Fruit and Vegetable Freshness Detection using Deep Learning

by

Md. Mahidul Haque
19101387
Rehanul Ahmed
19101548
Somak Saha
19101286
Chamak Saha
19101401
Mayurakshmi Dutta
19101410

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
September 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

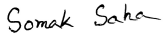
Student's Full Name & Signature:

Md. Mahidul Haque

19101387

Rehanul Ahmed

19101548

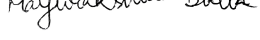


Somak Saha

19101286

Chamak Saha

19101401



Mayurakshmi Dutta

19101410

Approval

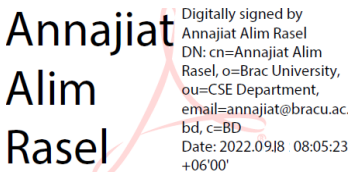
The thesis/project titled "Fruit and Vegetable Freshness Detection using Deep Learning" submitted by

1. Md. Mahidul Haque (19101387)
2. Rehnaul Ahmed (19101548)
3. Somak Saha (19101286)
4. Chamak Saha (19101401)
5. Mayurakshmi Dutta (19101410)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 22, 2022.

Examining Committee:

Supervisor:
(Member)



Annajiat Alim Rasel

Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Dewan Ziaul Karim

Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Moin Mostakim

Senior Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi , PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Bangladesh's economy depends on agriculture, which contributes significantly to GDP. It's time to automate agriculture for increased productivity, efficiency, and sustainability. Computer Vision can assist in ensuring agricultural product quality. CNN is more efficient than other (ML) algorithms for Computer Vision applications since it automatically extracts features and handles complex problems. We deployed CNN architectures to identify fruit and vegetable freshness. Using Computer Vision technology, we want to make food production, sorting, packaging, and delivery more efficient, inexpensive, feasible, and safe at the production and consumer level. Manual quality testing is laborious, inaccurate, and time-consuming. In the study, we have compared 7 pre-trained CNN models (VGG19, InceptionV3, EfficientNetV2L, Xception, ResNet152V2, MobileNetV2, and DenseNet201) with our custom, CNN-based image classification model, "FreshDNN". Our custom small Deep Learning model classifies fresh and rotten fruits and vegetables. Using this custom model, users may snap food images to determine their freshness. Farmers may utilize it to embedded systems and map out their agricultural areas on the basis of freshness of their fruits or vegetables. We trained the models on our dataset to recognize fresh and rotting fruit using image data from 8 distinct fruits and vegetables. We observed that FreshDNN had a 99.32% training accuracy, 97.8% validation accuracy and beat pre-trained models in various performance measures like Precision (98%), Recall (98%), F1 Score (98%) except for VGG19. However, our own custom model surpassed every pre-trained model for our dataset in terms of the number of parameters (394,448), training time (65.77 minutes), ROC-AUC score (99.98%), computational cost, and space (4.6 MB). We have also implemented 5-fold cross-validation where our model has performed similarly better where train, validation and test accuracy was 99.35%, 97.62% and 97.658% respectively. We believe it will perform comparably better than other pre-trained models.

Keywords: freshness, fruits, neural network, VGG19, image recognition, MobileNetV2, automation, k-fold, Image Classification

Acknowledgement

At first, all glory and praises belongs to Allah alone. It is His mercy and will to let us complete our thesis successfully.
Moreover, we are expressing our gratitude to our supervisor and co-supervisor for their kind support and advice in our work. Without their sincere and kind help it would have been impossible for us to complete our thesis.
Finally, we wholeheartedly appreciate our guardians. With their kind support, motivation and prayer we are on the verge to successfully complete our graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nutrition is essential for all living things. As a result, food is the primary source of nourishment. To sustain their daily lives, almost every organism relies on the energy they get from food. Humans, in particular, get their energy from eating. As a result of factors such as humidity, temperature, and other environmental conditions, bacteria and other live creatures are able to get into food and then spread. As a consequence, food quality degrades over time. Detecting food freshness at each step is critical (production to consumption). So many methods are used to identify food freshness, but the combination of image processing and Machine Learning (ML) is the most recent methodology that has enormous promise in the food business [1]. Among those Convolutional neural networks (CNN) based models provide the most precise results [2]. Therefore, we are going to introduce a novel CNN-based model that will make a positive impact on the food industry. In addition, our custom model is user-friendly because in several criteria for example time, space, accuracy, and computational complexity our custom model's performance is very satisfying. Besides, at the industry level tasks for example, in food sorting tasks, this model can play a vital role. On top of that, at the consumer level, the proposed model can be used in fruit and vegetable type detection as well. In a nutshell, the proposed model, FreshDNN performs better in overall performance metrics compared to other used pre-trained CNN-based models for example VGG19, ResNet152V2, InceptionV3, Xception, DenseNet201, MobileNetV2, EfficientNetV2L.

1.1 Research Problem

In a country like Bangladesh, agriculture contributes a lot to GDP, there needs to be an autonomous system to detect food on the basis of freshness. Doing these things using manual labor is costly as the production cost also increases a lot though its precision is not satisfactory. Besides, existing food freshness detection models are slow and expensive and the precision is low. Sometimes it requires chemical analysis and most of them are laboratory oriented. So, it is necessary to introduce new methods that can make the food detection process easier, less time-consuming, cheaper, and more efficient. Therefore, CNN can play a vital role in this case. From going through multiple research papers, we came to the conclusion that several ML algorithms, for example, KNN, SVM, LDA, etc are not good enough in image classifications. These ML algorithms are slow, especially for large datasets [3]. Besides these algorithms are sensitive to data size. On top of this, some of the ML algo-

rithms have limitations in multi-class classifications [4]. Further, few of these ML algorithms are costly for training purposes. Therefore, for image data classification and object detection problems, CNN plays a vital role. Because it has very good accuracy in detection or classification tasks when it comes to images. From the input data, it can extract features very well for classification. So, we have decided to work with CNN-based architecture.

1.2 Research Objectives

Our goal in this research is to propose and develop a CNN-based fruit and vegetable freshness detection model, which can contribute to the food industry and farming to make food quality maintenance easier, overall food safety management more efficient, and make the sector more profitable. Our research also focuses on the other various types of CNN-based models for image recognition, classification, and feature extraction among other deep learning algorithms. One of our goals is to make this system more accessible to people by making it more compatible with the current and future handheld devices, drones and other small electronic sensory devices. As CNN can automatically learn features from the dataset for example, if we give a large number of images of fruits and vegetables it learns idiosyncratic properties for every category of the dataset.

1. To propose a CNN model which is lighter, faster, computationally less expensive and comparatively better in performance.
2. To make an automated food quality assessment system available for small and large food-oriented businesses, farmers, and food quality inspection and safety assurance organizations.
3. To make the food production system autonomous.
4. To reduce agricultural production costs and make production more convenient and precise.
5. To enhance the quality of agricultural production and make it faster.
6. To analyse the feasibility of improving the used approach.
7. To broadly understand the application of Deep Learning based solutions regarding food safety and quality assurance.

Chapter 2

Related Work

In the article [1], according to the authors, in the food industry, the classification of food freshness is very important. As a result, the detection of food spoilage plays a vital role. According to the authors, the traditional methods of food spoilage detection are archaic and less practical. So, automatic classification methods are introduced in the food industry to detect food spoilage. That brings mobility in food spoilage detection with accuracy. In this paper, using image data sets of 3 types of food, a comparative analysis is made to figure out food spoilage. Finally, in the paper, CNN was found to be the most accurate in classification problems.

According to the article [5], Indonesia is a country where fruits and food products grow very well. Soils are very much suitable for agricultural production. Fruits are very important for human beings, As they have an immense amount of vitamins and nutrients. So, it is important to have fresh fruit. From production to consumption, in every stage of the supply chain food spoils for mismanagement, So it is extremely necessary to detect fruit freshness before consumption. The authors have designed a Deep Learning model to detect food freshness more accurately. In the paper, it has proposed a food freshness detection technique using the Convolutional Neural Network (CNN).

The article [6] reflects on the fruit detection system by a Deep Convolutional Network-based approach. According to the writers, using this system it will be easier to detect fruit in an easy and fast way. That can help immensely to automate the agricultural system. They have used RGB and NIR spectrums to make it possible to detect the fruit with high precision and the performance of the DCNN will be very high. As a result, detection will be accurate.

In the article [2], the authors explore and analyze numerous articles related to the use-cases of Deep Learning in the food sector to examine how Deep Learning performs compared to more traditional data analysis approaches. The authors have shed light on the basics of deep learning and discussed broadly several prevailing mechanisms of deep neural networks and their procedure of implementation in the food sector. In addition, they have weighed up Deep Learning with various other similar methods for data analysis while keeping in mind their respective parameters. The article has a detailed discussion on its application and analysis of CNN (Convolutional Neural Network) and its other variations regarding big data anal-

ysis, food image recognition, classification, and regression tasks. Moreover, they have presented extensive comparisons among research done by scientists with other CNN-based models in terms of their accuracy of recognition and predictability with respect to different food image-based datasets. However, it has been suggested to have more parameters for recognizing various food items. Furthermore, in this article they have investigated several real-life Mobile App based deep learning implementations, discussing their advantages and constraints they have suggested possible solutions to the problems. Additionally, the article focused on the challenges Deep Learning might face and the future potential of App-based solutions for data analysis in the food domain. Finally, the authors have observed Deep Learning be more accurate and gives better outcomes compared to other methods of data analysis and expressed their goal of this article to have more people do research work on food via deep learning mechanisms.

The authors in their paper [7], have examined the feasibility of having an inexpensive, fast, and non-destructive sensor system for evaluating the quality and classification of various foods. In this paper, they have used conventional commercial farm-produced labeled apples and organic apples as a test element for both commercially available spectrometer and their own custom-built sensor system which uses chemometric analysis, computer vision, and supervised Machine Learning (ML) mechanisms. The authors have assessed ten other methods in terms of performance for image-derived data classification tasks. Among them were, PLS-DA, LW-PLSC, SIMCA, k-NN, and others. Furthermore, for acquiring test data they have used several techniques such as diffraction image acquisition, feature vector representation, and rainbow image extraction method. In addition, in the result analysis part, the authors have compared all the algorithms on how accurate (in %) they are in classification and validation tasks on spectral data collected from the apple test samples. In their evaluation, KPLS-DA was more accurate. Moreover, in the paper, the examined data records less accuracy of the aforementioned classifiers compared to its commercial counterpart by 6.7%. However, the authors suggested applying Machine Learning (ML) mechanisms to mitigate the inaccuracies. In conclusion, the writers have claimed to have a much lower, inexpensive food quality assessment system and hoped for improving the system in the future.

In the article [8], the authors assess the feasibility of implementing a machine-learned VNIR hyperspectral imaging system to identify contaminants in minced meat products, specifically chicken meat as a contaminant in minced beef. At first, the authors shed light on existing various efforts to detect meat fraudulency and its limitations from the perspective of economic, technological, and environmental factors. They have emphasized Hyperspectral Imaging as being a faster, more reliable, and technologically more accessible methodology to detect the aforementioned adulteration. However, the authors in this paper also elucidated the limitations of Hyperspectral Imaging for real-time detection due to its high time and monetary cost and inability to acquire data at high speed. In addition, they have expressed their objectives to build a multispectral imaging system and algorithm for the detection mechanism which can analyze both spectral and spatial data for accurate prediction. Moreover, they have used the R-PLSR model to obtain a pixel map that can easily illustrate the amount of adulteration in the test sample. In conclusion, the authors came to

the conclusion that hyperspectral imaging and Machine Learning (ML) techniques combined can produce an accurately predicting tool for meat adulteration which can work online, offline, and also in the meat production line due to having the fast response to input timing, non-destructiveness, and user-friendly features.

In the article [9], according to the author, complicated self-service systems may lose customers. Keep in mind that customers are the lifeblood of every business. Consumers expect constant time savings, thus processing time must be reduced. The goal of this paper is to improve the fruit and vegetable identification process using technology. Specifically, a faster and more user-friendly system is required. Using computer vision reduces the number of possible objects and hence the user's effort. Using computer vision in self-service systems may potentially replace people in the item detection process. Eliminating the human aspect may speed up and eliminate mistakes in product identification. This research aims to look on the basics of the identifying system. The hardware consists of a camera, a display, a scale activation mechanism, and a CPU. A camera-based image classifier was developed and assessed to categorize fruits and vegetables. The author has mentioned the Convolutional Neural Network (CNN) for classification, considering their recent success in object identification and classification problems. The authors apply transfer learning to fine-tune pre-trained architectures for their application's pictures. The project comprises two phases: experimentation and implementation. The experimentation phase seeks to discover the best network for this project. It determines how the system will work in the end. The implementation phase describes how software and hardware are combined to build an identifying system. MobileNet enabled quick identification and accurate forecasts. The accuracy discrepancies across networks are less than the propagation time disparities. According to the authors, retraining the network on real-world data sets might improve accuracy. Images of fruits and vegetables in various situations were created using ImageNet data sets. Classifying each fruit or vegetable separately might be difficult for the network. Using networks allows for more precise behavior. Several problems must be addressed before this product may be sold commercially. Weaknesses in this device include a camera that may capture people's faces. In order to minimize this impact, it is possible to position the camera above the scale.

According to the author of the article [10], In a CNN, each layer of neurons receives a little portion of data from the preceding layer. It is capable of resisting even the tiniest of movements. A convolutional layer and pooling or subsampling layer makes up a CNN system. The various aspects of an image are retrieved by the convolution layer for use in image recognition. The outputs of the pooling layer are generated by activating rectangular areas. For collecting datasets they used Food-logging apps which are very familiar resources on smartphones and they can provide a large amount of data for food recognition. CNN has hyperparameters related to the number of layers, training, preprocessing, and initialization. These are the main paremeters of CNN. By comparing CNN with the Baseline method they got significantly greater accuracy in CNN than the baseline method. For food picture recognition and detection, the authors used CNNs. The author's first step was to create a food picture dataset from genuine user uploads. Then, they examined CNN's performance on 10 food items. In comparison to handmade features,

CNN outperformed. Third, they found that color characteristics are critical for food picture identification using trained convolution kernels. Fourth, they used CNN to identify food and found that it outperformed a baseline technique significantly.

The article [11], reflects on food recognition using Deep Learning. Authors hoped that the system will help people to calculate the calories of the foods of the given meal by using their system. According to this paper, Because of this system a user can just simply take photos of foods of the given meal and this system will recognize and classify multiple foods in a given meal by using image processing and computational intelligence and determine the calories of the food. In the system, there is used image data and afterward, the CNN algorithm is applied to recognize food. Because of using the CNN algorithm, the accuracy of prediction is 94.11%.

The author of the article [12] mentioned that in the fishery industry, aquatic products are spoiled very quickly due to the change in biochemical properties. For this purpose, Using the traditional method of detection of the freshness of fish is difficult as it is complex enough and so there needs to be a huge amount of samples. The authors tried to minimize the complexity of fish freshness detection by using the CNN algorithm. They introduced image data and CNN Algorithm was applied to that data to detect biochemical changes. In this paper, It is shown that CNN is more effective than any other algorithm in this case and its accuracy is very high.

In this paper [13], the authors have used a combination of Handheld near-infrared (NIR) and classification algorithms in order to correctly predict chicken filet adulteration. NIR is a spectroscopic technique used for discrimination of the food materials. The NIR technique is used for drug and food quality control. To measure meat authenticity, this paper focuses on the ensemble method of Machine Learning (ML) algorithms. Here, data is collected from supermarkets and analyzed to extract the best models for NIR spectra. Different parts of the algorithm were then simply implemented to evaluate meat authenticity using Measure and Monitor technology. Implementing Machine Learning (ML) algorithms in handheld NIR technology for checking meat authenticity can be examined efficiently. The authors came to the conclusion that this method is effective in differentiating fresh and frozen meat in food markets as it is fast, non-destructive, and simple to use.

