RipeVision.AI: Development of Tomato Maturity Level Prediction Software Using Advanced Artificial Intelligence Techniques

A Dissertation Report

submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Agricultural Engineering

By

Sumedha Koranga : 54798

Under the Supervision of

Dr. N. C. Shahi

DEPARTMENT OF POST HARVEST PROCESS AND FOOD ENGINEERING
COLLEGE OF TECHNOLOGY
G. B. PANT UNIVERSITY OF AGRICULTURE AND TECHNOLOGY
UTTARAKHAND, INDIA

JUNE, 2023

APPROVAL

The dissertation entitled **RipeVision.AI: Development of Tomato Maturity Level Prediction Software Using Advanced Artificial Intelligence Techniques** submitted by **Sumedha Koranga (Id. No. 54798)** is hereby approved.

Dr. N. C. Shahi
Professor and Project Advisor _____

Dr. N. C. Shahi
Professor and Head _____

Department of Post Harvest Process and Food Engineering
College of Technology
G. B. Pant University of Agriculture and Technology
Uttarakhand, India

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to **Dr. N. C. Shahi** for his invaluable guidance and support throughout the course of this project. His expertise in the field of post-harvest, specifically fruit and vegetable ripeness detection, and his insightful suggestions greatly contributed to the success of this research endeavor. His unwavering encouragement and constant availability for discussions have been instrumental in shaping this project.

I would also like to extend my heartfelt appreciation to **Dr. N. C. Shahi** for his role as the department head. His visionary leadership and administrative support provided a conducive environment for carrying out this project. His commitment to academic excellence and his dedication to fostering research initiatives have been an inspiration to me and many others.

I am also grateful to **Dr. Anil Kumar**, Coordinator, Agricultural Engineering, **Dr. Devendra Kumar**, Former Coordinator, Agricultural Engineering, **Dr. Triveni Prasad Singh**, Professor and Head, FMPED, **Dr. Harish Chandra**, Professor, IDED, **Dr. P. S. Kashyap**, Professor, SWCED and **Dr. Sachin Kumar**, Assistant Professor, PHPFED, members of the project evaluation committee.

In conclusion, this project would not have been possible without the guidance and support of **Dr. N. C. Shahi**, my project advisor, as well as his role as the department head. I am truly grateful for their contributions, and I am confident that the knowledge and skills gained during this project will have a lasting impact on my academic and professional journey.

Sumedha Koranga
54798

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

1. INTRODUCTION

The major issue with the fruit industry is to ensure that the produce in the market is at an appropriate maturity level. The produce should not be overripe so they rot quickly, neither should they be underripe that they won't sell. The major issue with the fruit industry is to ensure that the produce in the market is at an appropriate level of quality. This is a huge concern when exporting this produce to overseas markets. This issue frequently arises as a result of insufficient or even excessive fruit maturity at the appropriate time of harvest. Fruit ripeness is typically determined by a number of factors, including size, weight, color, fruit scent, etc. Fruit ripeness characteristics can be determined by the color of the fruit's skin, which is both a significant factor and an accessible method **(Taofik *et al.*, 2018)**. The importance of size and color in determining the stage of fruit maturity is the primary motivation of this project in using Computer Vision and artificial intelligence techniques, specifically convolutional neural networks, to develop an easy-to-use and scalable application for determining tomato maturity stage.

According to estimates, India produced more than 20 million metric tonnes of tomatoes in the fiscal year 2022 **(A. Minhas, 2022)**. Approximately 16% of these tomatoes are wasted due to rotting as suggested by the Central Institute of Post-Harvest Engineering and Technology (CIPHET) **(Sharma, 2023)**. Various industries, such as agriculture, food processing, and quality control, depend greatly on the evaluation of fruit ripeness. To ensure optimal harvesting, minimize waste, and provide consumers with high-quality food, precise and prompt identification of tomato maturity is crucial.

In several industries, including agriculture, food processing, and quality control, it is extremely difficult to determine when tomatoes are ripe. Visual inspection is a time-consuming, subjective, and error-prone traditional technique of determining tomato ripeness. This makes it difficult for efficient harvesting, sorting, and grading procedures to take place, which results in less-than-optimal resource utilization and consequent losses for farmers and distributors.

In addition, a reliable and automated method for determining tomato maturity is required due to the rising demand for high-quality products and the requirement to satisfy consumer preferences. Although existing methods, such as spectroscopy and image processing, have demonstrated promise in obtaining objective data, they still have drawbacks like costly equipment, difficult calibration procedures, and scaling problems.

The following are the objectives of the project:

(i) To develop an accurate, successful, and user-friendly tomato ripeness detection model and application.

(ii) To make use of artificial intelligence and computer vision techniques to automate the evaluation of tomato ripeness based on visual traits.

(iii) To give farmers, distributors, and consumers a trustworthy tool to make decisions about harvesting, sorting, and purchasing.

2. REVIEW OF LITERATURE

In this chapter, a comprehensive literature review is presented, examining the maturity level of tomatoes and other fruits and exploring their correlation with advanced artificial intelligence techniques.

A lot of recent research has been carried out using artificial intelligence techniques in an effort to speed up processing and generate reliable and accurate results. Scientists have employed computer vision and digital image processing extensively as methods to address issues in agriculture **(Chen *et al.*, 2002)**. Applications for computer vision systems range from regular inspection to computer vision system-guided robotic control, and they have demonstrated success in the objective of online measurement of numerous agricultural and food goods **(Brosnan and Sun, 2004)**. A lot of recent research has been carried out using artificial intelligence techniques in an effort to speed up processing and generate reliable and accurate results. Scientists have employed computer vision and digital image processing extensively as methods to address issues in agriculture **(Chen *et al.*, 2002)**. Applications for computer vision systems range from regular inspection to computer vision system-guided robotic control, and they have demonstrated success in the objective of online measurement of numerous agricultural and food goods **(Brosnan and Sun, 2004)**.

Heron and Zachariah, (1974). utilized a color CCD camera to capture images of fruits. The camera outputted color information in three channels: red, green, and blue (RGB). To better represent the subjective color sensation of humans, the RGB images were transformed into the HSV color space, and a new color space conversion technique was introduced to simplify color grading. The result was a one-dimensional array that represented different color levels, known as the dominant colors of fruits. This representation reduced computation consumption significantly **(Huang *et al.*, 2018)**. To distinguish between the various tomato ripening phases and to forecast their hardness, a nondestructive approach for evaluating tomato quality was devised. The multi-attribute information fusion decision model was built using computer vision, an electronic nose, and data fusion technologies. Support vector classifier (SVC) and Fisher discriminant analysis (FDA) models for tomato grading were developed. For forecasting the hardness, support vector regression (SVR) and partial least squares (PLS) models were both employed. Better results were obtained with the combined system that incorporated computer vision and an electronic nose than with the single detection technique **(Lu *et al.*, 2016)**.For evaluating tomato ripeness without causing damage,

visible/near-infrared spectroscopy and machine vision have been examined. Application of chemometrics techniques based on partial least squares (PLS) regression allowed for the evaluation of the relationship between spectral wavelengths and green grayscale value.

A new system that can recognize the four stages of fruit ripening in tomato and chili fruits was created to operate on a mobile device. Using a novel method, training data are acquired. The training data was gathered from unbiased observation of the same tomato and chili fruit from one month before harvest till the time of harvest. Fuzzy logic is used for ripeness detection while the K-Means Clustering Method is used for image segmentation **(Taofik *et al.*, 2018)**.

Cho *et al.*, (2021) introduced a prediction system that uses a combination of deep neural networks (DNNs) and machine learning (ML) to automatically determine the stage of fruit ripening, combining them to optimize various image datasets. The color feature vectors most suited for identifying them were first extracted from the observed photos denoting each stage of ripening using eight DNN algorithms. Second, they used seven ML techniques to identify the fruit's ripening stage from the retrieved color data. Third, they presented a system that uses a combination of DNN and ML techniques to automatically predict the level of ripeness in photos of various fruits, including strawberries and tomatoes.

