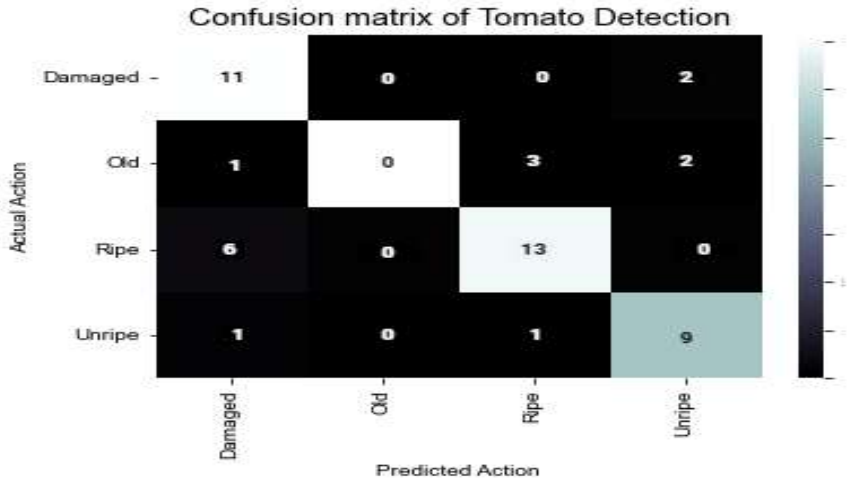


Figure: Confusion matrix of Adam Optimizer for our model

As the number of epochs rose, it is clear from the Figure confusion matrix. Adam optimized in this respect and achieved 68.00% testing accuracy.

Result Analysis of model

Adam performed marginally better when dealing with my model, which has a learning rate of 0.001.

Table : COMPARISON TABLE OF ABOVE ONE OPTIMIZERS IN OUR MODEL:

Optimizer	**Learning Rate**	**Epoch**	**Validation Accuracy** Secondary	**Validation Loss** Secondary	**Validation Accuracy** Primary	**Validation Loss** Primary
Adam	0.001	25	0.9536	0.2965	0.6835	0.3165

Predicted Sample

Examine an image of tomatoes.ResNet50V2 predicts with 68% accuracy that the image contains tomatoes. Here's an illustration of a predicted sample.

Figure : Present a visual representation demonstrating the examination of predicted tomato images.

Primary Data Outcomes

Firstly, I collected primary data(raw data) for testing purposes. My model was able to predict the primary data. I separated into four classes ripe, unripe, damaged, old. I collected 49 photos, and my accuracy is 68%.Due to time constraints, I couldn't gather enough old data, impacting accuracy. I plan to address this by working with larger datasets in the future. To compensate for the lack of data, I supplemented with data from Kaggle to complete the thesis.

The codes that I used in my experiment is given down below:

Primary Model Codes

```
from google.colab import drive
drive.mount('/content/drive')

train_dir = tf.keras.preprocessing.image_dataset_from_directory(
    " /content/drive/MyDrive/ tomatoes_dataset/train",
    seed=123,
    shuffle=True,
    image_size=(224,224),
    batch_size=32
)
classes_tn = train_dir.class_names
classes_tn

val_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "tomatoes_dataset/val",
    seed=123,
    shuffle=False,
    image_size=(224,224),
    batch_size=32
)
classes_vl = val_dir.class_names
classes_vl

test_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "tomatoes_dataset/test",
    seed=123,
    shuffle=False,
    image_size=(224,224),
    batch_size=32
)
image_class = test_dir.class_names
image_class
# Scaling Data
train_dir=train_dir.map(lambda x,y:(x/255,y))
val_dir=val_dir.map(lambda x,y:(x/255,y))
test_dir=test_dir.map(lambda x,y:(x/255,y))

scaled_iterator=train_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()
scaled_iterator=val_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()
scaled_iterator=test_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()
```

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
train_dir = train_dir.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_dir = val_dir.cache().prefetch(buffer_size=AUTOTUNE)
test_dir = test_dir.cache().prefetch(buffer_size=AUTOTUNE)

#Augmentation

aug = tf.keras.Sequential([
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.2),
  layers.RandomZoom(0.2)
 ])

train_dir= train_dir.map(
    lambda x, y: (aug(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)

import tensorflow as tf
tf.get_logger().setLevel('ERROR')
import matplotlib.pyplot as plt

import os
import cv2
import imghdr
import time
import numpy as np
import seaborn as sns

from tensorflow import keras
from tensorflow.keras.preprocessing import  image_dataset_from_directory
from tensorflow.keras import callbacks
from tensorflow.keras import models, layers
from tensorflow.keras.models import Sequential,Model

from tensorflow.keras.applications import MobileNetV2,ResNet50V2,ResNet50
from tensorflow.keras.layers import
experimental,MaxPooling2D,GlobalAveragePooling2D, \
                          Conv2D,BatchNormalization,Dense,Dropout,Flatten

from sklearn.metrics import  accuracy_score, recall_score, precision_score, \
                               f1_score,confusion_matrix,classification_report