The authors of this article [14] suggest using Machine Learning (ML), terahertz, and Swissto12 to analyze fruit quality. This approach observes the moisture content (MC) of fresh mango and apple slices. This non-invasive technique is an effective way to examine food quality, given the importance of freshness for good health, according to the author. Many strategies concentrating on fruit acidity, physical characteristics, and soluble solids content have been developed to solve this problem. Low sensitivity and resolution methods failed to detect cellular differences in fruits, the authors said. Terahertz time-domain spectroscopy was developed to gather cellular-level descriptive data, resolution, and high absorption ability. The article focuses on fresh fruit moisture content utilizing THz scattering characteristics to ensure food quality. k-nearest neighbor (KNN), Decision tree, and support vendor machine are employed. ML and THz show the potential of automated fruit freshness evaluation.

This article argues that employing this approach can assist identify early nutrient contamination in fruits, minimize health and purification costs, and boost economic advantages by assessing MC and nutritional levels in fruits. The scientists concluded that non-invasive sensing helps identify MC in fruits using KNN, Decision-tree, and SVM classifiers. It also analyzes fruits' real-time cellular data. This method detects fruit freshness.

In this paper [15], the main goal was to detect decayed items using a custom-made electronic nose. This paper emphasizes recognizing and identifying spoiled meat and fish on the basis of their smell signature. Usually, classical olfaction technologies are used in smelling air samples for determining an odor's strength. However, according to the authors, these methods used in these technologies are expensive, which is why the unbiased automatic odor analysis technologies combined with Machine Learning (ML) are preferable. The odor sensing method is actually based on metal oxide semiconductor-based electronic noses. With the help of different types of available sensors in the market, spoiled meat is detected. It is an effective method in the food industry to identify decayed fish and meat. Using this order sensing technique, the meat and fish quality can be examined. Finally, they have concluded that this technique is very effective in the food industry for quality control, freshness evaluation, process monitoring, authenticity assessment etc.

The authors [16] in their paper mainly focus on developing a system which is efficient as well as low cost to inspect fruit freshness on the basis of fruits' external appearances. To develop the model they used the technology combination of computer vision and deep learning. Initially, in the papers, the authors reflected on the previous work regarding fruit freshness detection using ML algorithms. Then the author explained the system architecture. The system Architecture basically consists of 3 components. The first component is the deep learning modeling, for which they used the CNN algorithm. Secondly, they used the Raspberry Pi and camera module (Hardware). Thirdly they used the touch screen module to make the system more feasible for the users to use. For the CNN training purposes, they used the data set of Apple and Banana. In the last part of the paper, the authors show the result of the models and show how their model(Efficient Net) is more efficient than other existing models.

In the article [17] the authors reflect on building a new version of MobileNet architecture for the purposes of image classifications. Initially in the paper, the authors talk about the importance of food classification. Besides, they also point out the recent and existing work on the food classification system and also discuss the challenges of food classification. According to them, the biggest challenge of food item classification is to find out the real food from an image that consists of different objects including foods. Then, the authors discuss the architecture of their proposed system. Their CNN model is the modified version of Mobile Net. For the modifications, they are extracting some layers with activation function of the original Mobile Net models and then they included some new layers with activation function in the model. Besides, to develop the model they used data augmentation techniques to bring more variation in the images. The main objective is to make a model that is faster and more accurate than the mobile Net first version and other models,

also, reducing the overfitting problem. For developing these classification models they used more than a hundred classes of foods images dataset. Lastly, the authors compared the performance of their proposed version of MobileNet with other models.

In the article [18] the authors (Wang et al., 2020) discuss the updated version of Mobile Net called DenseNet that is basically introduced into the Mobile Net by using Dense Blocks. In the paper, the authors try to build 2 DenseNet models. One is Dense 1 MobileNet another is Dense 2 MobileNet. Both Dense 1 MobileNet and Dense 2 Mobile net take depthwise convolution and it is separable. But one takes it as a basic unit (Dense 1 MobileNet) another takes it as a whole (Dense 2 MobileNet). Besides, both of these models have reduced the parameters compared to the original MobileNet version. On top of that, the authors also discussed the experiment procedures where for both of these models the same hyperparameter conditions are used. Moreover, in the paper, in the result analysis part, the authors showed the highest number of iterations (5000) Dense 2 mobile has shown the highest accuracy compared to other models. In a nutshell, the proposed models by the authors are very useful in mobile devices for image classifications.

In the article [19], the authors demonstrate a lightweight convolutional neural network model called MobileNet that can be fruitful in the use of handheld smart devices and embedded vision applications. Besides, the authors emphasize that MobileNet would be effective in various fields especially in detecting objects, face attributes and large scale geo-localization. The main aim of MobileNet architecture is to build a model that is small in size and low in latency, and comfortably meets the design standard of hand held device and embedded vision applications. Moreover, the authors reflect that earlier many efforts have been made regarding the building of small efficient neural networks but those mainly focused on the small size, not caring about speed. The architecture of the MobileNet model maintained depth wise separable filters besides. Besides, the authors demonstrate a hyperparameter called width multiplier and resolution multiplier in order to achieve accuracy and low latency. Finally, the authors have shown a comparison of their proposed model with other models and show the effectiveness of MobileNet in various applications.

In the article [20], the authors (Labiba et al., 2022) demonstrate the classification of fruit and vegetables using two deep learning models, VGG16 and YOLO. Firstly, they will detect the object from the image whether it is a vegetable or fruit and then they classify the object into 3 types such as pure fresh, medium fresh and rotten. The authors also assert that in their approach the most unique and difficult part is to detect the medium fresh fruit and vegetable and that is very crucial in terms of minimizing the wasting of vegetables and fruit. Besides, the authors reflect that earlier several Machine Learning (ML) approaches have been applied for classification. But CNN performs much better than Machine Learning (ML) approaches. Moreover, the authors show different types of comparison between VGG 16 and Yolo and found less classification accuracy in VGG16. Lastly, the authors make an android application that is capable of doing classification from images as input and giving results of freshness degree.

Chapter 3

Proposed Model

We have created a Convolutional Neural Network (CNN) model which will be able to identify food freshness based on given image data. This is a classification problem that will extract features from a given image to identify distinct patterns and be able to differentiate different images based on those extracted features, thus identifying which food is fresh and which is not.

As this is a pictorial-based classification task it requires graphical processing power to process input data/images. Hence, it requires a dedicated GPU (Graphics Processing Unit). GPU can handle large graphical processing tasks that's why we needed a dedicated GPU. By using GPU our classification task will be performed efficiently and quickly. To acquire a dedicated GPU for our work we have used the Brac University research lab PC as it has a GPU support option.

Furthermore, we are using Python as the base programming language as it is the most preferred programming language for Machine Learning (ML). It has a huge library which gives us the flexibility to solve various artificial intelligence-related problems. We have used TensorFlow Keras API Library rather than PyTorch to import functions necessary for our model because it has a bigger community which is why it is easier to find resources and find solutions to the problems. Plus, for production, scalability and models, TensorFlow is comparatively better than PyTorch.

Besides, we have collected data sets from reliable sources, for example, Kaggle, Google Data sheets search engine etc, and also have collected data manually by taking pictures. Then we merged the data sets collected from different sources altogether. After collecting the data we have renamed every single sample image in a similar format so that later on it will be easier to do the automated splitting. Then we analysed how much data on each label set we collected. In our collected data set, we have in total of 36,800 sample images.

By this, we were able to determine the amount of data we need in the train, validation and test set. We have done data prepossessing before splitting it into train, test and valid sets. After that, we separated the data for training, validation and testing. Our ratio of Train:Validation: Test is 7:2:1. In that data set, we have 25760 images for the train set, 3680 images for the test set and 7360 validation images which belong to 16 classes. We have created a CNN model, where we have set opti-

mizer and activation functions. Then we trained the neural network after compiling the model. After extracting features from the images we have predicted our model using a test set by applying a prediction function. Thus detecting food freshness.

To further evaluate our model, we have implemented 5-fold cross-validation method. Where each of the model have been trained 5 times on train and validation combined data set. As it is a 5-fold, all the models were trained 5 times and after that the average score was taken. Furthermore, the models were tested on the unseen test data set to complete our 5-fold cross-validation evaluation.

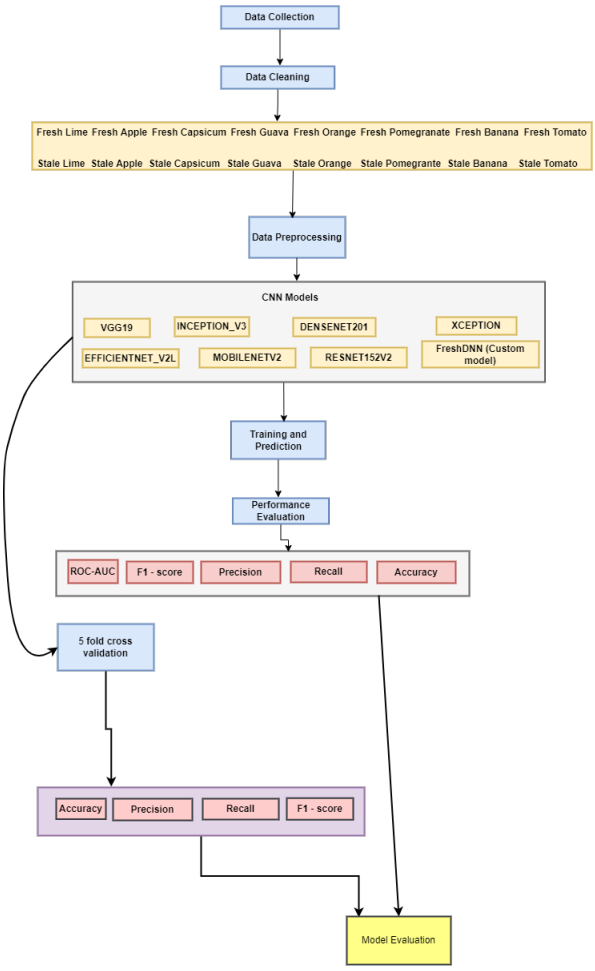


Figure 3.1: Our Workflow

3.1 Data Analysis

In every research, data analysis is essential. Data Prepossessing and Input Data are the two sections of our suggested system's analysis.

3.1.1 Input Data

For data collection purposes, initially, we went to Local Bazaars, for example, Mohakhali Kacha Bazar and other local Bazaars to collect data manually. Moreover,

we collected fruit and vegetable image data from some online resources such as Kaggle, Mendeley Data, google search engine, and so on [21] [22]. Besides, we collected a good amount of data by Web Scraping. Also, we did some data augmentation. Then we extracted some invalid image data manually as that is not related to our research. Then we made multiple category folders for multiple categories, merged all valid image data collected from different sources on the basis of category, and put them in the respective category folders. After that, using python programming, we did file naming of each category folder serially. Altogether, we collected raw data from 16 categories such as fresh apple (9677 images), fresh banana(6908 images), fresh capsicum(2749 images), fresh guava(2909 images), fresh lime(2788 images), fresh orange (6192 images), fresh pomegranate (5940 images), fresh tomato (3718 images), stale apple(3483 images), stale banana (2554 images), stale capsicum(2374 images), stale guava (2843 images), stale lime(2775 images), stale orange(2754 images), stale pomegranate(3014 images), stale tomato(2530 images).

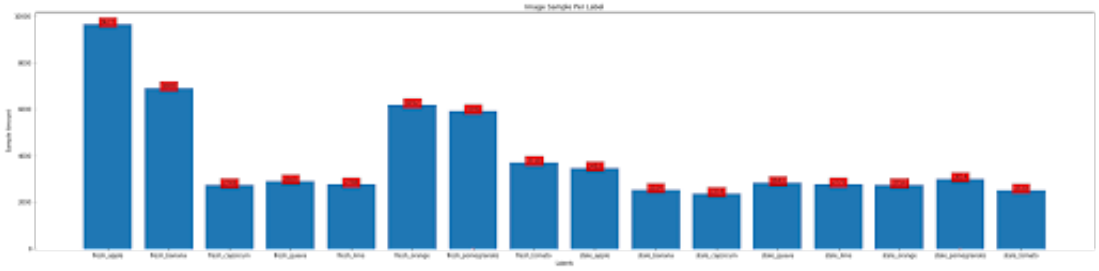


Figure 3.2: Initial imbalance in data set

As the amount of raw data is imbalanced among different categories, we cannot use all the data for model training because if we do so, ultimately it will provide better results for the categories that have higher numbers in amount. Therefore, for the model training purpose, we have taken 2300 image data of each category and then we separated the 2300 image data of each category on the basis of training, validation and test. For each category, the amount of training data is (2300* 0.7) = 1610, Validation data is (2300* 0.2) = 460 and Test data is (2300*0.1) = 230.

Figure 3.3: Image sample of the data set

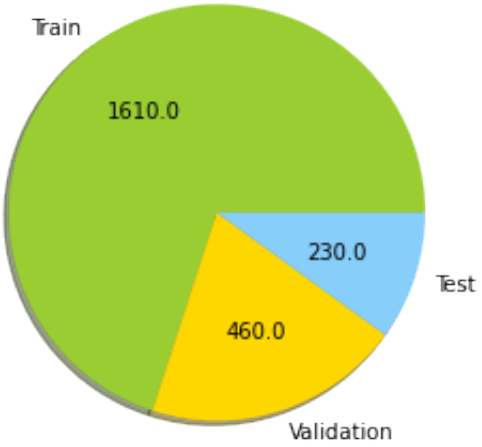


Figure 3.4: Post-splitting state of the data of each category

3.1.2 Data Preprocessing

Image prepossessing is done to the image quality to analyse in a more accurate way. It helps to suppress unwanted distortions as well as enhance some necessary features for our projects. We also do image prepossessing in order to generalise the shapes or structure of the image so that every image sample trained on the model has the same outlook. Thus, image preprocessing helps us to get a more precise result.

We have performed data augmentation below are the values of our augmentation range

- Image size: 224x224
- Batch size: 32
- Rotation Range: 10
- Width Shift Range: 0.1
- Height Shift Range: 0.1
- Shear Range: 0.15
- Zoom Range: 0.1
- Channel Shift Range: 10
- Horizontal Flip = True

Our model is a multi-class classification problem where the classes are label sets. By applying these techniques we have made our data set ready to be pushed into our CNN models to train the data set.

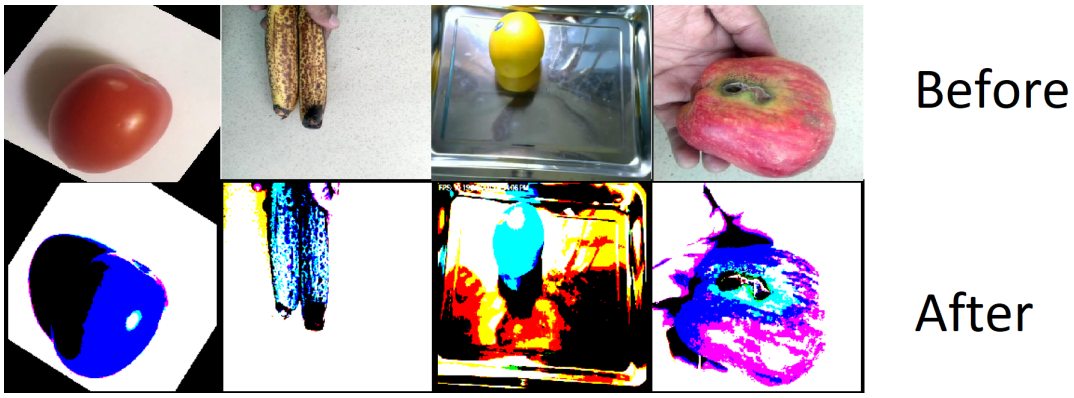


Figure 3.5: Before and After prepossessing images

3.2 Model Architecture Using CNN

Our research is entirely Convolution Neural Network(CNN) based. Because, in the research, we are dealing with completely image data. For image classification and object detection problems, CNN is a far better algorithm compared to other Machine Learning (ML) algorithms, for example, SVM, KNN, LDA and so on. Because for image classification tasks people need to create features from the image data and later they need to feed those features into ML algorithms [23]. But for CNN, image features extractions are done automatically. Besides, transfer learning happens in CNN so that it learns more and causes less error. On top of that, there happens a comparison of images piece by piece, which makes it easier to find the similarity between the images. That is why CNN provides better accuracy in image classification problems compared to other ML algorithms.
In our research, we have trained our collected image data sets into pre-trained CNN models such as VGG19, MobileNetV2, InceptionV3, DenseNet201, Xception, EfficientNetV2L, and ResNet152V2. Besides, we have also designed a customised novel model called FreshDNN and analyzed the model on various performance metrics with pre-trained models.