Minagawa and Kim *et al.*, (2022) provided a novel technique to forecast when specific tomato fruits will be ready for harvest. To identify tomato bunches, the method employs Mask R-CNN, and to determine the freshness of individual tomato fruits, it employs two different methods. The results of the experiment demonstrated that when predicting the harvesting time of tomatoes, the accuracy of the prediction using the ratio of R values was better, and when predicting the harvesting time of tomatoes, the accuracy of the prediction using the average of the differences between R and G in RGB values was better.

Sandra *et al.*, (2020) introduced a database method based on the RGB digital picture and vitamin C content is utilized to forecast tomato ripeness. 30 pieces of unripe, medium-ripe, and ripe tomatoes were employed in the performance test. The outcome demonstrates that the tomato's various stages of development have varying RGB values. As the ripeness of the tomato increases, the Green value drops. Because tomatoes have a red index from the start of their ripening, red values frequently remain constant.

3. MATERIALS AND METHODS

This chapter describes the methodology and the approach taken towards the project.

3.1 Dataset

There are a total of 2,036 photos in this collection of tomatoes divided into four maturity states: Unripe, Ripe, Rejected, and Damaged **Khang *et al.*, (2023)**. Figure 3.1 shows some sample images taken from the dataset. The dataset has been split into a training set and a validation set in an 8:2 ratio, with 80% of the images used for training and the remaining 20% used for validation. To detect the condition of tomatoes, in particular, this dataset can be utilized for picture classification tasks. The pictures came from a variety of sources and were carefully chosen to represent a variety of tomato states.

The Tomato Ripeness Detection app (RipeVision.AI) analyses tomatoes to assess their levels of ripeness using the RGB color model. Widely employed in digital imaging, the RGB color model expresses colors by fusing distinct red, green, and blue light intensities.

The image of a tomato that a user takes using the app is displayed as a matrix of pixels, each of which contains RGB values. The RGB values, which range from 0 to 255, show how intensely red, green, and blue light are present in each pixel.

The deep learning algorithm utilizes the RGB values of the tomato image as key input elements. To determine the color and level of ripeness of the tomato, the algorithm examines the distribution and patterns of RGB values throughout the image. Changes in color in the ripeness of tomatoes might be a sign of various ripeness degrees. For instance, mature tomatoes have a redder or orange color than unripe tomatoes, which typically have a greener hue. The system can identify these color fluctuations and forecast the degree of ripeness by looking at the RGB values of the tomato image.

3.2 Deep Learning

3.2.1 Convolutional neural network - computer vision

For its computer vision skills, the Tomato Ripeness Detection app uses convolutional neural networks (CNNs). To effectively assess the levels of tomato maturity, CNNs are essential

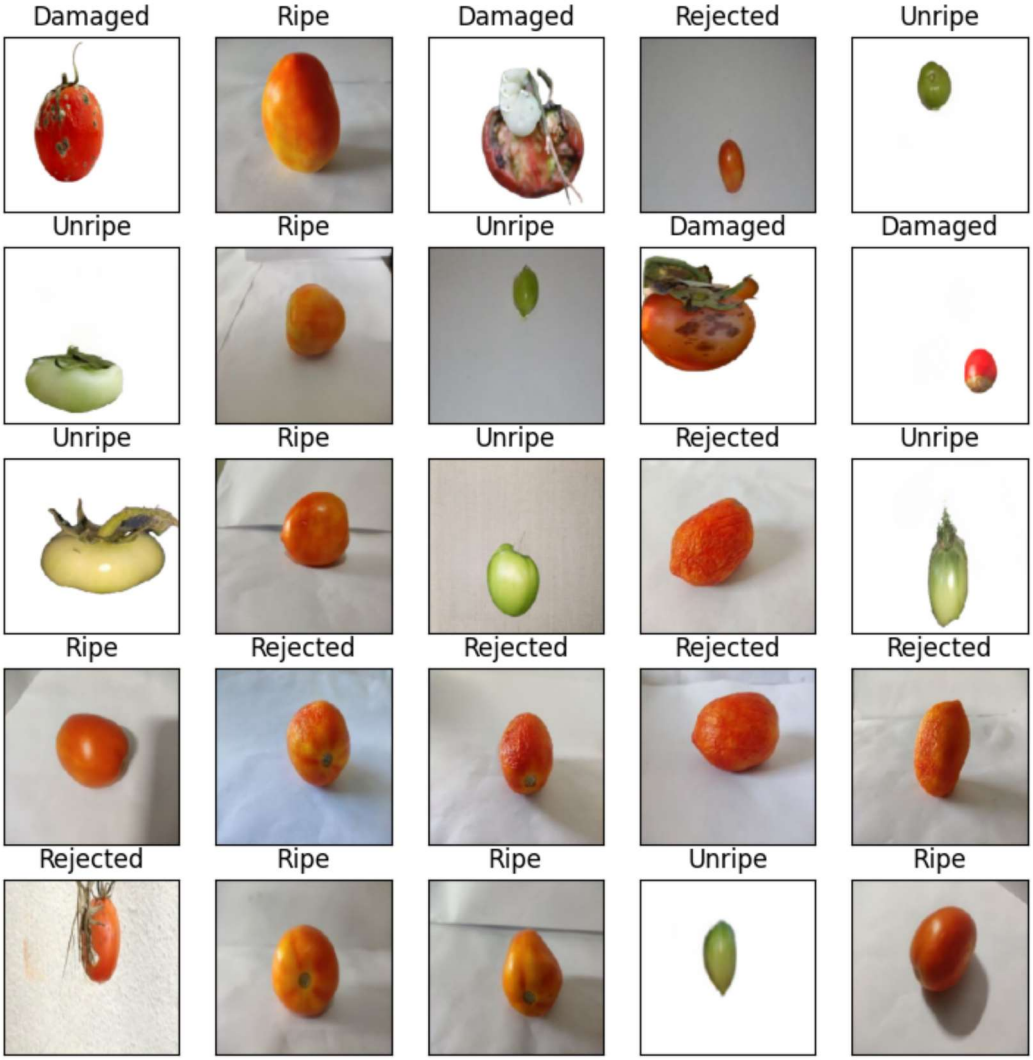

Figure 3.1: Tomatoes Dataset

for analyzing picture data. The image of the tomato is put into CNN for processing when a user scans it using the app. A sizable dataset of labeled tomato photos is used to train the CNN, where each labeled tomato image corresponds to a certain level of ripeness.

A number of convolutional layers, pooling layers, and fully linked layers make up CNN's architecture. Together, these layers help us identify important elements in the tomato photos. Convolutional layers apply filters to the input image in order to capture several kinds of visual patterns, including edges, textures, and color gradients. The model architecture of the neural network used in the project is described in Section 3.5.

The pooling layers downsample the features as the image travels through the network, bringing down the spatial dimensions while retaining the crucial data. This procedure aids in making the previously learned traits more universal and resistant to minute spatial variations.

The output of the convolutional and pooling layers is then flattened and fed through fully connected layers. Based on the learned traits, these layers carry out high-level reasoning

and decision-making. With the Tomato Ripeness Detection app (RipeVision.AI), CNN can forecast the tomato's level of ripeness in the image because of the fully connected layers.

The app's CNN was trained using a broad dataset of tomato photos with various degrees of ripeness. The network gains the ability to identify patterns and features in the photos that correspond with various degrees of ripeness during the training phase. As a result, CNN is able to forecast the degree of tomato ripeness without ever seeing the tomatoes.

The Tomato Ripeness Detection app (RipeVision.AI) offers an automatic and impartial approach to determining tomato freshness by utilizing CNNs. The CNN's capacity to identify significant details in images and generate precise predictions improves the app's usability and functionality. It helps prevent food loss and maintain high quality by enabling growers, distributors, and customers to rapidly and accurately assess the ripeness of tomatoes.

Convolution is the outcome of applying a filter to an input and activating it. When the same filter is applied repeatedly to an input, a feature map is created, representing the positions and strength of a recognized feature in the input, such as an image. Convolution, like a typical neural network, is a linear operation that includes multiplying a set of weights with the input.