ADAM

# compile the model
firstR_model.compile(
  optimizer=keras.optimizers.Adam(learning_rate=1e-3,decay= 1e-3),
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
```

```
  metrics=['accuracy']
)

checkpoint_path = 'Model/check_am/weights-improvement-{epoch:02d}-
{val_accuracy:.2f}.hdf5'
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = callbacks.ModelCheckpoint(checkpoint_path,
                                        monitor='val_accuracy',
                                        verbose=1,
                                        save_best_only=True,
                                        mode='max',
                                        save_freq='epoch')
early_stoping=callbacks.
EarlyStopping(monitor="val_loss",
                                patience=2,
                                verbose=1
                                )

#fitting model

start = time.time()
history = firstR_model.fit(
    train_dir,
    batch_size=32,
    validation_data=val_dir,
    verbose=1,
    epochs=25,
    callbacks = [cp_callback]
)
print("Total time: ",
      time.time() - start,
      "seconds")

scores = firstR_model.evaluate(val_dir)
scores
HISTORY
history.params
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

EPOCHS=range(1, len(acc) + 1)

plt.figure(figsize=(10,6))
plt.subplot(1, 2, 1)
plt.plot(EPOCHS,acc, label='Training Accuracy',color='darkmagenta')
```

```
plt.plot(EPOCHS, val_acc, label='Validation Accuracy',color='cadetblue')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy',fontsize=15)

plt.subplot(1, 2, 2)
plt.plot(EPOCHS,loss, label='Training Loss',color='darkmagenta')
plt.plot(EPOCHS, val_loss, label='Validation Loss',color='cadetblue')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss',fontsize=15)
plt.show()

def actual_predicted_labels(val_dir):

        actual = [labels for _, labels in val_dir.unbatch()]
        prediction = firstR_model.predict(val_dir)

        actual = tf.stack(actual, axis=0)
        predicted = tf.concat(prediction, axis=0)
        predicted = tf.argmax(prediction, axis=1)

        return actual, predicted

def confusion_matrix(actual, predicted,labels=classes_tn):
   cm = tf.math.confusion_matrix(actual, predicted)
   ax = sns.heatmap(cm, annot=True,cmap="bone",cbar=True, fmt='g')
   sns.set(rc={'figure.figsize':(8, 6)})
   sns.set(font_scale=1.2)
   plt.xticks(rotation=90)
   plt.yticks(rotation=0)
   ax.xaxis.set_ticklabels(labels)
   ax.yaxis.set_ticklabels(labels)
   ax.set_title('Confusion matrix of Tomato Detection')
   ax.set_xlabel('Predicted Action')
   ax.set_ylabel('Actual Action')

actual, predicted = actual_predicted_labels(val_dir)
print(confusion_matrix(actual, predicted, classes_tn))
print(classification_report(actual, predicted, target_names=classes_tn))
```

ADAMAX

```
# compile the model
firstR_model.compile(
  optimizer=keras.optimizers.Adam(learning_rate=1e-3,decay= 1e-3),