In a typical CNN architecture, there are three layers besides the input and output layers:
1) Convolutional Layer
2) Pooling Layer
3) Fully Connected(FC) Layers
Convolution Layer takes a picture as input and extracts features using filters. The filter glides over the input picture, performs a mathematical operation (dot product), and generates feature map as the result [24]. A feature map provides information about the image's corners and perimeter. If required, the feature map may be fed into other layers to extract more picture features. Primarily Pooling Layer following the Convolution Layer is the Pooling Layer. This layer's primary purpose is to minimize the size of convoluted feature maps and reduce computing expenses. There are a variety of pooling types, including Max Pooling, Average Pooling, and Sum Pooling. The Pooling Layer functions as a bridge between the Convolution and fully linked layers. Prior to the output layer is the Fully Connected Layer. In addition, this layer contains the weights, biases, and hence the neurons that are

utilized to link neurons from two separate layers. Additionally, the preceding layers are flattened and included into this layer. This layer is where categorization begins.

To create our model we have used the TensorFlow Keras library. We have also used functional API to create the structure of the model as functional API is more flexible and better than Sequential API. Adding three Convolution layers and three Dense layers we have built our hidden layers. Our input layer shape is 224x224x3. In total, we gained 394,448 total parameters and all of them are trainable.

3.2.1 The Convolutional Layer

As the number of layers in a deep neural network rises, the number of parameters expands exponentially, which may make training the model computationally demanding. Optimizing these several parameters may be a big undertaking. CNN's minimize the time required to fine-tune these parameters, and they are also excellent at reducing the number of parameters while maintaining model quality. Convolution may be used to blur or sharpen a picture by adjusting the filter parameters.

Moreover, in CNN many convolution filters are active in all its layers, which scan the entire feature matrix as well as perform dimensional reduction. That is why CNN is an excellent network for picture categorization and analysis [25].

$$SAME\,Padding(Both\,Sides) = (Stride - 1) * (Input\,Height) - Stride + Kernel. \tag{3.1}$$

Equation: *Number of parameters in our model*

$$\frac{(Input\,Height - Kernel + 2 * Padding) + 1}{Stride} \tag{3.2}$$

3.2.2 Pooling Layer

The pooling Layer connects the Convolutional Layer to the Fully Connected Layer. It is another CNN component that is beneficial for image prepossessing. If the picture is enormous, the pre-process reduces the number of parameters to compress it. Also, as the image is downsized, the pixel density is reduced, and the preceding layers are used to form the down scaled image.
In our model, we have included Max Pooling, which takes the maximum of an area and helps to begin with the image's most significant characteristics. It is a technique for transforming continuous functions to discrete counterparts based on samples. Its primary objective is to minimize the dimensionality of an input, reduce the size of produced feature map to reduce processing costs, and make assumptions about the rejected sub region's characteristics. In addition, this technique is used more often than typical pooling.

3.2.3 Fully Connected Layer

The pixel values of the input picture are not explicitly linked to the output layer in partly attached layers. In the fully-connected output layer, however, each node is directly linked to the node in the layer before. The FC layer i.e. Fully Connected layer primarily conducts classification related tasks based on the collected features and also the filters from the layers before it. At this stage, one Dense layers and one flattening layer were used.

3.2.4 Proposed Model Architecture

FreshDNN is our custom CNN model that is being applied in our research. It is a novel model that is dedicated to detecting fruits and vegetable freshness. It is a light weight small Deep Learning model.

Architecture:

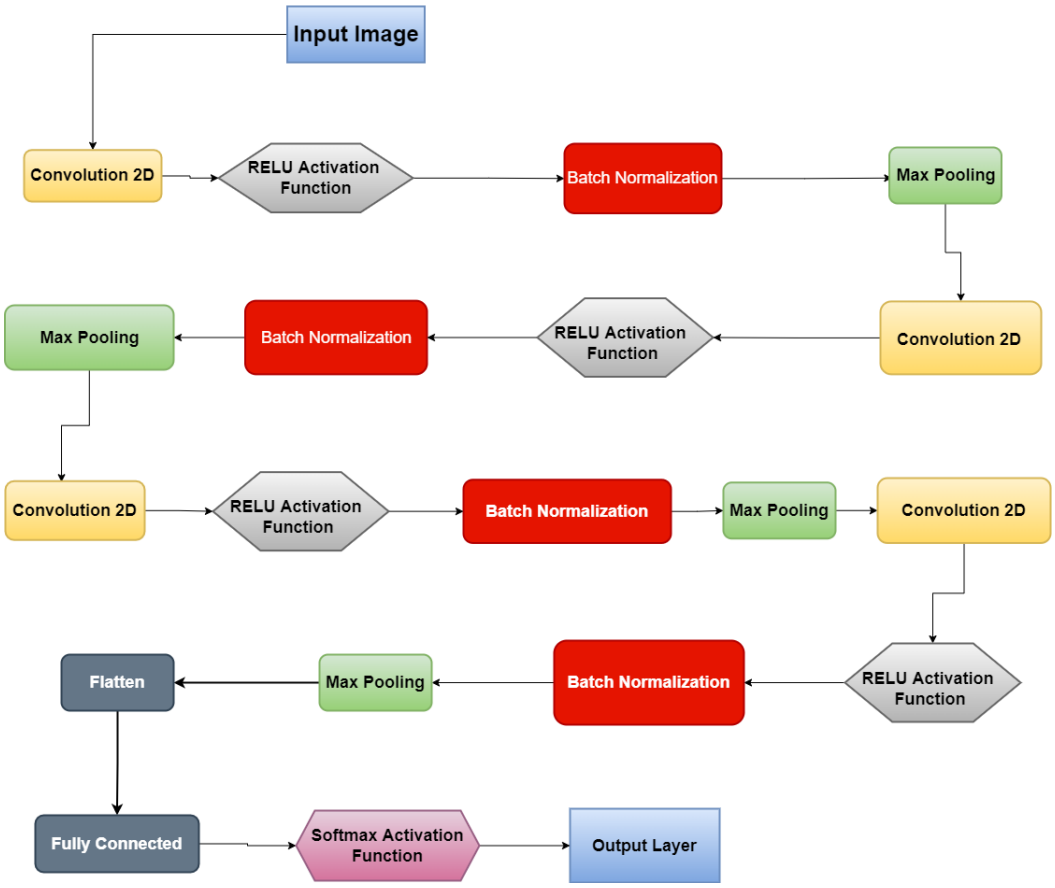


Figure 3.6: Model Design

1. In the input layer the model takes a (224,224,3) dimensional input image.

2. Then these input images are fed through the Conv2D layers and perform operations to do feature extraction. From the very first convolution layer to do the Convolution operation, we used 32 filters. To keep the output size of the image the same, we used the same padding. Besides, the stride of 2 and kernel size of 3 has been used, and to prevent exponential growth of computation we used the ReLU activation function. For making the learning of the layer independent in the neural network we used Batch Normalization.

3. The feature map constructed using the first Convolution layer and then is being passed to the Pooling layer where Maxpooling of stride 2 is done to extract the major information of the image.

4. Then the output extracted from the Maxpooling operation is passed to the next convolution layer that consists of 64 filters. Here stride, kernel size, and

the activation function are similar to the previous convolution layer. Also, in this layer, the model did Batch Normalization and max pooling was applied to the output feature map.

5. Using 128 and 256 filters we used 2 more consecutive convolution layers and also a pooling layer similar to previous convolution layers.

6. Then the output of the last max pooling layer is flattened. The model used a dropout of 0.3 to fix the over-fitting issue.

7. Lastly, we used the Dense layer to classify the classes to detect freshness and we used the Softmax activation function to classify multi-class precisely.

Table 3.1: Number of parameters in our model

Layer (type)	**Output Shape**	**Parameters**
Conv2d_286 (Conv2D)	(None,112,112,32)	896
batch_normalization_286 (BatchNormalization)	(None,112,112,32)	128
max_pooling2d_15 (MaxPooling2D)	(None,56,56,32)	0
Conv2d_287 (Conv2D)	(None,28,28,64)	18496
batch_normalization_287 (BatchNormalization)	(None,28,28,64)	256
max_pooling2d_16 (MaxPooling2D)	(None,14,14,64)	0
Conv2d_288 (Conv2D)	(None,7,7,128)	73,856
batch_normalization_288 (BatchNormalization)	(None,7,7,128)	512
max_pooling2d_17(MaxPooling2D)	(None,3,3,128)	0
Conv2d_289 (Conv2D)	(None,2,2,256)	295168
batch_normalization_289(BatchNormalization)	(None,2,2,256)	1024
max_pooling2d_18(MaxPooling2D)	(None,1,1,256)	0
flatten_7(Flatten)	(None,256)	0
dropout (Dropout)	(None,256)	0
dense_6 (Dense)	(None,16)	4112
		Total: 394,448

3.2.5 Pre-Trained Model Architecture

InceptionV3:

Inception V3 is a CNN model and its objective is to minimise computational costs. The given below techniques are used to build the model.

Factorization into smaller convolutions: It helps to reduce the computational cost by reducing parameters. Using Factorization it is possible to replace bigger convolutions with smaller multiple convolutions. Therefore, it helps to train faster. For example, the number of parameters for a 5×5 filter is 25 whereas it is 18 in total for two 3×3 filters. That is how it reduces parameters and so computational resources. Using factorization 28% of parameters can be reduced.

Factorization into Asymmetric convolutions: Any convolutions filter size

greater than 3 × 3 may not be beneficial as its computational cost is higher. So, it can be resolved by replacing the convolution with multiple smaller convolutions. A 3 × 3 convolution can be replaced with 2× 2 convolutions but a better result can be obtained if we replace 3 × 3 convolutions with 1 × 3 convolutions which are also followed by 3 × 1 convolution (asymmetric convolution). Using this technique 33% of parameters can be decreased.

Auxiliary Classifiers: The component is put between the layers. During training, the loss is computed and added to the overall network loss. The auxiliary classifier is used to mitigate the gradient vanishing issue. Besides, it functions as a regularizer.

Grid Size Reduction: Pooling operation is used to make grid size reduction. The above-given techniques are integrated into the Inception V3 architecture

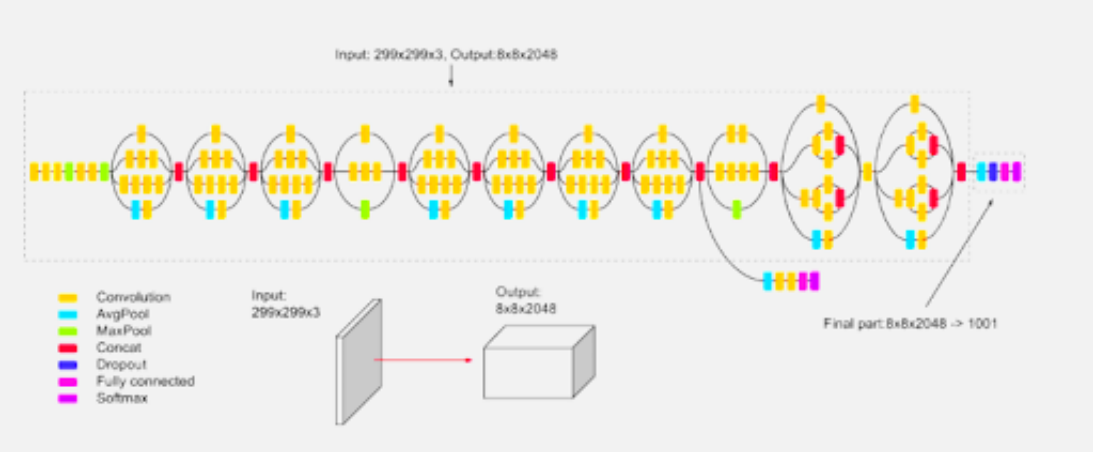


Figure 3.7: InceptionV3

DenseNet 201:

DenseNet is a CNN model which is a bit different from traditional CNN models because traditional CNN models have n layers with n connections but in DenseNet n layers have n(n+1) connections. For DenseNet, as input, in each layer, preceding feature maps are used. Each layer is directly connected to all previous layers.

Architecture: Basic DenseNet Composition layer: Batch Normalisation layer, ReLU activation function, and a 3×3 convolution respectively follow this kind of layer.

BottleNeck DenseNet (DenseNet-B): In DenseNet there happens to concatenate features in every layer, therefore the computational costs of the networks become high, so, the author proposed a bottleneck structure. In bottleneck structure firstly 1×1 convolutions are used then 3×3 are used.

DenseNet compression: For the sake of model compactness, in DenseNet, in the transition layer, there tried to reduce the feature maps.

Multiple Dense Blocks with Transition Layers: 1×1 convolutions and 2×2 average pooling layers follow dense blocks. In this architecture, transition layers are convenient to concatenate as feature maps are the same in size.

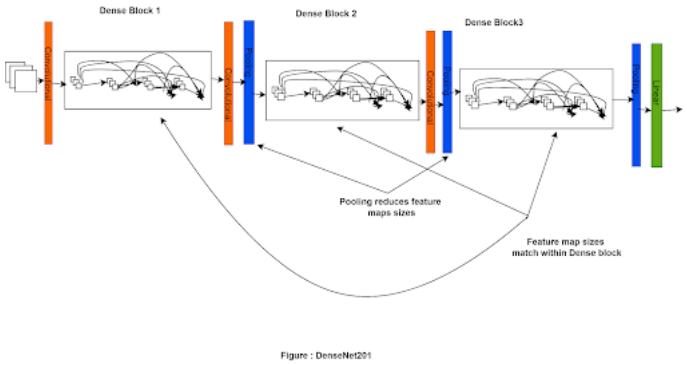


Figure 3.8: DenseNet201

Xception:

Xception is basically the extreme form of inception. Inception there used 1×1 convolution to compress the input then there used multiple types of filters to extract features. Xception does the opposite of inception. In Xception, firstly it uses the filters to input images to extract features then 1×1 convolution compresses the input space. This method is nearly the same as a depthwise separable convolution, a neural network design operation that has been utilised since 2014.