For high-dimensional input data, like the speech data of our project, ANN is unable to extract the best features and is prone to overfit the data. Convolutional neural networks solve this problem as they take care of feature selection and they generally require a lesser number of neurons. Convolution involves sliding the kernel over the input signal, which is also known as the shift-compute procedure. The multiplication is done between an array of input data and a two-dimensional array of weights, called a filter or a kernel because the approach was created for two-dimensional input. The filter is smaller than the input data, and the dot product is used to multiply a filter-sized patch of the input with the filter. It is intentional to use a filter that is smaller than the input because it allows the same filter (set of weights) to be multiplied by the input array several times at different points on the input. Specifically, the filter is applied sequentially to each overlapping section or filter-sized patch of the incoming data, left to right, top to bottom. This notion of applying the same filter to an image in a systematic way is a great one. If the filter is designed to detect a specific type of feature in the input, then applying it systematically throughout the entire image gives the filter a chance to find that feature anywhere in the image. Because the filter is applied to the input array several times, the outcome is a two-dimensional array of output values that indicate input filtering. As such, the two-dimensional output array from this procedure is called a "feature map".

3.2.2 Mathematical and technical discussion of different deep learning layers

3.2.2.1 Mathematical understanding of convolutional layer (1D)

The aforementioned process of convolution can be explained mathematically as follows, Let,

Input = x

Length of input = n

Kernel = h

Kernel size = k

Stride = s

Then, the output y can be defined as

$$y(n) = \begin{cases} \sum_{i=0}^{k} x(n+i)h(i) & n = 0 \\ \sum_{i=0}^{k} x(n+i+(s-1))h(i) & otherwise \end{cases} \tag{3.1}$$

For example, if $n = 5$, $k = 3$ and $s = 1$ then,

$$y(0) = x(0)h(0) + x(1)h(1) + x(2)h(2)$$

$$y(1) = x(1)h(0) + x(2)h(1) + x(3)h(2)$$

$$y(2) = x(2)h(0) + x(3)h(1) + x(4)h(2)$$

The length of the output is o can be computed by,

$$o = \lfloor (n-k).s \rfloor + 1 \tag{3.2}$$

This is true if we provide the padding argument to "valid" in Tensorflow Keras. If we want the output length to be equal to input length, i.e., $o = n$ then we can provide the padding argument to "same", this adds 0s at the start and end of the input such that the output length is equal to n.

3.2.2.2 Pooling

Pooling is used to reduce the dimensionality of a given mapping while highlighting the prominent features. Generally, pooling is employed after the convolution layer to reduce the dimension of the convolution output. Pooling also helps to reduce overfitting. We use max-pooling in our model, which refers to the technique of picking up the max value from each window of size f which slides over the input with stride s.

3.2.2.3 Flatten

Traditional neural network layers, called Dense layers or ANN layers, take input of dimension one where the output of convolution layers usually has a depth of greater than one. Therefore, we flatten the output of the convolutional layer to make it compatible with upcoming neural network layers.

3.2.2.4 Dense

A dense layer in a neural network is one that is deeply connected to the layer before it, meaning that the layer's neurons are connected to every neuron in the layer before it. In a model, the dense layer's neuron receives input from every neuron in the preceding layer, and the dense layer's neurons conduct matrix-vector multiplication. The row vector of the output from the preceding layers is identical to the column vector of the dense layer in matrix-vector multiplication. In matrix-vector multiplication, the row vector must have the same number of columns as the column vector.

Let,

X = Input
W_i = Weights corresponding to layer i
B_i = Bias corresponding to layer i
f_i = Activation function for layer i

Then, the output Y of a neural network with 3 layers is

$$Y = f_3(f_2(f_1(X * W_1 + B_1) * W_2 + B_2) * W_3 + B_3) \tag{3.3}$$

For our neural network, we have used the categorical cross-entropy loss function, which is effective in multi-class classification tasks. This loss is a perfect measure of how distinguishable two discrete probability distributions are from each other.

$$Loss = -\sum_{i=1}^{outputsize} y_i . \log \hat{y}_i \tag{3.4}$$

where y_i is the ith scalar value in the model output y_i is the corresponding target value and output size is the number of scalar values in the model output.

In this context y_i is the probability that an event occurs, and the sum of all is 1, meaning that exactly one event may occur. The negative sign ensures that the loss gets smaller when the distributions get closer to each other.

Optimization algorithms update the network weights iteratively based on training data. We use the Adam optimization algorithm to train our neural network model.

3.3 Transfer Learning

Transfer learning is a method that is frequently utilized in computer vision tasks as well as machine learning and deep learning. It entails using the information and learning representations from previously trained models to address brand-new or connected issues. Transfer learning is used in the Tomato Ripeness Detection app (RipeVision.AI) to improve the efficiency of the tomato ripeness detection algorithm. The software uses pre-trained models that have been learned on a large-scale dataset, generally for a related task like image classification, rather than developing a deep learning model from scratch using a vast collection of tomato photos.

To apply transfer learning, the Tomato Ripeness Detection app (RipeVision.AI) takes the pre-trained model and fine-tunes it on the dataset described in Section 3.1. Some additional dense layers are added to adapt the model to the tomato ripeness detection task. The project has two significant advantages in utilizing transfer learning. Since the model doesn't need to be developed from scratch, it first saves time and computational resources. Second, the app makes use of the learned representations from the pre-trained model, which has already been trained to recognize important aspects for visual recognition tasks. Transfer learning enhances the app's performance by leveraging pre-trained models and fine-tuning them for tomato ripeness detection. The MobileNet architecture explained in Section 3.3.1, optimizes computational resources, enabling a real-time analysis on mobile devices with limited processing power.

3.3.1 MobileNet

MobileNet **(Howard *et al.*, 2017)** is a popular deep-learning architecture that is well-suited for computer vision tasks, including tomato ripeness detection. It is designed to be efficient and lightweight, making it particularly suitable for resource-constrained environments such as mobile devices or embedded systems. MobileNet is based on a concept called depthwise separable convolution, which aims to reduce computational complexity while maintaining good accuracy. It achieves this by decomposing the standard convolution operation into two separate layers: a depthwise convolution and a pointwise convolution. A visualization of the architecture of the MobileNet model is given in Figure 3.2 The depthwise convolution applies a single convolutional filter to each input channel independently, while the pointwise convolution performs a 1x1 convolution to combine the outputs of the depthwise convolution. This approach significantly reduces the number of parameters and operations, leading to faster inference and lower memory requirements. One of the key advantages of using MobileNet for the tomato ripeness project is its efficiency in terms of both computational resources and model size. This makes it well-suited for deployment on resource-constrained devices, enabling real-time inference on edge devices or in embedded systems. The lightweight nature

of MobileNet allows for the efficient processing of tomato images, even in scenarios with limited computational power or memory constraints.

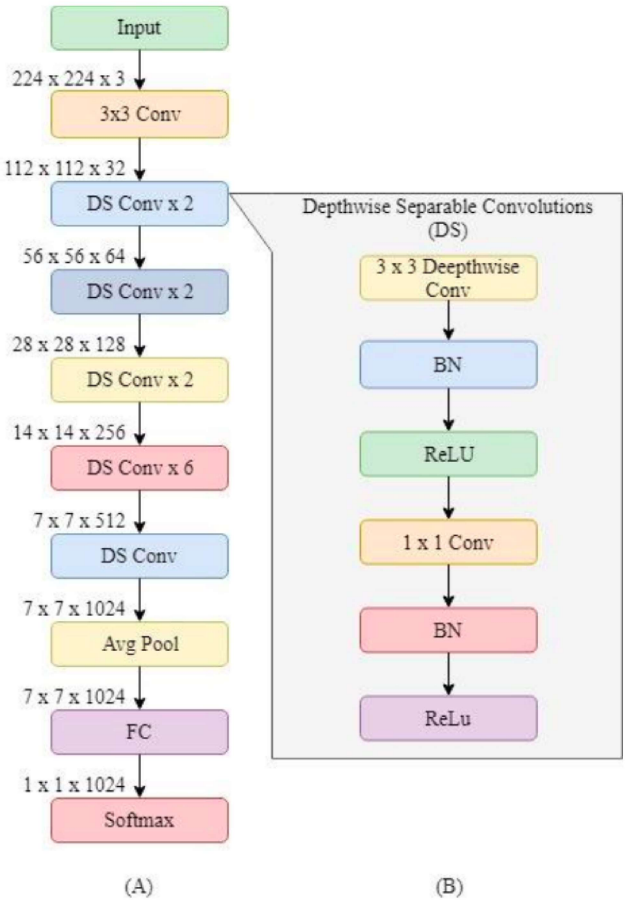

Figure 3.2: MobileNet Architecture (A) The overall architecture and (B) an in-depth explanation of the Depthwise Separable layer