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

```
)
checkpoint_path = 'Model/check_am/weights-improvement-{epoch:02d}-
{val_accuracy:.2f}.hdf5'
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = callbacks.ModelCheckpoint(checkpoint_path,
                                        monitor='val_accuracy',
                                        verbose=1,
                                        save_best_only=True,
                                        mode='max',
                                        save_freq='epoch')
early_stoping=callbacks.EarlyStopping(monitor="val_loss",
                                      patience=2,
                                      verbose=1
                                      )

#fitting model

start = time.time()
history = firstR_model.fit(
    train_dir,
    batch_size=32,
    validation_data=val_dir,
    verbose=1,
    epochs=25,
    callbacks = [cp_callback]
print("Total time: ",
      time.time() - start,
      "seconds")

scores = firstR_model.evaluate(val_dir)


HISTORY
history.params
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

EPOCHS=range(1, len(acc) + 1)

plt.figure(figsize=(10,6))
plt.subplot(1, 2, 1)
plt.plot(EPOCHS,acc, label='Training Accuracy',color='darkmagenta')
plt.plot(EPOCHS, val_acc, label='Validation Accuracy',color='cadetblue')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy',fontsize=15)


plt.subplot(1, 2, 2)
plt.plot(EPOCHS,loss, label='Training Loss',color='darkmagenta')
plt.plot(EPOCHS, val_loss, label='Validation Loss',color='cadetblue')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss',fontsize=15)
plt.show()

def actual_predicted_labels(val_dir):
actual = [labels for _, labels in val_dir.unbatch()]
        prediction = firstR_model.predict(val_dir)
        actual = tf.stack(actual, axis=0)
        predicted = tf.concat(prediction, axis=0)
        predicted = tf.argmax(prediction, axis=1)
return actual, predicted

def confusion_matrix(actual, predicted,labels=classes_tn):
   cm = tf.math.confusion_matrix(actual, predicted)
   ax = sns.heatmap(cm, annot=True,cmap="bone",cbar=True, fmt='g')
```

```
    sns.set(rc={'figure.figsize':(8, 6)})
    sns.set(font_scale=1.2)
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    ax.xaxis.set_ticklabels(labels)
    ax.yaxis.set_ticklabels(labels)
    ax.set_title('Confusion matrix of Tomato Detection')
    ax.set_xlabel('Predicted Action')
    ax.set_ylabel('Actual Action')

actual, predicted = actual_predicted_labels(val_dir)
print(confusion_matrix(actual, predicted, classes_tn))
print(classification_report(actual, predicted, target_names=classes_tn))
```

Existing Model

```
from google.colab import drive
drive.mount('/content/drive')

import tensorflow as tf
tf.get_logger().setLevel('ERROR')
import matplotlib.pyplot as plt

import os
import cv2
import imghdr
import time
import numpy as np
import seaborn as sns

from tensorflow import keras
from tensorflow.keras import callbacks
from tensorflow.keras import models, layers
from tensorflow.keras.models import Sequential,Model

from tensorflow.keras.applications import MobileNetV2,ResNet50V2,ResNet50
from tensorflow.keras.layers import experimental,MaxPooling2D,GlobalAveragePooling2D, \
                                    Conv2D,BatchNormalization,Dense,Dropout,Flatten

from sklearn.metrics import  accuracy_score, recall_score, precision_score, \
                             f1_score,confusion_matrix,classification_report
```

```
train_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "tomatoes_dataset/train",
    seed=123,
    shuffle=True,
    image_size=(224,224),
    batch_size=32
)
classes_tn = train_dir.class_names
classes_tn

val_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "tomatoes_dataset/val",
    seed=123,
    shuffle=False,
    image_size=(224,224),
    batch_size=32
)
classes_vl = val_dir.class_names
classes_vl

test_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "tomatoes_dataset/test",
    seed=123,
    shuffle=False,
    image_size=(224,224),
    batch_size=32
)
image_class = test_dir.class_names
image_class

train_dir=train_dir.map(lambda x,y:(x/255,y))
val_dir=val_dir.map(lambda x,y:(x/255,y))
test_dir=test_dir.map(lambda x,y:(x/255,y))

scaled_iterator=train_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()
scaled_iterator=val_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()
scaled_iterator=test_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()

AUTOTUNE = tf.data.experimental.AUTOTUNE
train_dir = train_dir.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_dir = val_dir.cache().prefetch(buffer_size=AUTOTUNE)
test_dir = test_dir.cache().prefetch(buffer_size=AUTOTUNE)

aug = tf.keras.Sequential([
```

```
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.2),
  layers.RandomZoom(0.2)
 ])

train_dir= train_dir.map(
    lambda x, y: (aug(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)


plt.figure(figsize=(10, 7))
for images, _ in train_dir.take(3):
    for i in range(6):
        augmented_images = aug(images)
        ax = plt.subplot(2, 3, i + 1)
        plt.imshow(augmented_images[0].numpy())
IMG_SIZE=[224,224]
baseR_model = ResNet50V2(input_shape=IMG_SIZE + [3],
             weights='imagenet',
             include_top=False)

baseR_model.summary()

tf.keras.utils.plot_model(
    baseR_model, to_file='baseR_model.png',show_shapes=True,show_layer_names=True
)
# compile the model
firstR_model.compile(
  optimizer=keras.optimizers.Adam(learning_rate=1e-4,decay= 1e-3),
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
  metrics=['accuracy']
)

checkpoint_path = 'Model/check_A/weights-improvement-{epoch:02d}-{val_accuracy:.2f}.hdf5'
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = callbacks.ModelCheckpoint(checkpoint_path,
                                        monitor='val_accuracy',
                                        verbose=1,
                                        save_best_only=True,
                                        mode='max',
                                        save_freq='epoch')
early_stoping=callbacks.EarlyStopping(monitor="val_loss",
                                      patience=2,
                                      verbose=1
                                      )

#fitting model

start = time.time()
```

```
history = firstR_model.fit(
    train_dir,
    batch_size=32,
    validation_data=val_dir,
    verbose=1,
    epochs=25,
    callbacks = [cp_callback]
)
print("Total time: ",
      time.time() - start,
      "seconds")

scores = firstR_model.evaluate(val_dir)
scores

history
history.params
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

EPOCHS=range(1, len(acc) + 1)

plt.figure(figsize=(10,6))
plt.subplot(1, 2, 1)
plt.plot(EPOCHS,acc, label='Training Accuracy',color='darkmagenta')
plt.plot(EPOCHS, val_acc, label='Validation Accuracy',color='cadetblue')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy',fontsize=15)

plt.subplot(1, 2, 2)
plt.plot(EPOCHS,loss, label='Training Loss',color='darkmagenta')
plt.plot(EPOCHS, val_loss, label='Validation Loss',color='cadetblue')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss',fontsize=15)
plt.show()

def actual_predicted_labels(val_dir):

        actual = [labels for _, labels in val_dir.unbatch()]
        prediction = firstR_model.predict(val_dir)

        actual = tf.stack(actual, axis=0)
        predicted = tf.concat(prediction, axis=0)
        predicted = tf.argmax(prediction, axis=1)
```

```
        return actual, predicted

def confusion_matrix(actual, predicted,labels=classes_tn):
    cm = tf.math.confusion_matrix(actual, predicted)
    ax = sns.heatmap(cm, annot=True,cmap="bone",cbar=True, fmt='g')
    sns.set(rc={'figure.figsize':(8, 6)})
    sns.set(font_scale=1.2)
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    ax.xaxis.set_ticklabels(labels)
    ax.yaxis.set_ticklabels(labels)
    ax.set_title('Confusion matrix of Tomato Detection')
    ax.set_xlabel('Predicted Action')
    ax.set_ylabel('Actual Action')

actual, predicted = actual_predicted_labels(val_dir)
print(confusion_matrix(actual, predicted, classes_tn))
print(classification_report(actual, predicted, target_names=classes_tn))
```