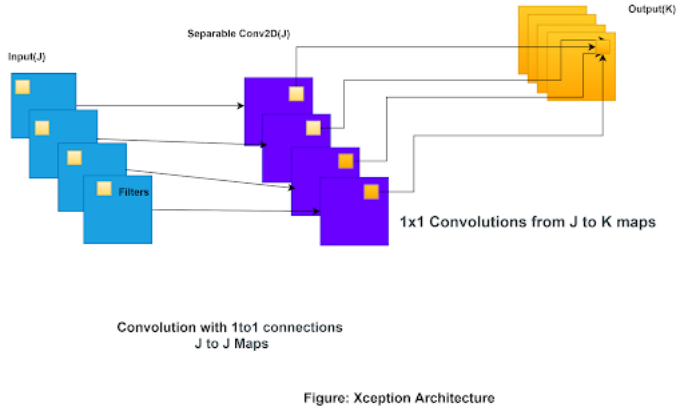


Figure 3.9: Xception Architecture

ResNet152v2:

Plain Network: The convolutional layers mostly adhere to two fundamental design criteria. The first is that each layer has the same number of filters for the same output feature map size, and the second is that if the output feature map size is half, the number of filters would be twice to accommodate the time complexity per layer. By downsampling, it is simple to create a stride with two levels. Typically, the filter size in convolutional layers is 3X3.

Residual Network: Shortcut connections are inserted on the basis of a plain network. The identity shortcut can be applied directly for the same dimensions of input and output. And two approaches can be performed for the increased dimensions. By adding zero entries padding, dimensions can easily be increased which helps shortcuts to perform identity mapping and there is no use of parameters in this approach. In another approach, the projection shortcut can be performed by using a 1X1 filter to match dimensions. In both of the approaches stride of 2 is applied.

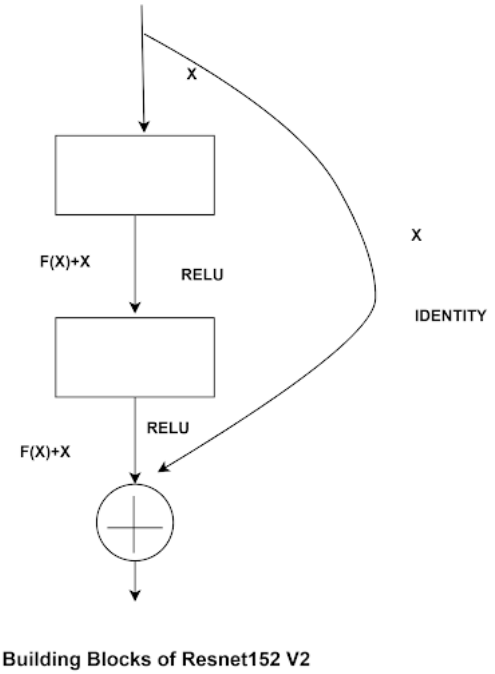


Figure 3.10: ResNet152V2 Architecture

EfficientNetV2:

Efficient Net V2 is a CNN model that is built by scaling the dimension of the network. But there needs to make a balance while scaling dimensions otherwise there will be a vanishing gradient problem. Scaling up resolution means increasing the pixels that help to extract more complex features of the image. Scaling up depth means adding more layers in the network that is required to extract features when the resolution of the image is high. Scaling up width means adding more feature maps. To scale up the model there needs a compound scaling method. Efficient Net architecture is built by scaling up the base network.

$$F = \alpha\beta^{\phi}\gamma^{\phi} \tag{3.3}$$

$$F = dw^{\phi}r^{\phi} \tag{3.4}$$

α is d; depth scaling factor
is w; width scaling factor
γ is r; resolution scaling factor.

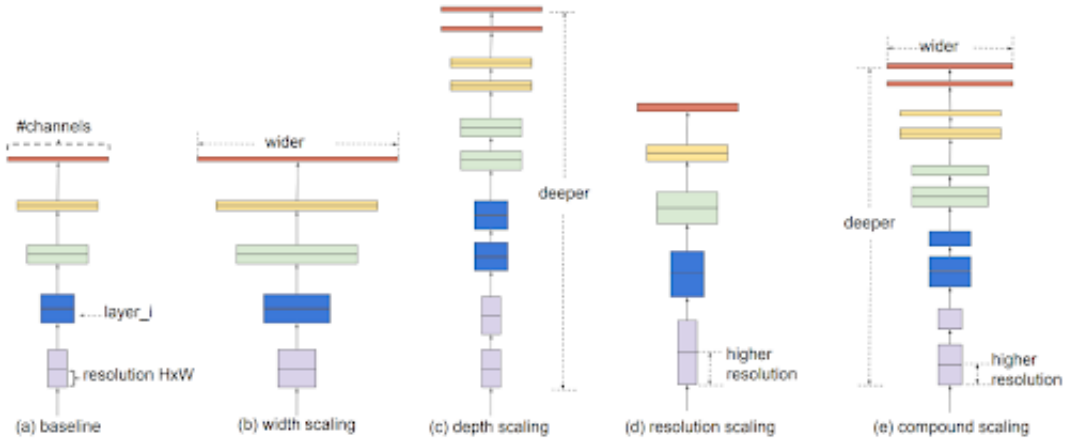


Figure 3.11: EfficientNet Architecture

VGG19:

VGG19 is a Convolutional Neural Network and a modified VGG model consisting of 16 convolution layers and 3 connected layers; in total 19 layers. Additionally, it has 1 softmax layer and 5 MaxPool layers. Besides, it has been trained on millions of image data sets from the ImageNet database and it can easily classify the images into a number of categories, which is why this neural network has lettered rich attribute representations for a good amount of images.

VGG19 is created by a group named Visual Geometry Group. It was created based on a few ideas from AlexNet. However, the ideas are improved in this network and it uses convolution layers in order to enhance the accuracy. VGG19 is now commonly used for a lot of difficult and challenging tasks, in particular, in 2014 this model surpassed other models providing the most accurate and since then it is considered an important one.

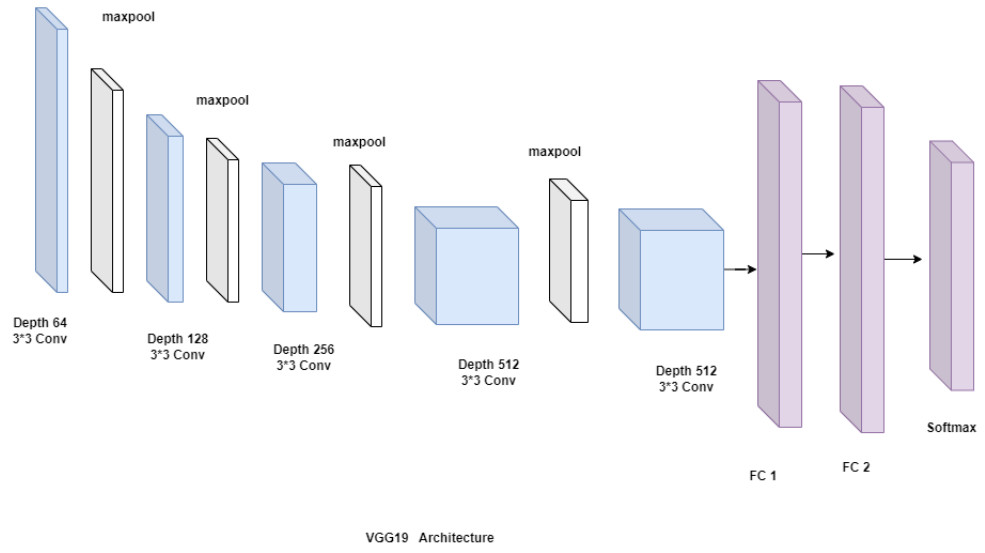


Figure 3.12: VGG19 Architecture

Here, "Depth 64 3*3 Conv" means there are in total 64 kernels in this convolution layer having a size of 3x3 matrix.

At first, an RGB image with a fixed size of (224x224) is given as input which is converted as a (224x224x3) sized matrix.

For preprocessing, the model subtracted the mean value of RGB from each of the pixels that were computed over the training set.

Then (3x3) size kernels have been used with a 1-pixel stride size which enabled it to cover the image's overall features. In order to preserve the image's spatial resolution spatial padding of 1 pixel is used.

Furthermore, max pooling is performed with stride 2 over a window of 2x2 pixels. MaxPool is used to reduce the overall size of the computed matrix pooling out the maximum value from each 2x2 sized window. Then to enhance the computational

efficiency as well as make the model classify more accurately the ReLu (Rectified linear unit) is used. This helped to introduce non-linearity and also gave better results than other models.

Then it implements 3 Fully Connected layers. Among the 3 layers, the first two have a size of 4096 and the final one is the softmax layer with 1000 channelled layers for ILSVRC classification in 1000-way is used here. Convolution layers mainly extract data from an RGB image and Fully Connected layers classify the data accordingly.

VGG19 is mainly designed for the ILSVRC, but its use is not limited to this. It is also used for tasks like facial recognition. Moreover, weights with some other frameworks like Keras are available so that they can be modified and we can use them as we want.

Not to mention, a direct comparison of VGG19 with any of the other models is neither logical nor possible as different sampling algorithms will give different results. Nevertheless, when using any technique for single central-crop sampling, then VGG-19 beats the others producing good accuracy.

MobileNetV2:

MobileNetV2 is the new and improved version of MobileNetV1; which was designed with the help of mobile devices for supporting detection, classification, and many more. This model gives the flexibility to run and implement deep networks on mobile phones. As a result, it provides an enhanced user experience with no access barrier as well as more benefits for energy consumption, security, and privacy.

The MobileNetV2 model is used widely for its significant improvement in visual recognition which also includes semantic segmentation, image classification, and object detection.

For structured building blocks, the MobileNetV2 uses depthwise separable convolution. This procedure is very much similar to its previous version MobileNetV1. However, for better performance V2 has introduced 2 new attributes in its architecture.

MobileNetV2 includes 53 convolution layers which can be categorised into two types: 1x1 Convolution and 3x3 Depthwise Convolution. This model also has 1 AvgPool with about 350 GFLOP. Here we get two types of components. 1st one is an Inverted Residual Block with 1 stride and the other is Bottleneck Residual Block with 2 strides. Each of these blocks has three different layers.

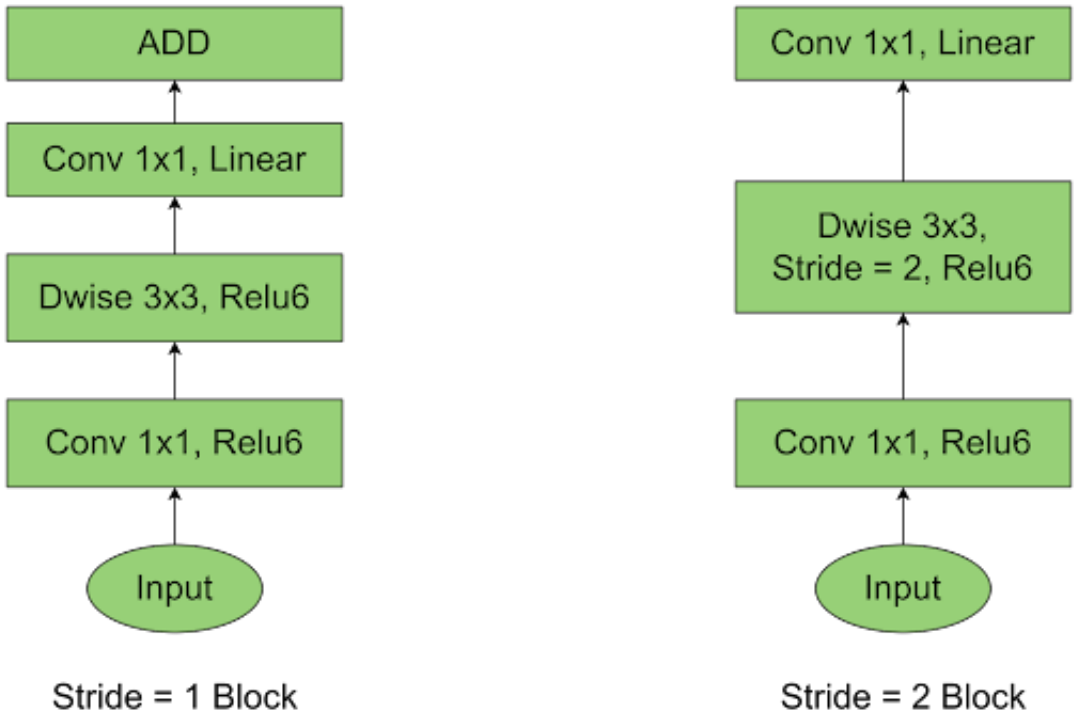


Figure 3.13: MobileNetV2 Architecture

This version has two 1x1 Convolution layers. 1x1 Convolution with Relu6 is referred to as the expansion layer since it increases the number of channels prior to the depthwise convolution layer. From depthwise convolution, layer data passes via a non-linear 1x1 Convolution layer. This layer is also known as the projection layer since it projects data with a greater number of channels to a dimension with fewer channels. This layer actually reduces the size of the channels, contrary to v1's practice. By reducing the size of the channels, the data flow via the network becomes congested. Consequently, this layer is also known as the bottleneck layer. Data flow through the layers is shown in the figure:

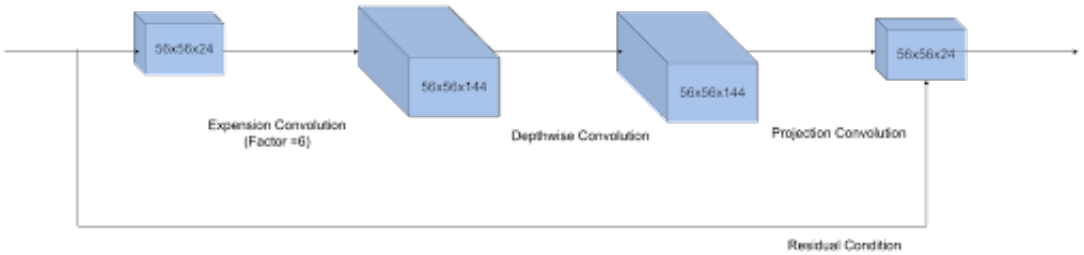


Figure 3.14: MobileNetV2 Dataflow

The residual connection is an additional innovation offered by V2. It has comparable features as ResNet. It facilitates the movement of gradients across the network. In conclusion, the MobileNet V2 contains a total of 17 blocks, as described before. Each of these blocks consists of three levels. A conventional 1x1 Convolution layer, a global average pooling layer, and a classification layer follow. The first of a total of 17 levels has a somewhat different construction. Instead of the expansion layer,

a conventional 3x3 Convolution with 32 channels is used. Overall layers are shown in the figure:

Op	Expansion	Repeat
Convolution	-	1
Bottleneck	1	1
Bottleneck	6	2
Bottleneck	6	3
Bottleneck	6	4
Bottleneck	6	3
Bottleneck	6	3
Bottleneck	6	1
Convolution	-	1
AvgPool	-	1
Convolution	-	1

Table 3.2: Overall layers

Chapter 4

Result Analysis

In the beginning, we applied some pre-trained models like VGG19, MobileNetV2, InceptionV3, ResNet152V2, Xception, EfficientNetV2L and DenseNet201 on our full dataset. For applying these pre-trained models, for several models, we have modified the layers of the original structure in order to suit our dataset. Then we added 16 label classes in our dense layer for every model. In that layer, we have used the Softmax Activation function. In our compile part, we have used Adam Optimizer, as Adam Optimizer has faster computation time and it needs fewer parameters to finetune the algorithm. It is also the best adaptive optimizer. Furthermore, we have applied categorical cross-entropy as we are doing a multi-classification problem, and we have set our metrics as accuracy.

We have used the model. fit() function to train our model. Also, we have used valid batches as validation data. Then set the epochs to 30.