Furthermore, MobileNet's architecture has been widely adopted and extensively trained on large-scale datasets, making it a reliable choice for transfer learning. Transfer learning involves leveraging a pre-trained model on a different but related task and fine-tuning it on the specific task at hand. By utilizing pre-trained MobileNet models, you can benefit from the general image recognition capabilities learned from large-scale datasets, such as ImageNet, and then fine-tune the model using your tomato ripeness dataset. This approach can lead to faster convergence and improved performance, even with limited training data. MobileNet's ability to balance computational efficiency with reasonable accuracy makes it an excellent choice for real-time tomato ripeness detection applications. It can effectively classify the ripeness of tomatoes while also allowing for efficient deployment on various devices. By leveraging MobileNet's capabilities, you can build a robust and efficient tomato ripeness detection system that can operate in real-world scenarios, such as sorting or quality control in agricultural or food processing industries. MobileNet is used as the basic convolutional

neural network architecture for tomato ripening detection in the Tomato Ripeness Detection app (RipeVision.AI). On mobile devices with constrained memory and processing power, MobileNet enables the software to carry out real-time tomato ripeness analysis.

MobileNet achieves its efficiency through a technique called depth-wise separable convolutions. Traditional convolutional layers have a high computational cost since only one filter is applied to each input channel. MobileNet, in comparison, reduces the number of computations while retaining the ability to capture pertinent characteristics by applying distinct filters to each input channel and combining the results. The Tomato Ripeness Detection app (RipeVision.AI) benefits from using MobileNet in a number of ways. To begin with, it provides real-time performance on portable devices, ensuring prompt and effective evaluations of tomato ripeness. For on-the-go applications, such as farmers or distributors evaluating tomato quality in the field or on-site, this is very advantageous.

3.4 Data Augmentation

Data augmentation is a method for generating new training data out of old training data. To do this, domain-specific approaches are applied to instances from the training data to produce brand-new and distinctive training examples. Shifts, flips, zooms and many more picture alteration techniques are included in the category of transforms.

3.4.1 Horizontal and vertical shift augmentation

When a picture is shifted, all of its pixels are moved uniformly in one direction—such as horizontally or vertically—while maintaining its original dimensions. This implies that some pixels will be removed from the image and that new pixel values will need to be given for a portion of the image.

The degree of horizontal and vertical shift is controlled by the `width_shift_range` and `height_shift_range` arguments to the `ImageDataGenerator` constructor, respectively. These arguments allow for the specification of a floating point number that represents the amount (between 0 and 1) of the image's width or height that should be shifted. The image can also be moved by a defined number of pixels. For each image, a value between no shift and the percentage or pixel value will be sampled, and the shift will then be applied.

3.4.2 Horizontal and vertical flip augmentation

A vertical or horizontal image flip, respectively, involves flipping the rows or columns of pixels. A boolean `horizontal_flip` or `vertical_flip` option to the `ImageDataGenerator` class constructor specifies the flip augmentation.

3.4.3 Random rotation augmentation

The image is randomly rotated from 0 to 360 degrees clockwise by a rotation augmentation. We must supply the `rotation_range` argument to the constructor of the `ImageDataGenerator` class in order to use it. The rotation will rotate some pixels out of the image frame, leaving blank spaces that must be filled in using the `fill_mode` argument.

3.4.4 Random brightness augmentation

Images can be randomly darkened, brightened, or both to increase the brightness of the image. The goal is to enable generalization across photos trained on various illumination levels for a model. The `ImageDataGenerator` constructor's `brightness_range` option, which sets the minimum and maximum range as a float indicating a percentage for choosing a brightening amount, can be used to do this. Values below 1.0 make the image darker, such as [0.5, 1.0], while values above 1.0 make the image brighter, such as [1.0, 1.5], where 1.0 has no impact on brightness.

3.4.5 Random zoom augmentation

A zoom augmentation randomly enlarges the image and either interpolates or adds new pixel values around the image, as appropriate. The `zoom_range` argument to the `ImageDataGenerator` constructor can be used to customize image zooming. The percentage of the zoom can be specified as a single float or a range can be specified as an array or tuple.

3.5 Model Architecture

The Tomato Ripeness Detection app (RipeVision.AI) employs the MobileNet architecture, as described in Section 3.3.1, for training the tomato ripeness classification model. In the model, an average pooling layer is incorporated as a critical component. Average pooling is a pooling operation commonly used in convolutional neural networks (CNNs) to downsample feature maps and reduce spatial dimensions. Dense layers are utilized as a crucial component. The model uses the LeakyReLU activation function. It has a tiny slope for negative values as opposed to a flat slope of ReLU activation function. Figure 3.3 shows the LeakyReLU activation function.

The output layer's activation function is softmax. In multi-class classification tasks, where the objective is to assign probabilities to numerous mutually exclusive classes, softmax is a prominent activation function. Softmax activation function is shown in Figure 3.4. The optimizer used during the training process is Adam. Adam, short for Adaptive Moment Estimation, is a popular optimization algorithm commonly used in deep learning models.

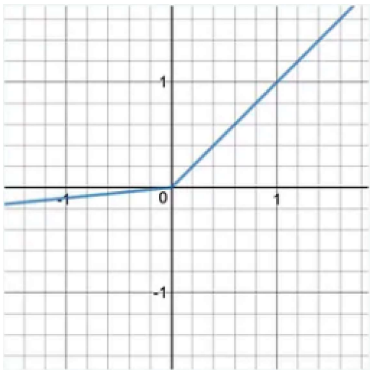

Figure 3.3: Leaky ReLU Activation Function

The approach is "computationally efficient, requires little memory, is invariant to gradient diagonal rescaling, and is well suited for problems with large amounts of data and parameters" **(Kingma and Ba *et al.*, 2017)**.

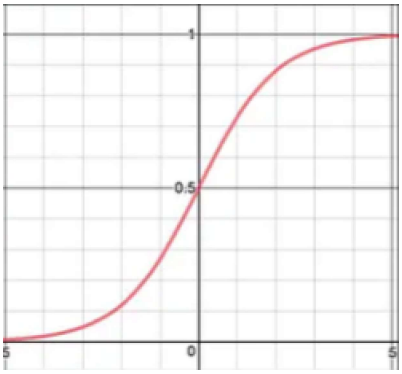

Figure 3.4: Sigmoid Activation Function

We evaluate the model on two metrics: Accuracy and AUC (area under the ROC curve). The standard way to evaluate any classification model is to measure the accuracy of any classification machine learning model is to count the number of test data points that were correctly categorised along with the total number of test data points. AUC is measured by calculating the area under the receiver operating characteris (ROC) curve. The ROC curve is essentially a plot of True Positive Rate vs. False Positive Rate. The ratio of negative events that were misclassified is known as the false positive rate. It is also known as the Specificity and has the definition of 1-True Negative Rate were misclassified is known as the false positive rate. It is also known as the Specificity and has the definition of 1-True Negative Rate. The more the AUC approaches 1, the better the classification were carried out.

We train three models to use in our android application. The model hyperparameters are given in Table 3.1. The first model is trained on the dataset without any data augmentations. The training of the second model utilized the following data augmentations: Rotation,

Brightness, Zoom and Flip. The dataset after applying these augmentations is shown in Figure 3.5. The third model in addition to the previous augmentation utilizes Shift and Shear too. Figure 3.6 shows the dataset after applying these data augmentations.