ADAMAX

```
# compile the model
firstR_model.compile(
  optimizer=keras.optimizers.Adam(learning_rate=1e-3,decay= 1e-3),
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
  metrics=['accuracy']
)

checkpoint_path = 'Model/check_AdmxB/weights-improvement-{epoch:02d}-
{val_accuracy:.2f}.hdf5'
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = callbacks.ModelCheckpoint(checkpoint_path,
                                        monitor='val_accuracy',
                                        verbose=1,
                                        save_best_only=True,
                                        mode='max',
                                        save_freq='epoch')
early_stoping=callbacks.EarlyStopping(monitor="val_loss",
                                      patience=2,
                                      verbose=1
                                      )
#fitting model

start = time.time()
history = firstR_model.fit(
    train_dir,
    batch_size=32,
    validation_data=val_dir,
```

```
    verbose=1,
    epochs=25,
    callbacks = [cp_callback]
)
print("Total time: ",
      time.time() - start,
      "seconds")

scores = firstR_model.evaluate(val_dir)
scores

history
history.params
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

EPOCHS=range(1, len(acc) + 1)

plt.figure(figsize=(10,6))
plt.subplot(1, 2, 1)
plt.plot(EPOCHS,acc, label='Training Accuracy',color='darkmagenta')
plt.plot(EPOCHS, val_acc, label='Validation Accuracy',color='cadetblue')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy',fontsize=15)

plt.subplot(1, 2, 2)
plt.plot(EPOCHS,loss, label='Training Loss',color='darkmagenta')
plt.plot(EPOCHS, val_loss, label='Validation Loss',color='cadetblue')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss',fontsize=15)
plt.show()

def actual_predicted_labels(val_dir):

        actual = [labels for _, labels in val_dir.unbatch()]
        prediction = firstR_model.predict(val_dir)

        actual = tf.stack(actual, axis=0)
        predicted = tf.concat(prediction, axis=0)
        predicted = tf.argmax(prediction, axis=1
        return actual, predicted

def confusion_matrix(actual, predicted,labels=classes_tn):
    cm = tf.math.confusion_matrix(actual, predicted)
    ax = sns.heatmap(cm, annot=True,cmap="bone",cbar=True, fmt='g')
```

```
    sns.set(rc={'figure.figsize':(8, 6)})
    sns.set(font_scale=1.2)
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    ax.xaxis.set_ticklabels(labels)
    ax.yaxis.set_ticklabels(labels)
    ax.set_title('Confusion matrix of Tomato Detection')
    ax.set_xlabel('Predicted Action')
    ax.set_ylabel('Actual Action')

actual, predicted = actual_predicted_labels(val_dir)
print(confusion_matrix(actual, predicted, classes_tn))
print(classification_report(actual, predicted, target_names=classes_tn))
```