4.0.1 Performance Metrics

True Positives(TP): When the models predicted the positive class correctly, the outcome becomes (TP)

True Negatives(TN): When the models predicted the negative class correctly the outcome is (TN)

False Negative (FN): When the models predicted the negative class incorrectly the outcome is (FN)

False positive(FP): When the models predicted the positive class Incorrectly, the outcome becomes (FP)

Accuracy:
Out of the total number of test samples, the number of test samples correctly identified by the model as either truly positive or truly negative.The formula of accuracy is

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{4.1}$$

Precision:
Precision implies in which proportion positively identified test samples are actually positive.

$$Precision = \frac{TP}{(TP + FP)} \tag{4.2}$$

Recall
Recall implies in which proportion of actual positive samples are recognized as positive.

$$Recall = \frac{TP}{(TP + FN)} \tag{4.3}$$

F1 Score:
F1 score implies the harmonic mean of the harmonic mean of precision and recall

$$F1score = \frac{(2 * precision * recall)}{(precision + recall)} \tag{4.4}$$

ROC-AUC:
ROC is a graph that depicts the performance of a classification model over all classification thresholds. TPR defines True Positive Rate(recall) FPR defines False Positive Rate

$$TPR = \frac{TP}{(TP + FN)} \tag{4.5}$$

$$FPR = \frac{FP}{(FP + TN)} \tag{4.6}$$

An ROC curve basically plots TPR vs FPR at different classification thresholds.
In multiclass classification to plot ROC curve there needs to take OvR (One vs Rest) strategy. In OvR one class is considered positive and the rest of the class are considered as negative. Using the OvR strategy, we convert the output of the multiclass classification to a binary classification
AUC defines the area under the ROC curve. It measures two things: one It assesses how well predictions rank, another It assesses the quality of the model's predictions regardless of the classification threshold used.

4.0.2 FreshDNN

This is our custom model. In this model, we have fed 224X224 sized image data. This model has 4 Convolutional layers and 1 Dense and Drop out layer. Our model took 1 hour and 5min to train. Upon finishing the training we produced accuracy-loss graph and confusion matrix. The orange color represents test accuracy or loss (accuracy or loss) per epoch, whereas the blue color represents train accuracy or loss (val_accuracy or val_loss) per epoch

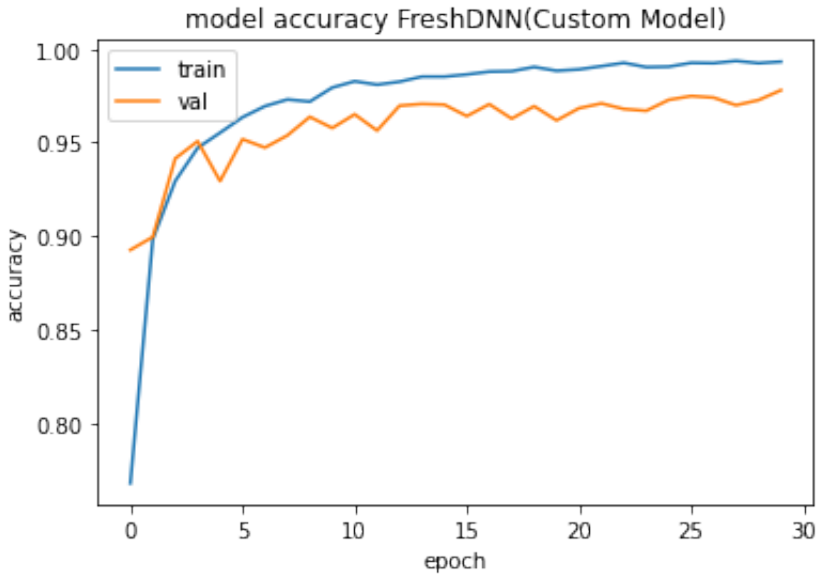


Figure 4.1: FreshDNN Accuracy

The above graph depicts the accuracy of FreshDNN over different epochs. In the graph, it is clear both training and validation accuracy are growing over the increase of epochs. Initially, the training accuracy is less than 80% whereas the vali- dation accuracy was around 90%. During the 3rd epoch, both the training and validation accuracy was the same. But as it moved forward the validation accuracy became lower than the training accuracy. In the final epoch, the training and validation accuracies are 99.3% and 97.7%.

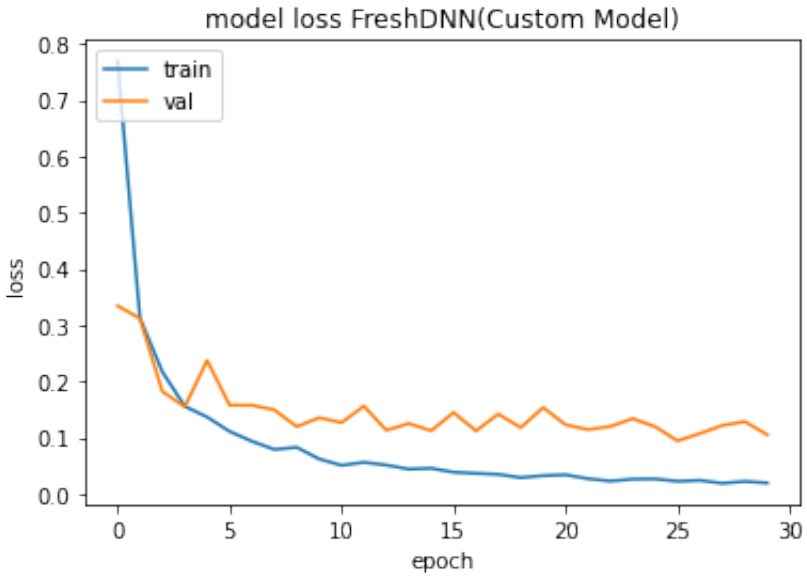


Figure 4.2: FreshDNN Loss

The graph represents the relationship between loss in each epoch. For the training case, the initial loss is 0.7699, then, it decreases gradually after each epoch. Additionally, the loss has turned down to 0.0195. In validation cases, at the 4th epoch, the loss is more than 0.2 and in the 15th epoch it has decreased to 0.18, then again, it moves down. Finally, the loss concludes to 0.1052

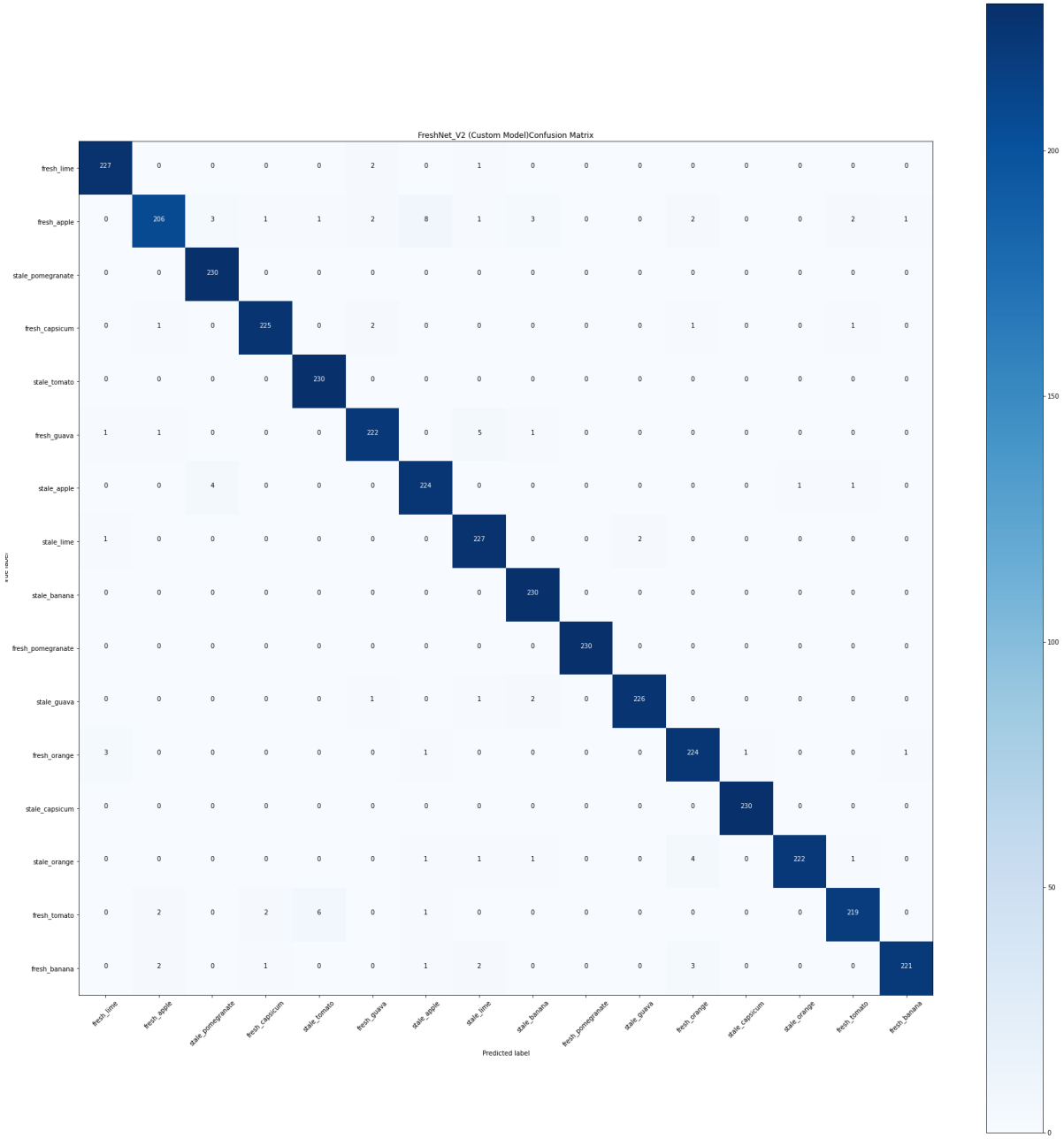


Figure 4.3: FreshDNN Confusion Matrix

The model has predicted fruits and their state perfectly in most cases. 4 times it confused fresh_apple as stale_apple and predicted fresh_orange as stale_orange 6 times. In two cases it failed to detect fruits correctly. It predicted stale_guava as stale_lime 4 times and fresh_orange as fresh_apple 4 times.

Precision, Recall, F1 score: For FreshDNN the precision of state apple and stale lime is 0.95 which is less than other classes and their recall is 0.97 and 0.99 respectively. The f1 score of FreshDNN is 0.98, recall and precision both are 0.98. **ROC-AUC score:** For the FreshDNN model, we can find the ROC-AUC score is 0.9997.

ROC-AUC curve:

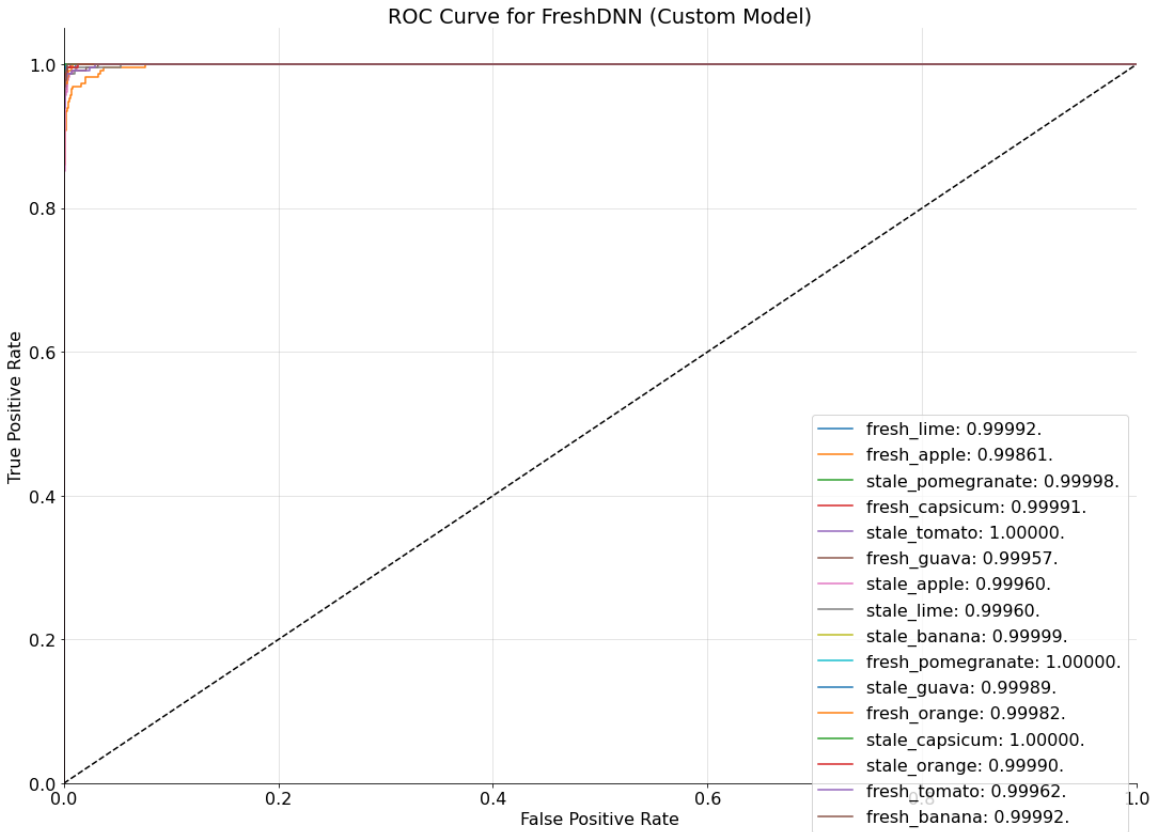


Figure 4.4: FreshDNN ROC-AUC Curve

The figure, explains the ROC curve area value for the different labels of our custom model, "FreshDNN". For the labels both stale_capsicum and fresh_pomegranate, the area under the curve is 1. Also, the area under the curve for the rest of the labels tends to be 1 which is excellent.

4.0.3 VGG19

In the pre-trained VGG19 model we took all of its layers but the last one. Consequently, none of the VGG19 designs changed. After assembling the model, we have begun training it. We need around 1 hour and 5mins to train on average. After training the model, the accuracy per epoch and loss per epoch graphs are shown below. The orange color represents test accuracy or loss (accuracy or loss) per epoch, whereas the blue color represents train accuracy or loss (val_accuracy or val_loss) per epoch

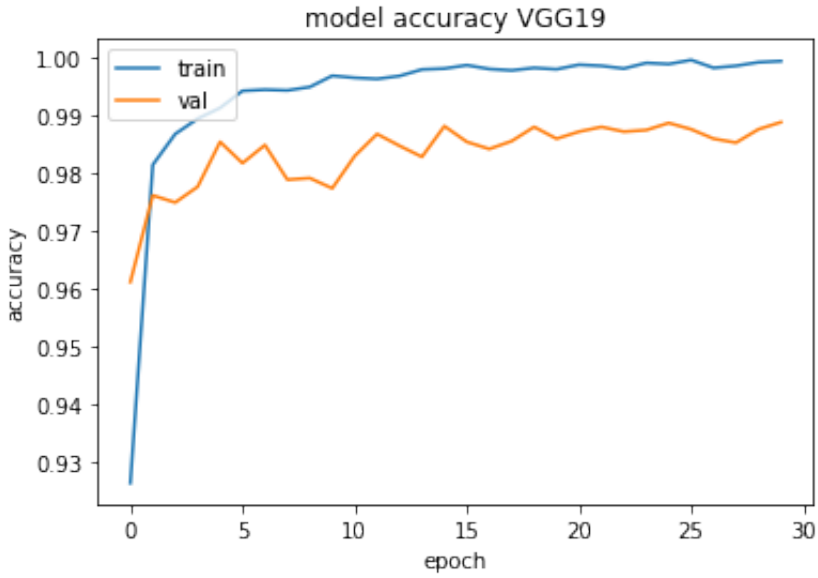


Figure 4.5: VGG19 Accuracy

The graph illustrates how accurate the model is in every epoch. For training, the accuracy is 92% in the first epoch, it is increasing at a higher rate in the first 10 epochs. Then it decreases a bit in the next few epochs, later it also has increased in a decent manner. From the graph, it is visible that for training the accuracy was just below 100% in the last epoch. But for the test case, the graph is different. There is a lot of fluctuation in the graph, there is a sudden up and down in the accuracy. The accuracy starts at 96% and after a lot of fluctuations it ends up at 98%.