Table 3.1: Model Hyperparameters

Hyperparameter	Value
Batch Size	128
Image Size	(224, 224, 3)
Optimizer	Adam
Learning Rate	0.001

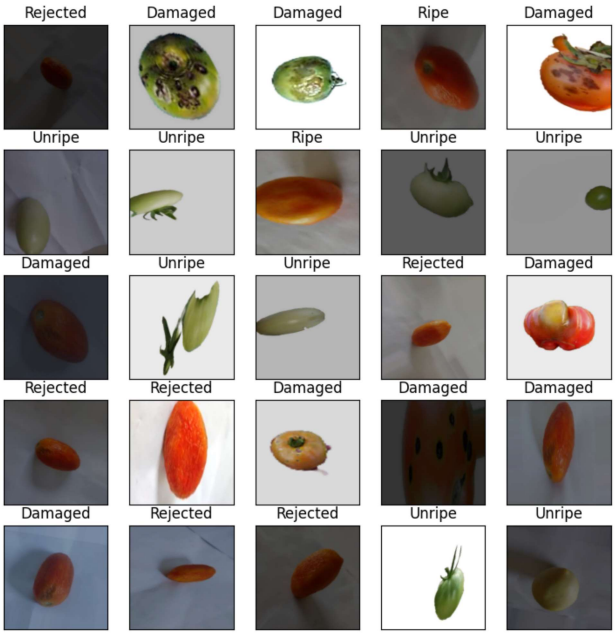

Figure 3.5: Images after Data Augmentations: Rotation, Brightness, Zoom and Flip

3.6 Android Application

RipeVision.AI is an innovative Tomato Ripeness Detection app that I have developed, which utilizes advanced deep learning techniques to classify the ripeness level of tomatoes accurately. By leveraging deep learning algorithms, RipeVision.AI achieves high accuracy by analyzing tomato images with CNNs trained explicitly on a large dataset. I used TensorFlow Lite **(Abadi *et al.*, 2015)** to integrate powerful machine learning capabilities into my application; it allows us to deploy and run machine learning models efficiently on mobile

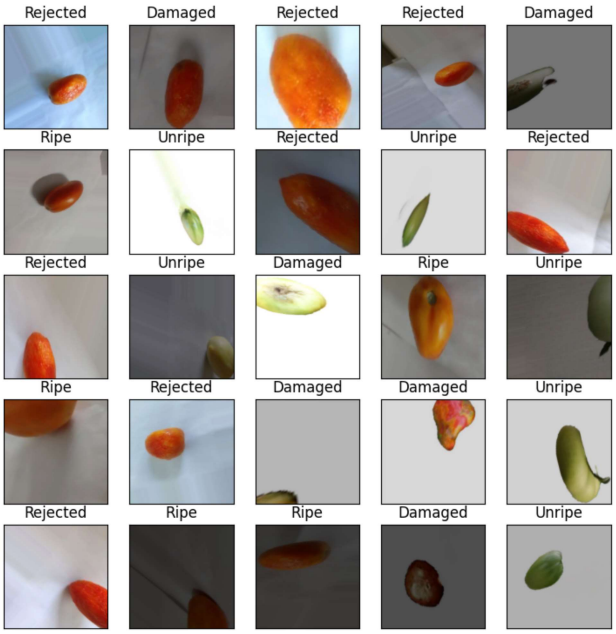

Figure 3.6: Images after Data Augmentations: Rotation, Brightness, Zoom, Flip, Shift and Shear

devices. TensorFlow Lite supports multiple platforms, including Android, iOS, and embedded systems. This cross-platform compatibility allows us to deploy our app on a wide range of devices, ensuring broad availability and maximum user reach.

The application is developed using Kotlin. It is a modern and versatile programming language that enhances our app development process. Kotlin, with its concise syntax and powerful features, provides several benefits for creating robust and efficient Android applications. The app provides accurate and real-time assessment of tomato ripeness, aiding farmers, distributors, and consumers in making informed decisions about the quality and readiness of tomatoes. The app incorporates various features to enhance its functionality and user experience.

In Figure 3.7, the app interface of the Tomato Ripeness Detection app (RipeVision.AI) is presented, showcasing the user-facing visual design and functionality of the app. The app interface serves as the user's gateway to interact with the app's features and functionalities. It is carefully designed to provide a seamless and intuitive user experience, enabling users to effortlessly navigate through the various screens and access the desired functionalities.

The various options presented in the application interface are explained below:

Scanner: The Tomato Ripeness Detection app (RipeVision.AI) incorporates a powerful scan feature that allows users to assess the ripeness of tomatoes with ease and precision. The scan feature utilizes the device's camera to capture images of tomatoes, enabling real-time analysis and providing instant feedback on their ripeness levels.

Confidence level: The confidence level is a quantitative measure that indicates how

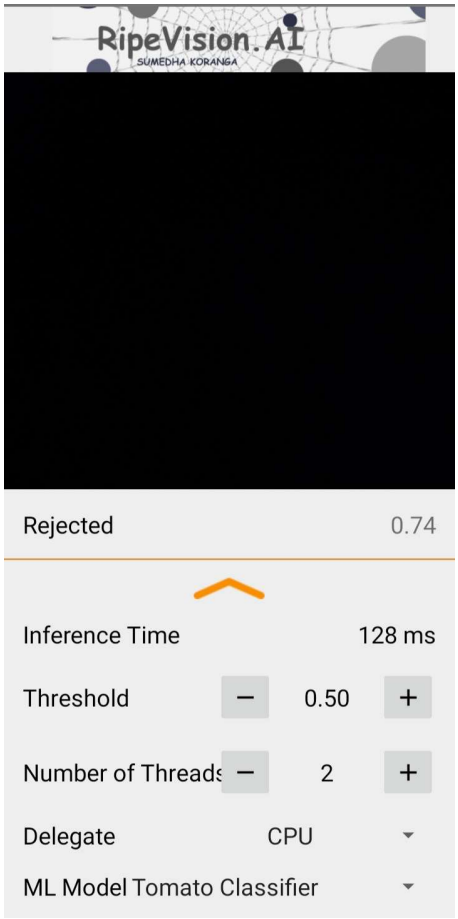

Figure 3.7: Android Application Interface

certain the app's analysis is regarding the tomato's maturity level. It is represented as a percentage, reflecting the app's confidence in its assessment. This provides users with valuable insights into the reliability and accuracy of the app's results. For example, if the app determines that a tomato is ripe and assigns it a confidence level of 90%, it signifies a high level of certainty in the analysis. Users can have confidence in the app's assessment and trust the recommendation provided.

Inference Time: Inference time refers to the time it takes for the Tomato Ripeness Detection app (RipeVision.AI) to predict the maturity level of a tomato using the trained model. It is a crucial performance metric that determines the speed and responsiveness of the app's ripeness assessment process. When a user captures an image of a tomato using the app's scanning feature, the captured image undergoes a series of processing steps, including preprocessing, feature extraction, and classification. The trained model analyzes the image's visual features and predicts the tomato's maturity level based on its color, texture, and shape characteristics. The inference time represents the duration from the moment the image is inputted into the model until the app produces the prediction or result. It is typically measured in milliseconds and directly impacts the user experience. A shorter inference time means

that users receive the ripeness assessment results more quickly, enabling more seamless and efficient user interaction.

Threshold: The threshold refers to the minimum confidence level required for the app to make a prediction regarding the maturity level of a tomato. It is a configurable parameter that helps determine the level of certainty needed for the app to provide a conclusive assessment. The threshold acts as a cutoff point, defining the minimum confidence level that the app considers acceptable for a prediction to be made. If the confidence level of a prediction exceeds the specified threshold, the app will provide the corresponding maturity level classification (e.g., unripe, ripe, damaged, rejected) along with any associated recommendations. However, if the confidence level falls below the threshold, the app may choose to withhold a prediction or indicate a lower confidence level for the result.

Setting an appropriate threshold is important to strike a balance between accuracy and the app's tolerance for uncertainty. A higher threshold ensures that predictions are made with a higher degree of confidence, potentially reducing the likelihood of false positives or misleading assessments. On the other hand, a lower threshold allows for predictions with lower confidence levels, which may increase the number of predictions made but also introduce the possibility of less reliable results.