DEMO CODE

```
from google.colab import drive
drive.mount('/content/drive')

import tensorflow as tf
tf.get_logger().setLevel('ERROR')
import matplotlib.pyplot as plt

import os
import cv2
import imghdr
import time
import numpy as np
import seaborn as sns

from tensorflow import keras
from tensorflow.keras import callbacks
from tensorflow.keras import models, layers
from tensorflow.keras.models import Sequential,Model

from tensorflow.keras.applications import MobileNetV2,ResNet50V2,ResNet50
from tensorflow.keras.layers import experimental,MaxPooling2D,GlobalAveragePooling2D, \
                                    Conv2D,BatchNormalization,Dense,Dropout,Flatten

from sklearn.metrics import  accuracy_score, recall_score, precision_score, \
                             f1_score,confusion_matrix,classification_report
#directory

train_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Colab Notebooks /tomatoes_datasets/train",
    seed=123,
    shuffle=True,
```

```
    image_size=(224,224),
    batch_size=32
)
classes_tn = train_dir.class_names
classes_tn

val_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Colab Notebooks/Ifat_bappy/tomatoes_datasets/val",
    seed=123,
    shuffle=False,
    image_size=(224,224),
    batch_size=32
)
classes_vl = val_dir.class_names
classes_vl

test_dir = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/Colab Notebooks/Ifat_bappy/tomatoes_datasets/sample_test",
    seed=123,
    shuffle=False,
    image_size=(224,224),
    batch_size=64
)
image_class = test_dir.class_names
image_class


# Scaling Data
train_dir=train_dir.map(lambda x,y:(x/255,y))
val_dir=val_dir.map(lambda x,y:(x/255,y))
test_dir=test_dir.map(lambda x,y:(x/255,y))

scaled_iterator=train_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()
scaled_iterator=val_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()
scaled_iterator=test_dir.as_numpy_iterator()
label_batch=scaled_iterator.next()

AUTOTUNE = tf.data.experimental.AUTOTUNE
train_dir = train_dir.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_dir = val_dir.cache().prefetch(buffer_size=AUTOTUNE)
test_dir = test_dir.cache().prefetch(buffer_size=AUTOTUNE)

#Augmentation

aug = tf.keras.Sequential([
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.2),
```

```
  layers.RandomZoom(0.2)
 ])

train_dir= train_dir.map(
    lambda x, y: (aug(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)

plt.figure(figsize=(10, 7))
for images, _ in train_dir.take(3):
    for i in range(6):
        augmented_images = aug(images)
        ax = plt.subplot(2, 3, i + 1)
        plt.imshow(augmented_images[0].numpy())
IMG_SIZE=[224,224]
baseR_model = ResNet50V2(input_shape=IMG_SIZE + [3],
             weights='imagenet',
             include_top=False)

baseR_model.summary()

tf.keras.utils.plot_model(
    baseR_model, to_file='baseR_model.png',show_shapes=True,show_layer_names=True
)
AccuracyVector = []
plt.figure(figsize=(30, 20))
for images, labels in test_dir.take(1):
    predictions = firstR_model.predict(images)
    predlabel = []
    prdlbl = []

    for mem in predictions:
        predlabel.append(image_class[np.argmax(mem)])
        prdlbl.append(np.argmax(mem))

    AccuracyVector = np.array(prdlbl) == labels
    for i in range(30):
        ax = plt.subplot(3, 10, i + 1)
        plt.imshow(images[i].numpy())
        plt.title('Pred: '+ predlabel[i]+ '\n'+ ' actual:'+image_class[labels[i]] )
        plt.axis('off')
AccuracyVector = []
plt.figure(figsize=(25, 25))
for images, labels in test_dir.take(1):
    predictions = firstR_model.predict(images)
    predlabel = []
    prdlbl = []

    for mem in predictions:
```

```
    predlabel.append(image_class[np.argmax(mem)])
    prdlbl.append(np.argmax(mem))

AccuracyVector = np.array(prdlbl) == labels
for i in range(49):
    ax = plt.subplot(7, 7, i + 1)
    plt.imshow(images[i].numpy())
    plt.title('Pred: '+ predlabel[i]+ '\n'+ ' actual:'+image_class[labels[i]] )
    plt.axis('off')
    plt.grid(True)
```