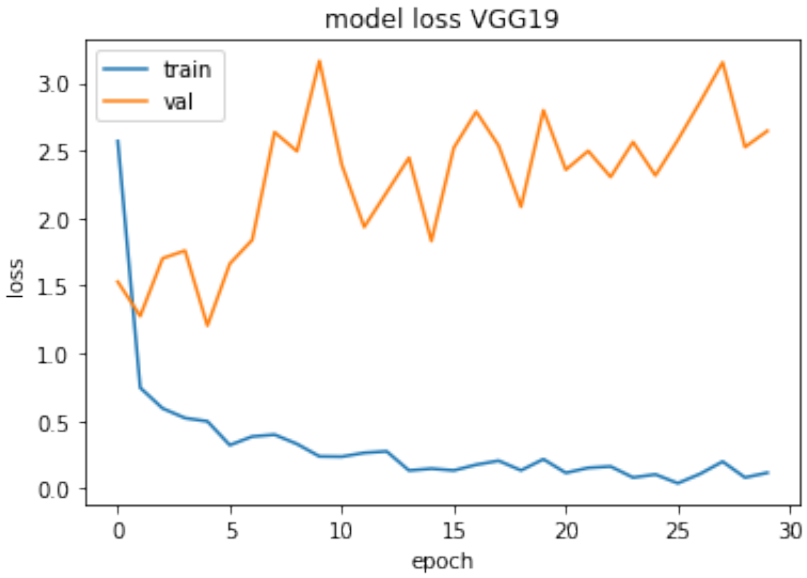
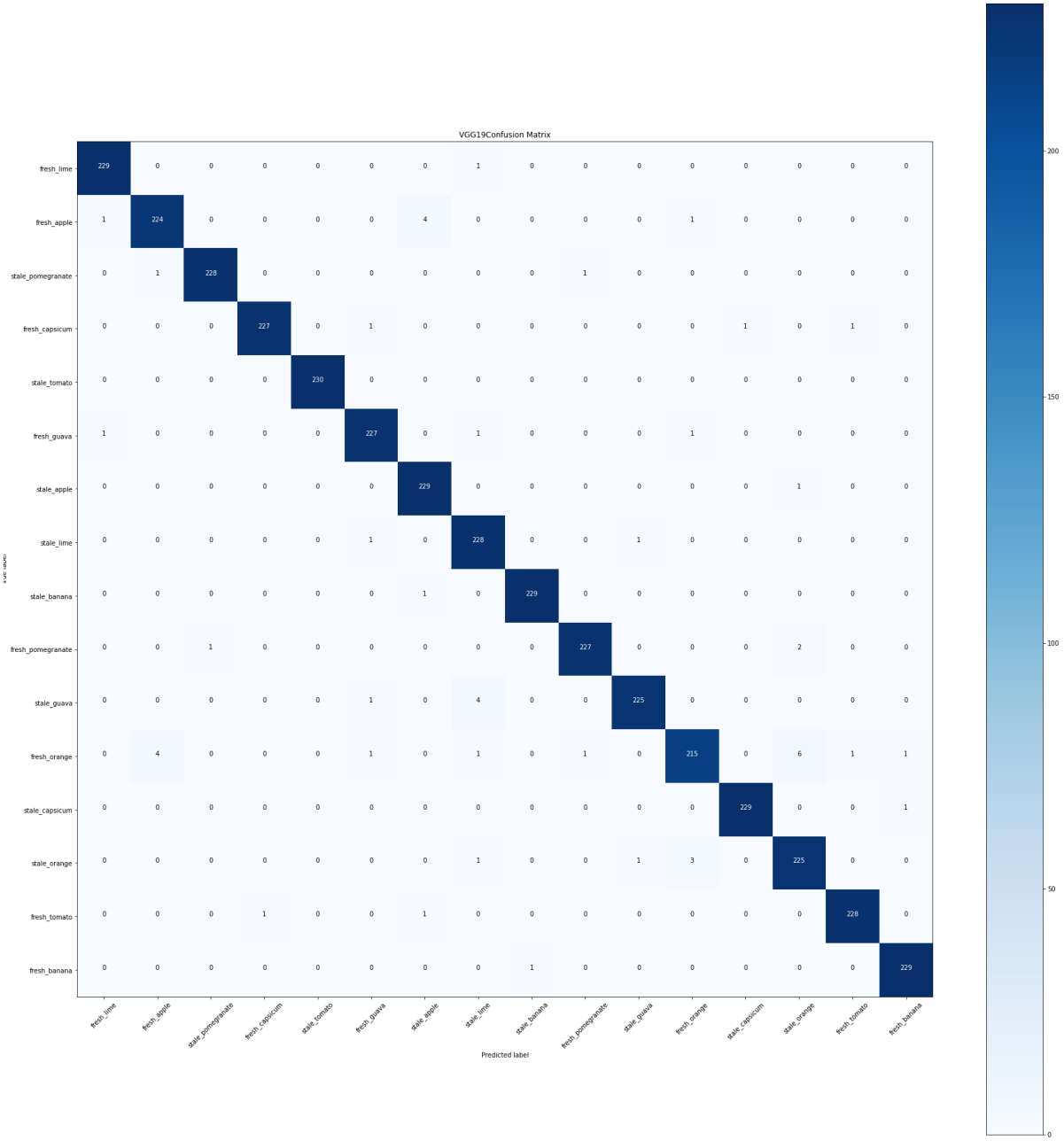


Figure 4.6: VGG19 Loss

The graph represents the relationship between loss in each epoch. For the training case, the initial loss is 2.57, then, it decreases decently epoch by epoch. Lastly, the loss has turned down to 0.11. In test cases, the loss has been fluctuating a lot. There is a sharp change in the graph. In the 7th epoch, the loss is 0.15 and in the 15th epoch it has increased up to 0.2, then again, it moves down. Lastly, the loss concludes to 0.1



The model has predicted fruits and its state perfectly in most of the cases. 4 times it confused fresh_apple as stale_apple and predicted fresh_orange as stale_orange 6 times. In two cases it failed to detect fruits correctly. It predicted stale_guava as stale_lime for 4 times and fresh_orange as fresh_apple for 4 times.

Precision, Recall, F1 score: For VGG19 the precision of state orange is 0.96 which is less than other classes and its recall is 0.98 and the f1 score is 0.97. Besides, for fresh orange, its precision is 0.96, recall 0.98, and f1 score 0.97. Apart from these, other classes' precision, recall, and f1 scores are satisfactory.

ROC-AUC score: For the VGG19 model we can find the ROC-AUC score is 0.994.

ROC-AUC curve:

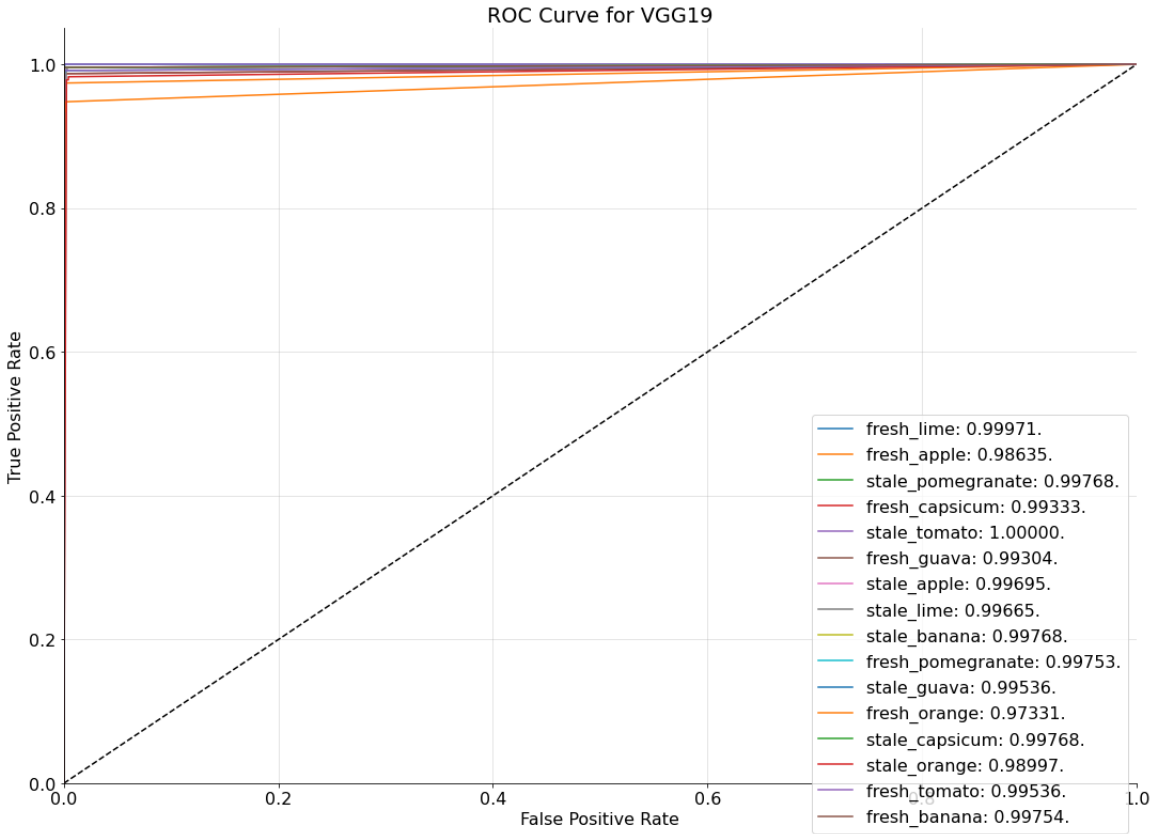


Figure 4.7: VGG19 ROC-AUC Curve

The figure depicts the ROC curve area value for the different labels of the models VGG19. For the label fresh_tomato, the area under the curve is 0.995 though it is 0.97331 for the label fresh_orange.

4.0.4 InceptionV3

In the pre-trained InceptionV3 model, we have taken all of its layers. Hence, all of the Inception V3 architectures remained the same. After compiling the model we have started to train our model. To train, we needed approximately 179.55 minutes on average. After training the model we have plotted below the accuracy per epoch graph and loss per epoche graph. The orange color denotes test accuracy or loss (accuracy or loss) and the blue color indicates train accuracy (val_accuracy) or loss (val_loss). To implement the code, we took help from [26]

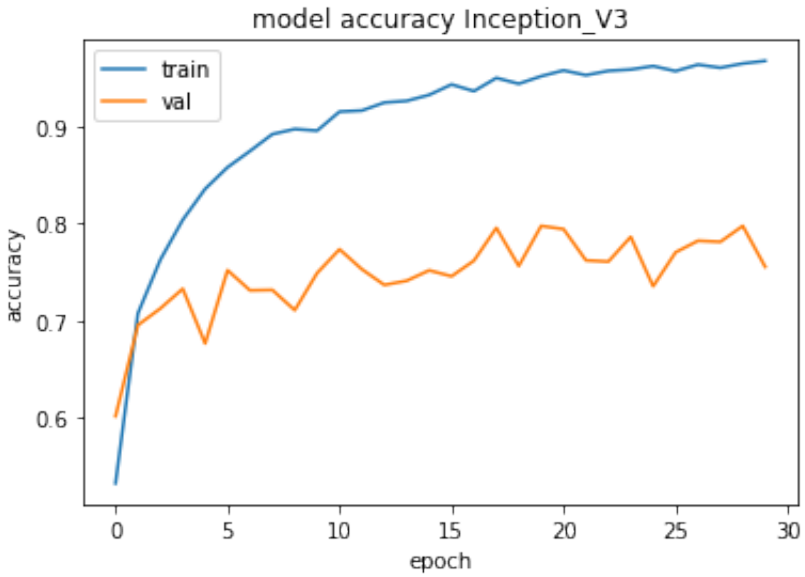


Figure 4.8: InceptionV3 Accuracy

The above graph shows the increase in accuracy with the increase of epoch over time. Initially, in the first epoch, our accuracy was 53.1% and val_accuracy was 60.1%. In the first four steps of epochs, we can see the validation accuracy and the training accuracy are increasing. Here, we got the highest accuracy in training in the 30 step which is 96.86%, and the highest accuracy in validation is 79.77% which we got in the 29th step.

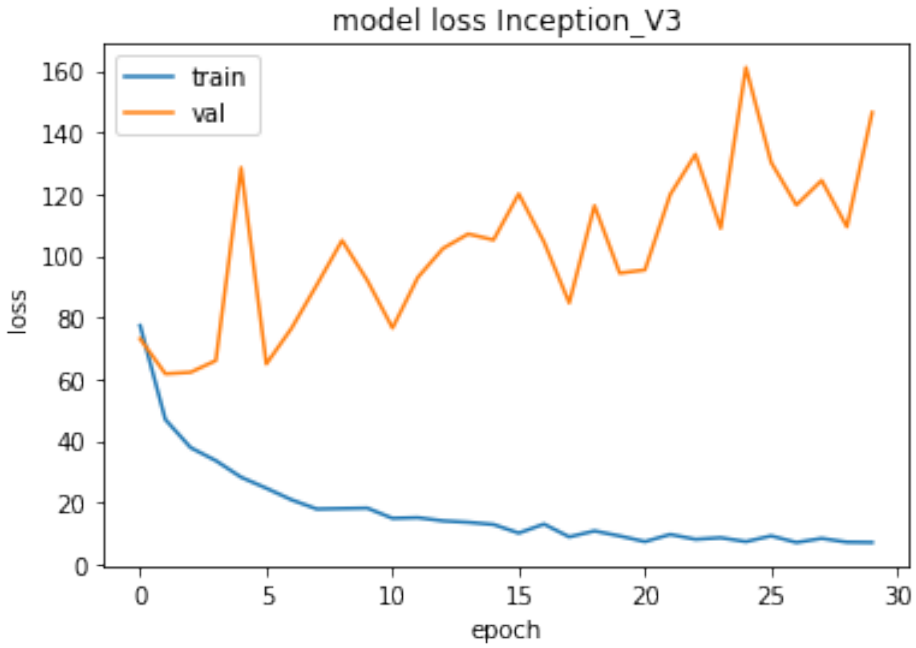
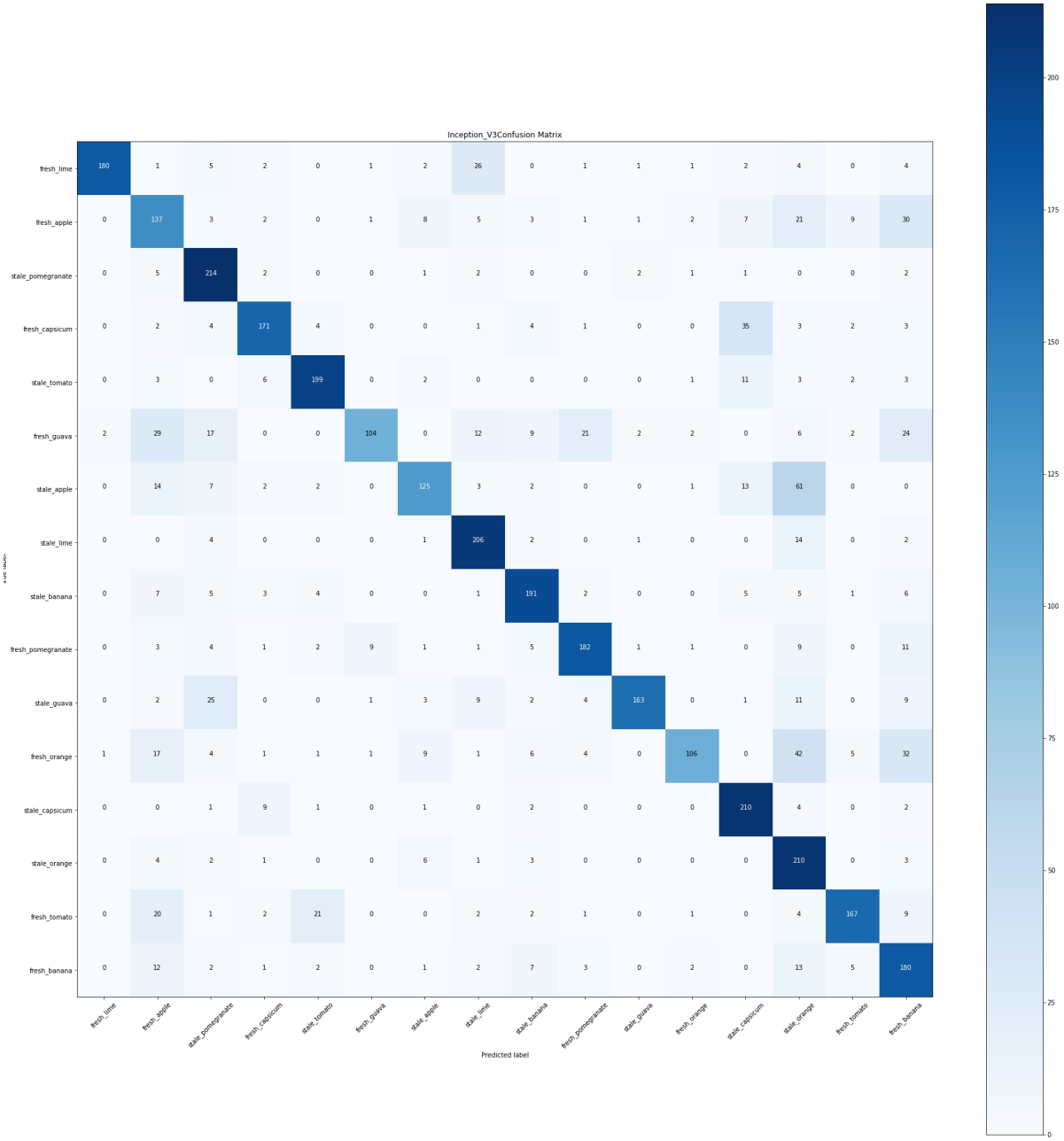