No. of Threads: The number of threads refers to the concurrent execution units utilized by the Tomato Ripeness Detection app (RipeVision.AI) during the inference process. When the app performs inference on a tomato image, it splits the workload into multiple threads, allowing different parts of the computational tasks to be executed simultaneously. This parallelization of tasks enables the app to distribute the processing load across multiple CPU cores or threads, harnessing the full potential of the device's processing power by utilizing more threads. The app can leverage the available CPU resources more efficiently, leading to faster execution of the inference process. This results in reduced inference time, allowing users to obtain ripeness assessment results more quickly.

Delegate: The delegate refers to a computational unit that assists in the execution of tasks related to tomato ripeness assessment. Similar to the way CPUs (Central Processing Units) and GPUs (Graphics Processing Units) handle different types of computations, delegates in RipeVision. AI can be utilized to offload specific tasks and enhance the overall performance of the app.

CPU Delegate: The CPU delegate represents the primary processing unit of the device. It is responsible for general-purpose computations and can perform a variety of tasks related to the app's functionality, including pre-processing of tomato images, executing the trained machine learning model, and performing post-processing tasks on the inference results. The CPU delegate is typically used when the computational requirements are moderate or when specific tasks are better suited for CPU-based processing.

GPU Delegate: The GPU delegate refers to the Graphics Processing Unit, which is spe-

cialized for handling highly parallelizable computations. GPUs excel in tasks that can be divided into numerous smaller jobs that can be carried out at once. In the context of RipeVision.AI, the GPU delegate can be utilized for accelerating certain computations, such as image processing and inference calculations. By leveraging the GPU's parallel processing capabilities, the app can achieve faster execution and reduced inference time, particularly for tasks that benefit from parallelization.

ML Models: The Tomato Ripeness Detection app (RipeVision.AI) utilizes three trained ML models for tomato ripeness classification. These models are designed to improve the accuracy and robustness of ripeness assessment in various conditions. Each model is trained with a different level of data augmentation to explore the impact of augmentation techniques on tomato ripeness detection. By comparing the performance of these models, the app aims to determine the most effective approach for accurate and reliable tomato ripeness classification.

Tomato Classifier: This model serves as the baseline for tomato ripeness detection. It is trained on a dataset without any data augmentation techniques applied. While this model provides a starting point for ripeness assessment, it may be more sensitive to variations in lighting conditions, angles, and other factors present in real-world tomato images.

Tomato Classifier Pro: The second trained model applies a lighter level of data augmentation compared to the heavy augmentation model. It strikes a balance between the baseline model and the heavy augmentation model. This approach incorporates some data augmentation techniques but to a lesser extent. The light data augmentation model aims to improve generalization while maintaining a certain level of sensitivity to the original tomato image characteristics.

Tomato Classifier Pro 2: The third trained model incorporates a heavy data augmentation approach. It is trained on a dataset enriched with a wide range of data augmentation techniques, such as geometric transformations, color space transformations, random erasing, kernel filters, and image mixing. This augmentation strategy helps the model to generalize better to unseen tomato images, handle different lighting conditions, and overcome potential challenges associated with variations in angles and other factors. The heavy data augmentation model aims to improve the robustness and accuracy of tomato ripeness detection in various scenarios.

4. RESULTS AND DISCUSSION

4.1 Performance of Deep Learning Models

In this section, I will outline the performance of the three deep learning models during the training process. For each of the deep learning models, we have built graphs depicting how the metrics and loss compare between the training and validation datasets at each epoch.

4.1.1 Model without data augmentation

First, I trained the model described in Section 3.5 without using any data augmentations. Figures 4.1, 4.2 and 4.3 show the loss, accuracy, and AUC score comparisons for the training and validation subsets. This model was trained for 37 epochs with the early stopping on the validation loss, which was reduced to the minimum value of 0.2188 as compared to 0.0244 for the training subset loss.

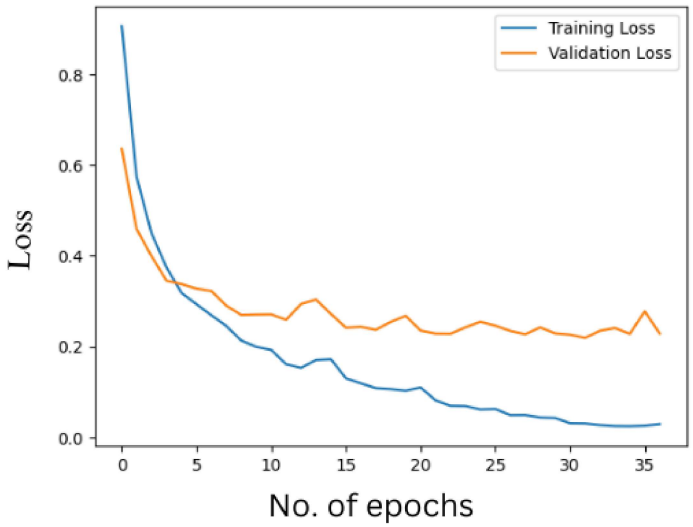

Figure 4.1: Model without Data Augmentation: Loss

4.1.2 Model with light data augmentation

For the second model, I employed a limited amount of data augmentation strategies. Specifically, rotation, brightness, zoom, and flip. Similar to the previous section, Figures 4.4, 4.5

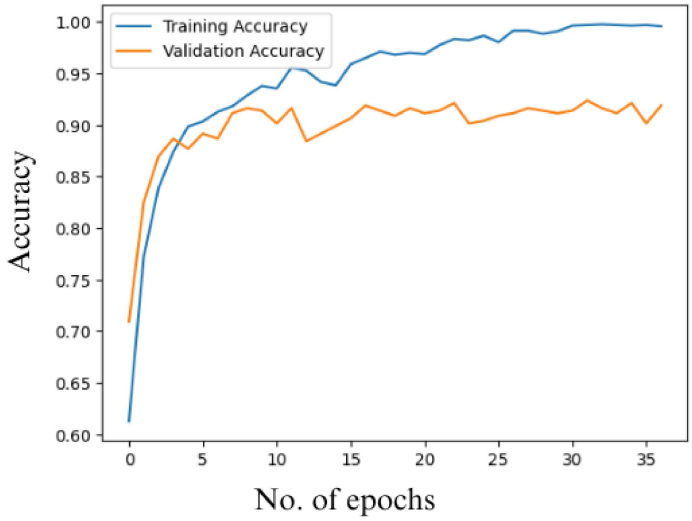

Figure 4.2: Model without Data Augmentation: Accuracy

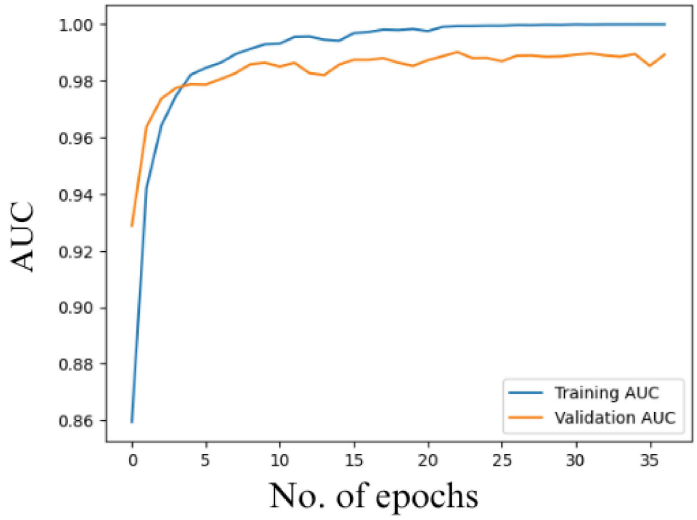

Figure 4.3: Model without Data Augmentation: AUC

and 4.6 show the loss, accuracy, and AUC score comparisons during the epochs. This model converged in 20 epochs. For this and the other data augmentation model, described in the next section, the augmentation techniques were not employed on the validation subset.

4.1.3 Model with heavy data augmentation

The last deep learning model was trained by employing shear and shift in addition to the other data augmentation techniques of the previous model. The comparison graphs of loss, accuracy and AUC scores for this model are shown in Figures 4.7, 4.8 and 4.9. This model only took 16 epochs to converge to the validation loss minima. As expected this model showed the most difference in the performance on the augmented training subset and the

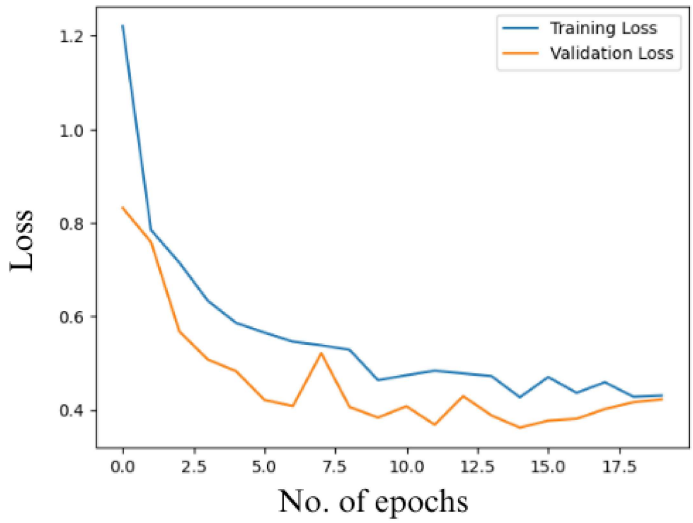

Figure 4.4: Model with Light Data Augmentation: Loss

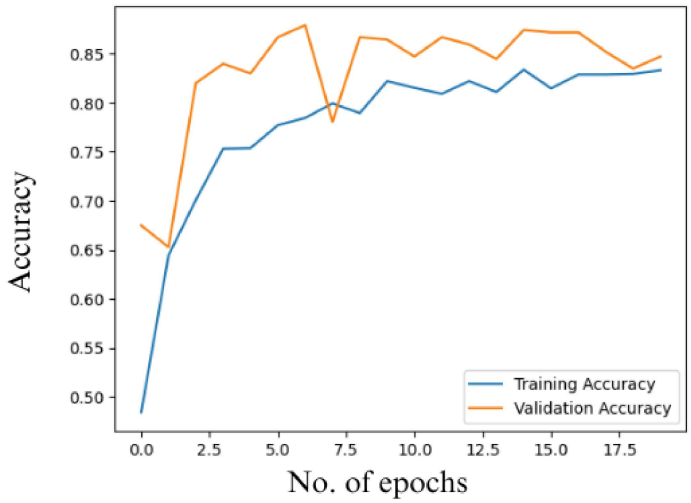

Figure 4.5: Model with Light Data Augmentation: Accuracy

unaltered validation subset.

4.1.4 Model comparison

In this section, I will give a comparison of the performance of the above three models on both the training and validation subsets. The performance is compared across the same three metrics: cross-entropy loss, accuracy, and AUC. The performance comparison is given in Table 4.1.

From Table 4.1, we can see that our transfer learning model achieved very high accuracy and AUC when the data augmentation strategies were not used. This suggests that the model architecture sufficiently learned the features required for the classification of the tomato in

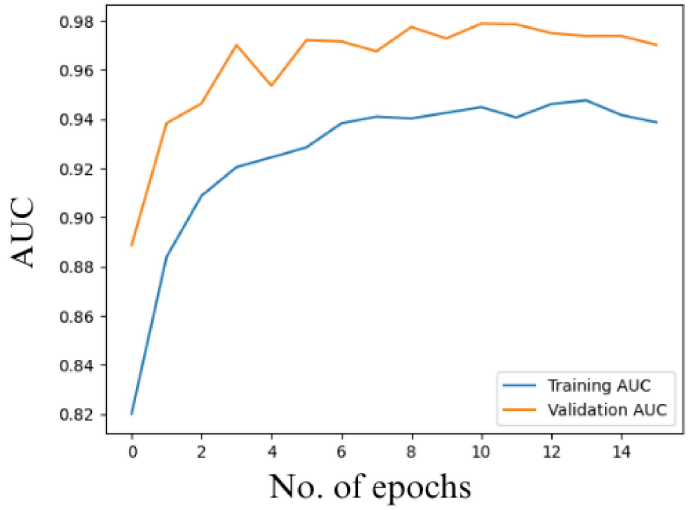

Figure 4.6: Model with Light Data Augmentation: AUC

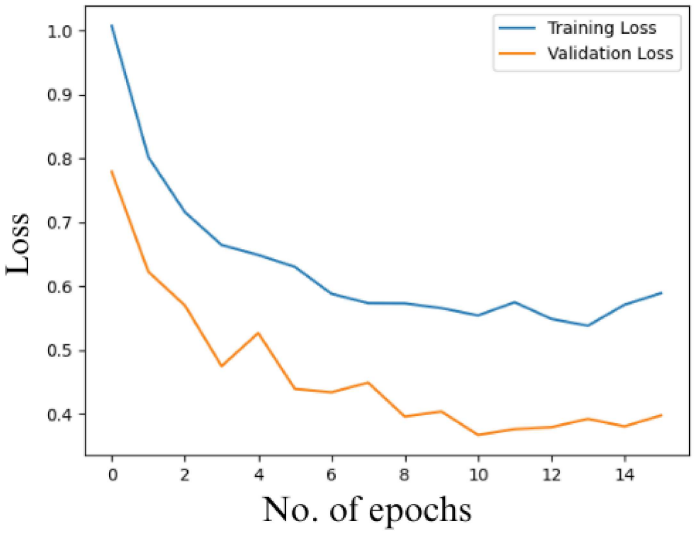

Figure 4.7: Model with Heavy Data Augmentation: Loss

Table 4.1: Comparison of the three models

Model	**Training set**			**Validation set**		
	Loss	**Acc.**	**AUC**	**Loss**	**Acc.**	**AUC**
Model without Data Augmentation	0.0303	0.9969	0.9999	0.2188	0.9236	0.9897
Model with Light Data Augmentation	0.4271	0.8338	0.9674	0.3621	0.8744	0.9767
Model with Heavy Data Augmentation	0.5538	0.7809	0.9449	0.3668	0.8892	0.9788

maturity stages. The two data augmentation models gave similar performance on the validation dataset but there is a significant decrease in the training subset performance as expected.

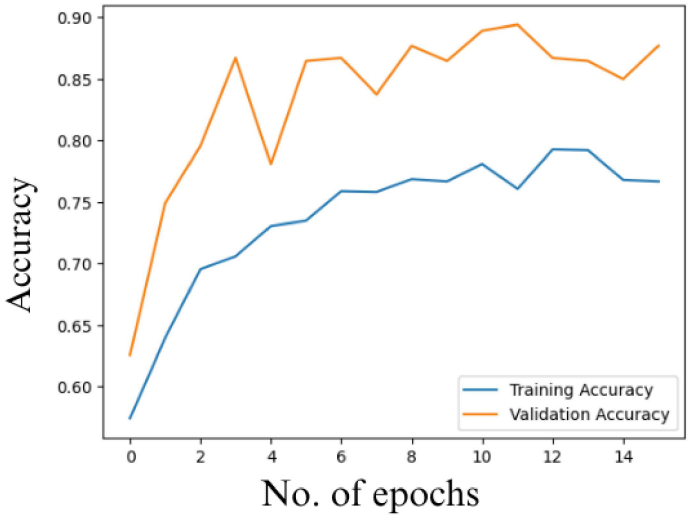

Figure 4.8: Model with Heavy Data Augmentation: Accuracy

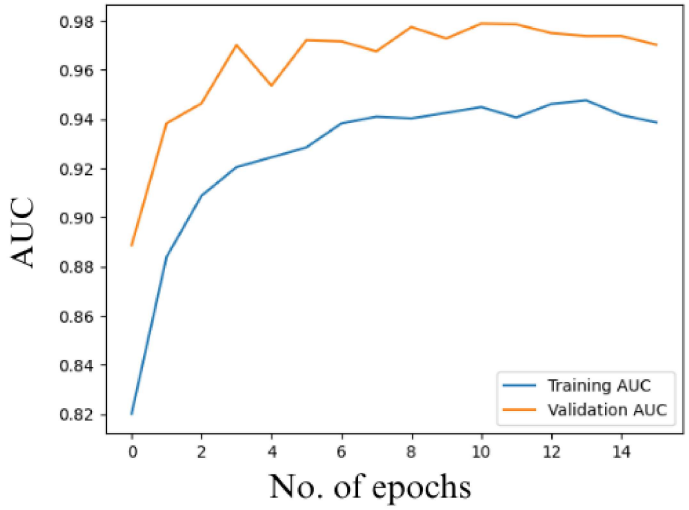

Figure 4.9: Model with Heavy Data Augmentation: AUC

4.2 Android App Performance

In this section, I will show the performance of the Android application, RipeVision.AI, on some randomly selected tomatoes. Figure 4.10 represents a model's confidence level of 0.97 regarding the ripeness of a tomato. A confidence level of 0.97 indicates a high level of certainty in the model's prediction and that the model is highly confident that the tomato is indeed ripe.