Figure 4.9: InceptionV3 Loss

The loss graph highlights the decrease of loss per epoch over time. In the beginning, the amount of train loss was 77.3% and the test loss was 56.6173. Here in the training, we have got the highest loss at the 1st step and the lowest loss at the 29th step which is 7.1. On the other hand, in validation, we have got the lowest loss 61.8 at the 2nd step and the highest loss 146.4 at the 30th step.



The model has predicted fresh_lime as stale_lime 26 times, fresh_capsicum as stale_capsicum 35 times, fresh_tomato as stale_tomato 25 times, and fresh_orange as stale_orange 42 times. In these cases, the model detected the fruit correctly but got confused about its state. On the other hand, the model confused fresh_apple as stale_orange 21 times and as fresh_banana as 30 times, stale_apple as stale_orange for 61 times, fresh_guava as fresh_pomegranate 21 times, and fresh_banana for 24 times. From the above data, it is clear that the model has performed very poorly both in detecting fruits correctly and also in detecting their state.

Precision, Recall, F1 score: For fresh apple, the precision is not satisfactory at all. It is 0.54 in precision, 0.6, and 0.56 on the basis of recall and f1 score. Besides, according to precision, and recall, the F1 score is 0.56,78, and 65 for fresh banana. Apart from fresh lime, the precision of 0.98 none of the classes are satisfactory in

terms of precision, recall, and f1 scores.

ROC-AUC score: We get ROC-AUC score for the model InceptionV3 is 0.878.

ROC-AUC curve:

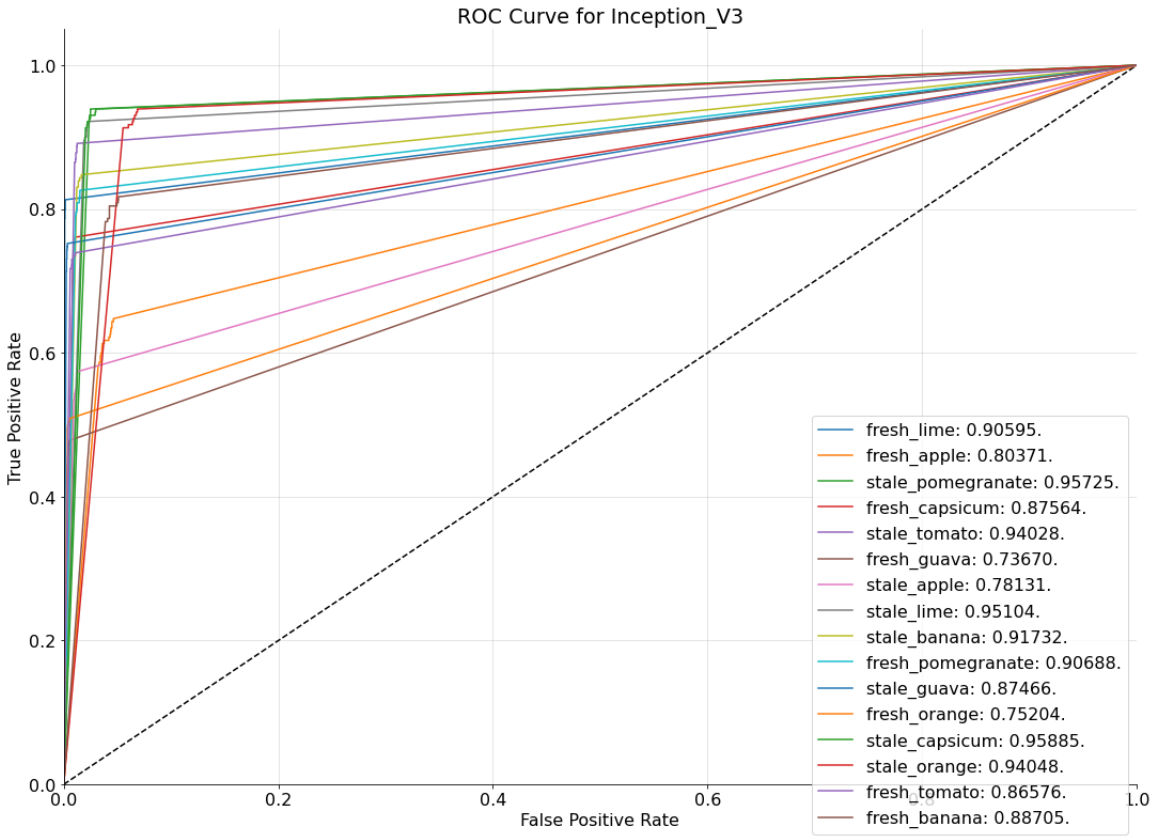


Figure 4.10: InceptionV3 ROC-AUC Curve

The figure visualizes the ROC curve area value of the model InceptionV3. For the label, stale_capsicumcum it is 0.95 which is the highest, and for the label fresh_guava, it is 0.73 which is the lowest.

4.0.5 EfficientNetV2L

In the beginning, we downloaded EfficientNetV2L from Tensorflow Keras. It was then decided to remove several layers to improve performance based on our findings from prior models. We completed the training in 1 hours and 6 minutes. A graph of training (accuracy) and testing (val_accuracy) accuracy was then created, with each epoch represented by a point on the graph. Both training (loss) and testing (val_loss) loss per epoch have been graphed in the same manner.

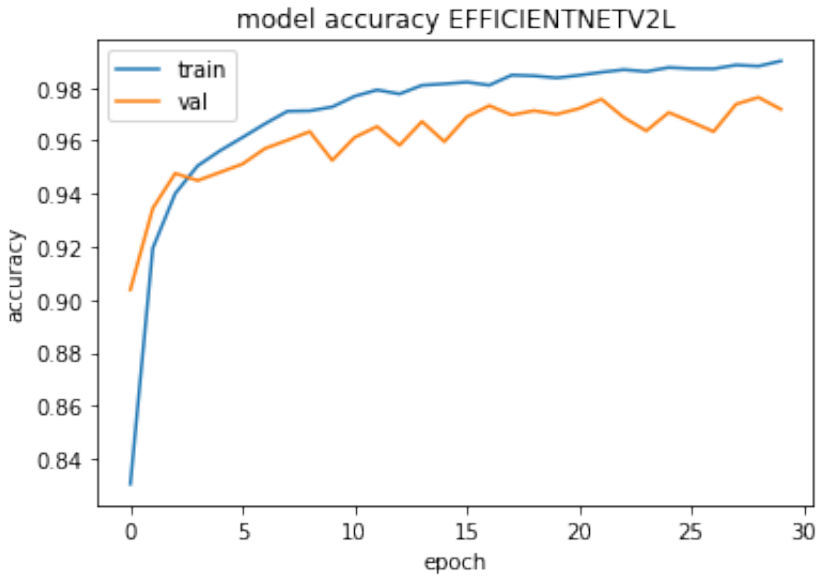


Figure 4.11: EfficientNetV2L Accuracy

The above graph depicts the accuracy of EfficientNetV2L over different epochs. In the graph, it is clear both training and validation accuracy are growing over the increase of epochs. Initially, the training accuracy is 83 percent whereas the validation accuracy is 90 percent. During 5 epochs, both the training and validation accuracy significantly rise to 96 percent and 94 percent consecutively. Lastly, the training and validation accuracies are 99% and 97%.

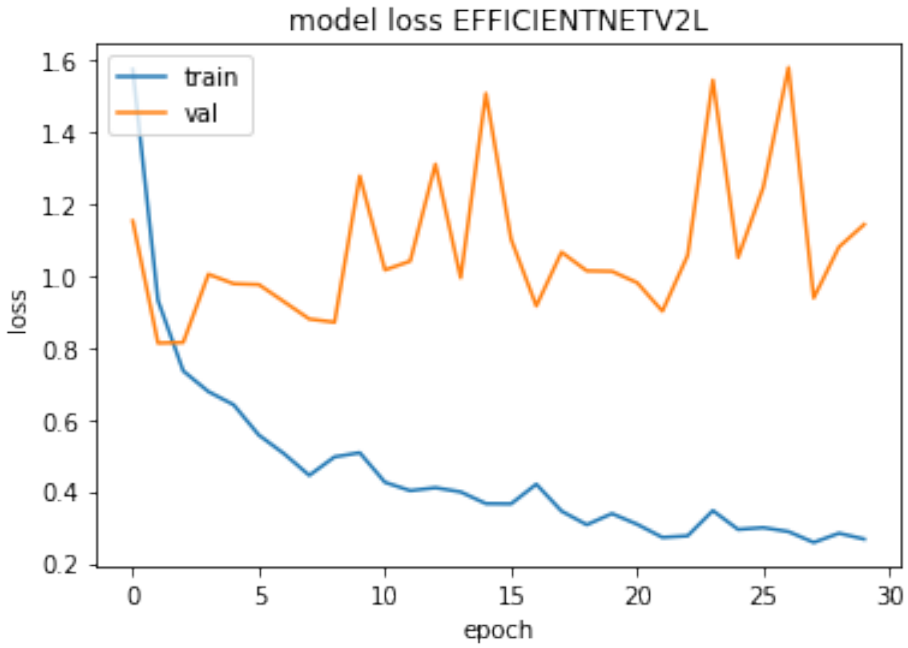


Figure 4.12: EfficientNetV2L Loss

The graph represents the loss with the increase of epochs. At the beginning both training and validation losses are high,1.57 and 1.15 consecutively. During 5 epoch training loss sharply falls to 0.64. Training loss declined to 0.36 and validation loss increased to 1.5 at 15 epochs. Noticeably, both training and testing loss continuously decrease over the rising of epochs.

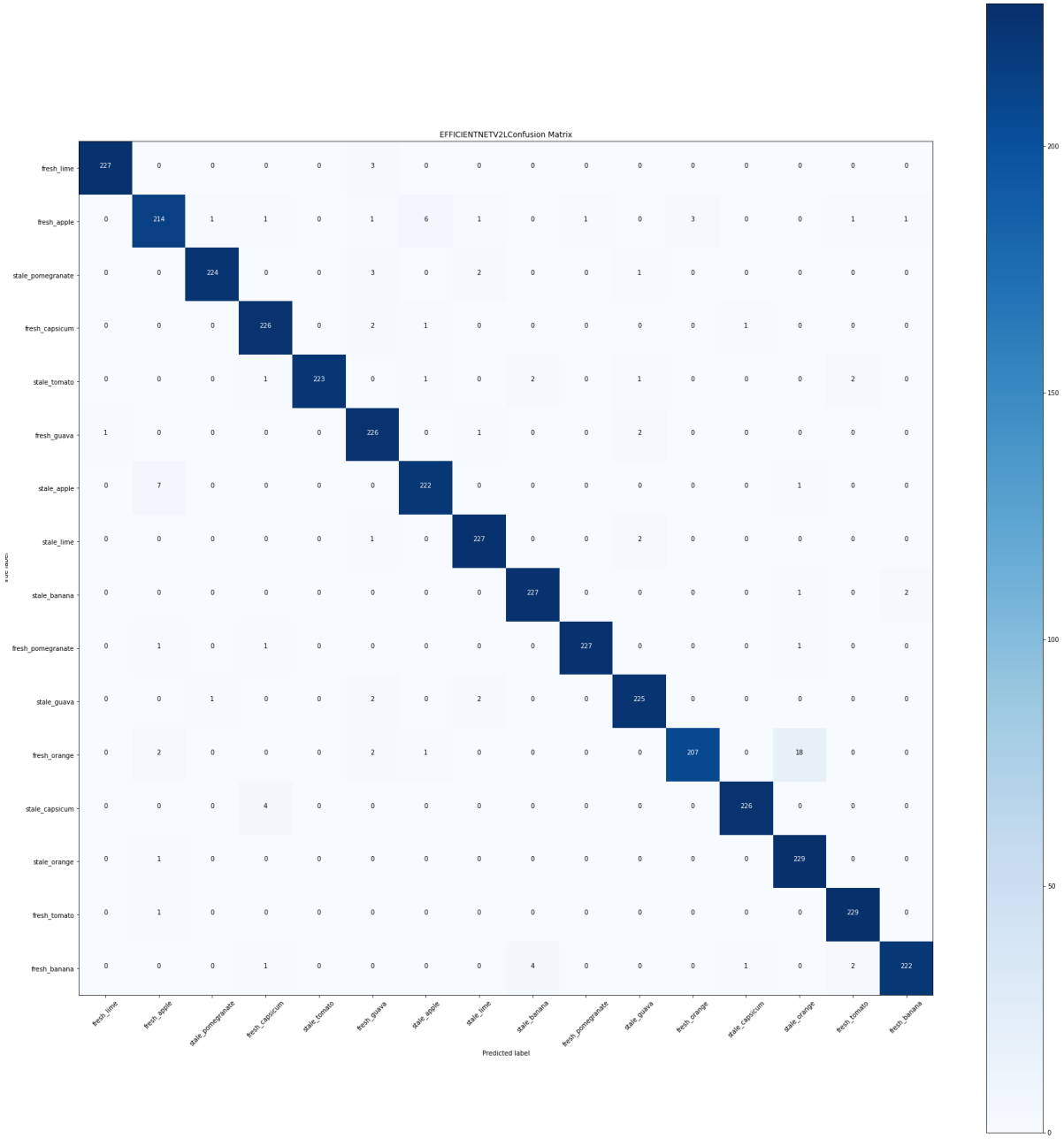


Figure 4.13: EfficientNetV2L Confusion Matrix

This model has predicted fresh_orange as stale_orange 18 times. The model confuses a fresh_orange with a stale_orange, but it can accurately guess which fruit it is. The algorithm performed almost perfectly while detecting the fruits but made mistakes in detecting the freshness state.