Figure 4.11 shows the model prediction on an old rotten tomato. The Android application classifies the tomato as rejected with a confidence of 0.97. This suggests that the model is 97% confident that the tomato shown in the camera belongs to the rejected category and is not fit for consumption.

Figure 4.10: Android Application: Ripe

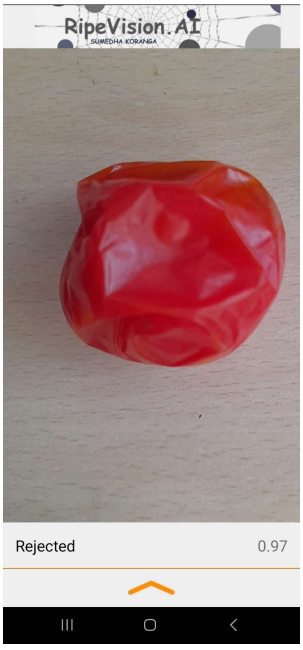

Figure 4.11: Android Application: Rejected

Lastly, Figure 4.12 shows a green tomato being predicted as unripe with 0.99 confidence by our deep learning model. This display of the performance of the Android application suggests that the deep learning model trained and the Android application developed as part of the project are robust and capable of working in a reasonable real-world scenario.

The high confidence of the trained model as shown in the above results is an indicator of the quality and high performance of the developed Android application. The users of the

Figure 4.12: Android Application: Unripe

application can take the confidence as a reference and accordingly make decisions regarding the classification results.

5. SUMMARY & CONCLUSIONS

The Tomato Ripeness Detection app, RipeVision.AI, utilizes advanced deep learning techniques to accurately classify the ripeness level of tomatoes. The app incorporates three trained ML models for tomato ripeness classification, each trained with different levels of data augmentation to improve accuracy and robustness.

The ML Model Tomato Classifier serves as the baseline, trained on a dataset without data augmentation. It provides a starting point for ripeness assessment but may be sensitive to variations in lighting and angles. The Tomato Classifier Pro is trained with heavy data augmentation, including geometric and color transformations, random erasing, kernel filters, and image mixing. This model aims to improve generalization and handle different lighting conditions and angles.

The Tomato Classifier Pro2 applies a lighter level of data augmentation, striking a balance between the baseline and heavy augmentation models. It aims to improve generalization while maintaining sensitivity to original tomato characteristics. The app uses the MobileNet architecture, incorporating an average pooling layer and dense layers with the LeakyReLU activation function. The output layer utilizes the softmax activation function for multi-class classification. The Adam optimizer is used during training. Evaluation metrics include accuracy and AUC. Accuracy measures the proportion of correctly classified samples, while AUC calculates the area under the ROC curve, indicating the model's performance in distinguishing between different classes.

The app's interface features a scan functionality using the device's camera for real-time tomato ripeness assessment. It provides a confidence level indicating the app's certainty in its assessment. Inference time is the duration for the app to predict the maturity level. The app's performance can be enhanced by leveraging multiple threads and delegates, such as the CPU and GPU delegates, to distribute tasks and utilize device resources efficiently.

Overall, RipeVision.AI aims to provide accurate and real-time tomato ripeness assessment, assisting farmers, distributors, and consumers in making informed decisions about tomato quality and ripeness.

The deliverables from the project are capability of accurately assessing the ripeness of tomatoes. Design and implementation of a user-friendly application interface that utilizes the trained tomato ripeness detection model.

REFERENCES

A. Minhas , Production volume of tomatoes across India from financial year 2015 to 2021, with an estimate for 2022. 2022. [Online; accessed 4-June-2023].

Abadi Martín, Agarwal Ashish, Barham Paul, Brevdo Eugene, Chen Zhifeng, Citro Craig, Corrado Greg S., Davis Andy, Dean Jeffrey, Devin Matthieu, Ghemawat Sanjay, Goodfellow Ian, Harp Andrew, Irving Geoffrey, Isard Michael, Jia Yangqing, Jozefowicz Rafal, Kaiser Lukasz, Kudlur Manjunath, Levenberg Josh, Mané Dandelion, Monga Rajat, Moore Sherry, Murray Derek, Olah Chris, Schuster Mike, Shlens Jonathon, Steiner Benoit, Sutskever Ilya, Talwar Kunal, Tucker Paul, Vanhoucke Vincent, Vasudevan Vijay, Viégas Fernanda, Vinyals Oriol, Warden Pete, Wattenberg Martin, Wicke Martin, Yu Yuan, Zheng Xiaoqiang. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Brosnan Tadhg, Sun Da-Wen. **2004** Improving quality inspection of food products by computer vision—a review. *Journal of Food Engineering.* 61(1): 3–16.

Chen Yud-Ren, Chao Kuanglin, Kim Moon S. 2002 Machine vision technology for agricultural applications. *Computers and Electronics in Agriculture*. 36(2):173–191.

Cho Wan Hyun, Kim Sang Kyoon, Na Myung Hwan, Na In Seop. 2021. Fruit Ripeness Prediction Based on DNN Feature Induction from Sparse Dataset. *Computers, Materials & Continua.* 69(3): 4003–4024.

Heron J. R., Zachariah G. L. 1974. Automatic Sorting of Processing Tomatoes. *Transactions of the ASAE.* 17(5):0987–0992.

Howard Andrew G., Zhu Menglong, Chen Bo, Kalenichenko Dmitry, Wang Weijun, Weyand Tobias, Andreetto Marco, Adam Hartwig. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

Huang Xing-yi, Pan Si-hui, Sun Zhao-yan, Ye Wei-tao, Aheto Joshua Harrington. 2018. Evaluating quality of tomato during storage using fusion information of computer vision and electronic nose. *Journal of Food Process Engineering*. 41(6.e):12832.

Khang Nguyen Quoc, Da Quach Luyl, Anh Nguyen Quynh. Tomatoes Dataset. 2023. [Online; accessed 4-June-2023].

Kingma Diederik P., Ba Jimmy. Adam. 2017. A Method for Stochastic Optimization.

Lu Huishan, Wang Fujie, Liu Xiulin, Wu Yuanyuan. 2016. Rapid Assessment of Tomato Ripeness Using Visible/Near-Infrared Spectroscopy and Machine Vision. *Food Analytical Methods. XI* 10(6): 1721–1726.

Minagawa Daichi, Kim Jeyeon. 2022. Prediction of Harvest Time of Tomato Using Mask R-CNN. AgriEngineering. 4(2):356–366.

Sandra , Damayanti R, Hendrawan Y, Susilo B, Oktavia S. 2020. Prediction of tomatoes maturity using TCS3200 color sensor. *IOP Conference Series: Earth and Environmental Science.* 475(1):012011.

Sharma Samrat. 2023. India wastes up to 16% of its agricultural produce; fruits, vegetables squandered the most. [Online; accessed 4-June-2023].

Taofik A, Ismail N, Gerhana Y A, Komarujaman K, Ramdhani M A. 2018. Design of Smart System to Detect Ripeness of Tomato and Chili with New Approach in Data Acquisition. *IOP Conference Series: Materials Science and Engineering.* 288, 1. 012018.