Precision, Recall, F1 score: In this model, the precision of fresh apple is not satisfactory. In terms of precision, it is 0.95. It is 93, and 0.94 for recall and f1 score consecutively. Besides, For stale orange, the precision is 0.92 and for fresh capsicum, it is 0.97,0.98, and 0.97 in terms of precision, recall, and f1 score respectively. Other classes' precision, recall, and f1 scores are satisfactory.

ROC-AUC score: EfficientNetV2L ROC-auc score is 0.9985.

ROC-AUC curve:

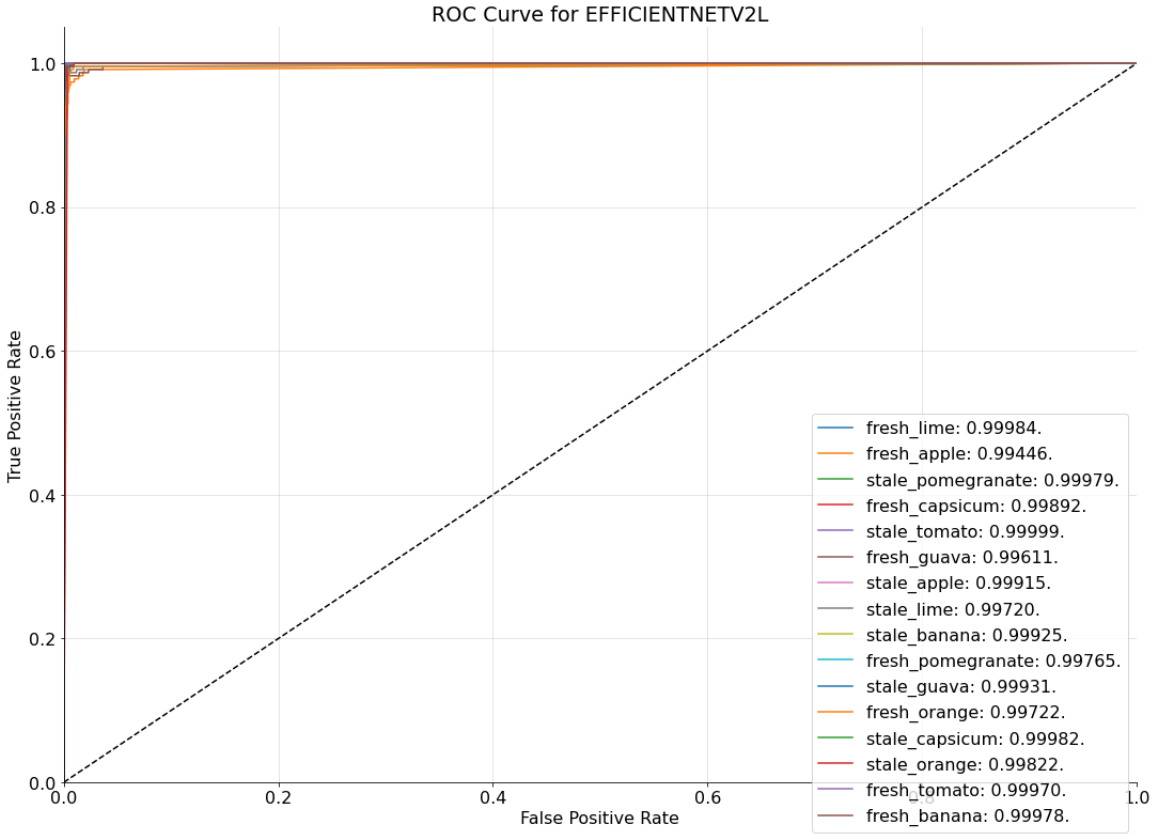


Figure 4.14: EfficientNetV2L ROC-AUC Curve

The figure visualizes, the ROC curve area of EfficientNetV2L, For all labels the curve area values tend to be 1.

4.0.6 ResNet152V2

According to the architecture mentioned in [27], we have downloaded the ResNet152V2 from tensorflow.keras.applications. Then we have taken all of its layers. As a result, no changes were made to the ResNet152V2 architecture. We have then begun the training process of building the model. We trained for an average of 292.41 minutes to complete 30 epochs. The following graphs show the model's accuracy with each epoch and its loss over per epoch after training. Test accuracy or loss (accuracy or loss) is represented by the color orange, whereas train accuracy (val_accuracy) or loss (val_loss) is represented by the color blue.

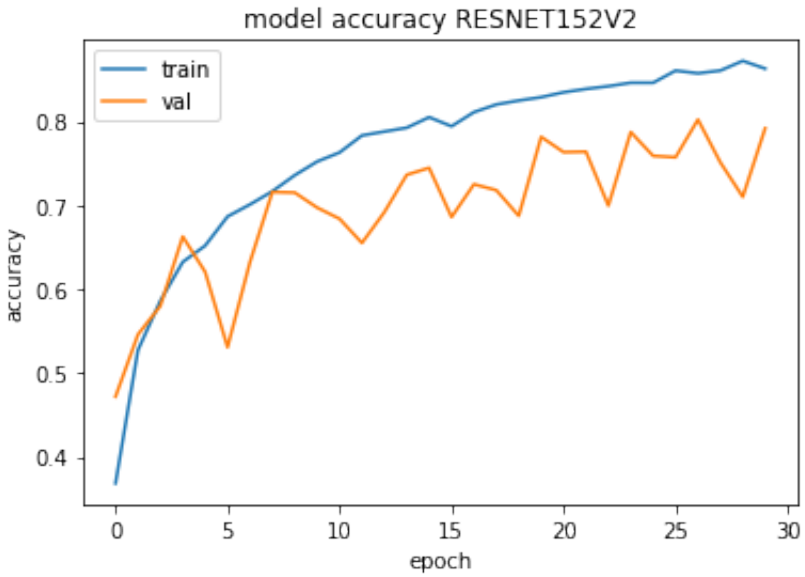


Figure 4.15: ResNet152V2 Accuracy

The graph shows the relationship between accuracy over epochs. Initially, in the first epoch, the training accuracy was 36%. In the second epoch, it is visible in the graph that there is a fluctuation in the accuracy and the training accuracy jumped to 52%. Later, as the epoch goes the training accuracy also increased at a moderate rate. Lastly, in the 30 epoch (last) the training accuracy is turned to 86%. Besides, in the graph, it is visible that initially, the validation accuracy was 47%, and then in the next few epochs validation accuracy is decreasing and then it is also increasing. In the graph, it is clear that testing the validation accuracy is fluctuating up and down and lastly the validation accuracy is landed at 79%.

Figure 4.16: ResNet152V2 Loss

The graph demonstrates the relationship between loss per epoch. Firstly, the training loss is 3306. In the next epoch, the training loss is decreasing and is turned down to 2108. Over epochs, the training loss is decreasing in small and sometimes it also is increasing a few in rate. That the loss put down to 924.75. For the validation case, the loss is initially 2382.61 and the significant rise in the loss is turned up to 3496.85 in the 6th epoch and it decreases. In the graph, it is vivid that there is always up and down in the test loss case. Lastly, the test loss turned to 1805.1.

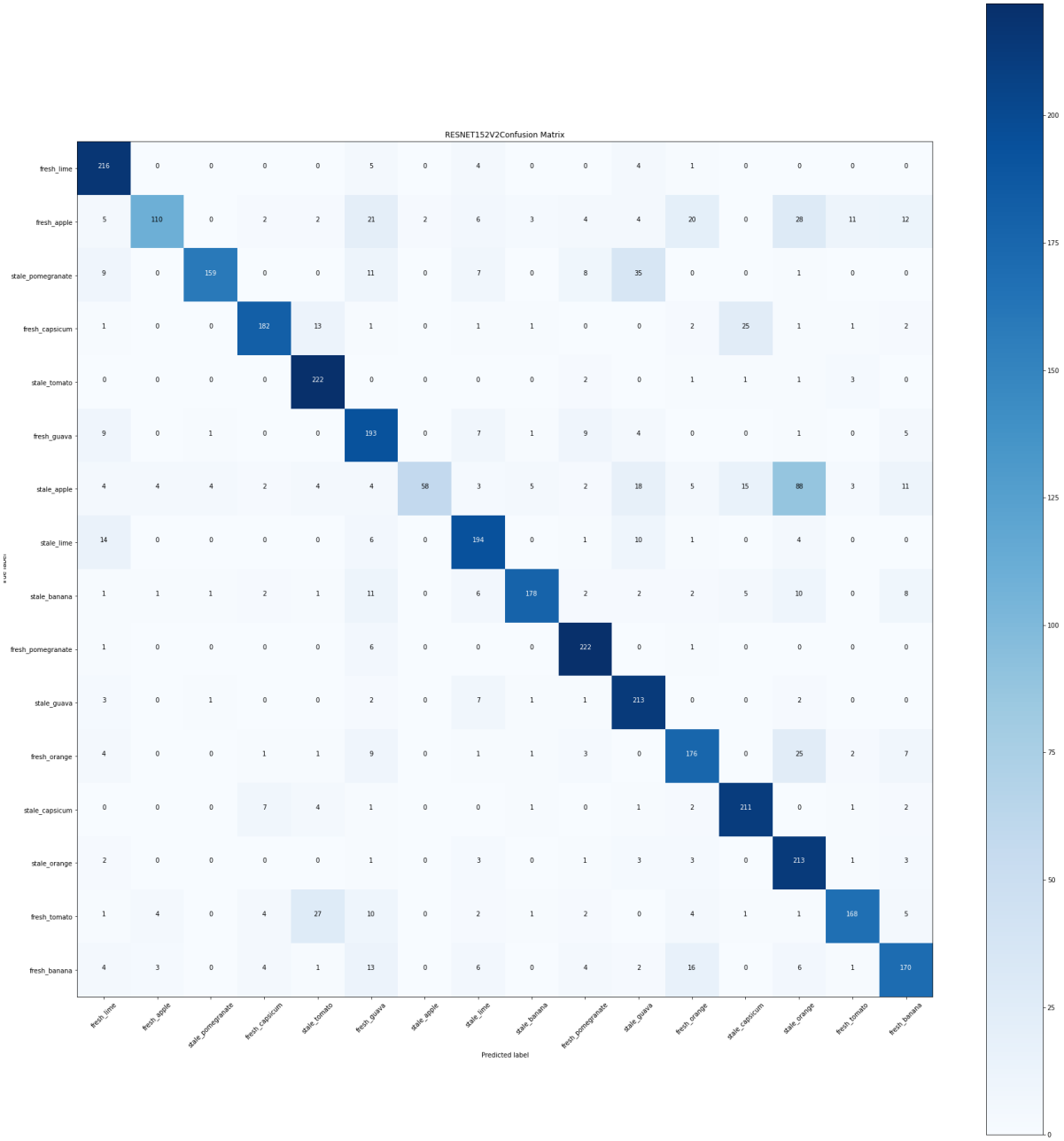


Figure 4.17: ResNet152V2 Confusion Matrix

This model has predicted fresh_tomato as stale_tomato 27 times and stale_lime as fresh_lime for 14 times, fresh_capsicum as stale_capsicum 25 times. On top of that, the model has predicted fresh_apple as fresh_guava 21 times, fresh_orange 20 times, stale_orange 28 times, stale_pomegranate as stale_guava 35 times, and stale_apple as stale_orange 88 times. In most of the cases, the model confused the fruit and in some cases, it confused its state.

Precision, Recall, F1 score: The precision for fresh orange is 0.75, recall is 0.77 and the f1 score is 0.76. Besides, for fresh bananas, it is 0.76, 74, and 75 in terms of precision, recall, and f1 score. For stale banana, the precision is 0.56 which is the lowest among all classes. Apart from these the classification reports for other classes are satisfactory.

ROC-AUC score: For the ResNet152V2 model we can find the ROC-AUC score is 0.886.

ROC-AUC curve:

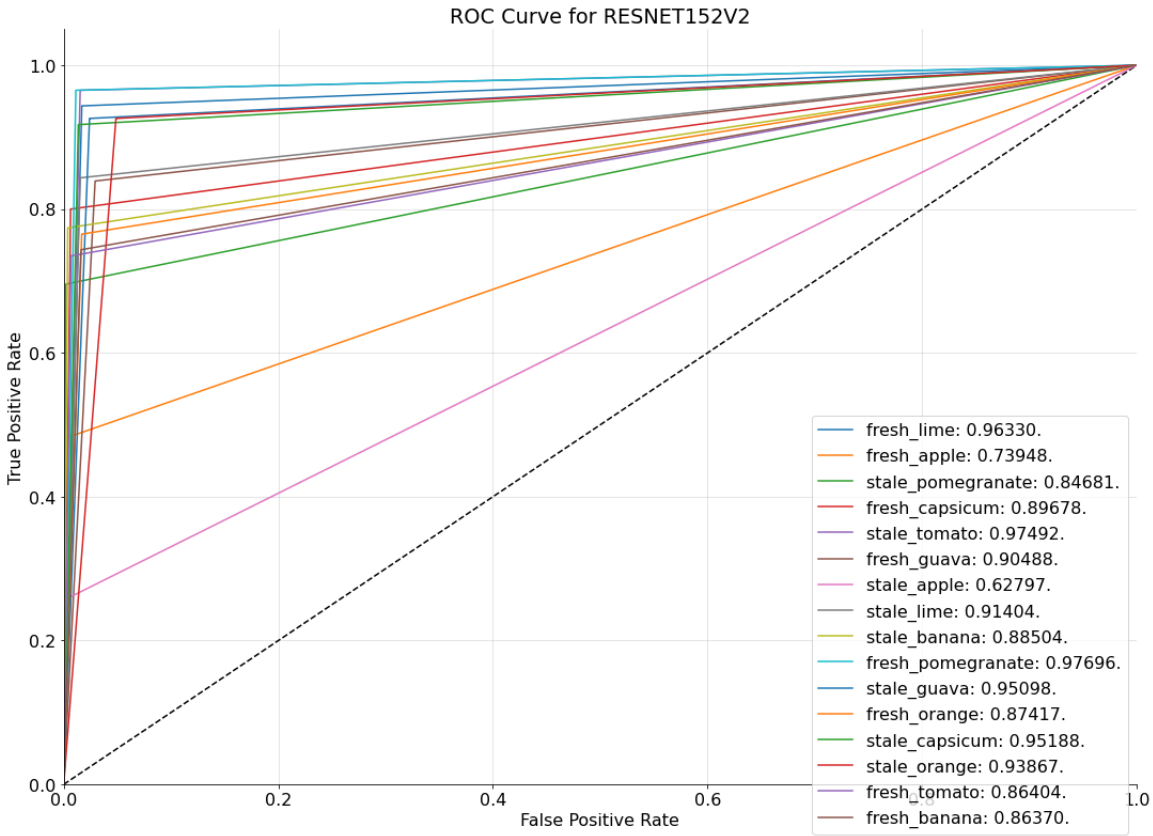


Figure 4.18: ResNet152V2 ROC-AUC Curve

The figure displays, the ROC curve area value of ResNet152V2, For the labels both fresh_pomagranate is 0.97 but for the label stale_apple it is 0.62. Other, labels have the ROC curve area value in between.

4.0.7 Xception

According to the paper [28], to implement Xception, we have to import Xception Large from tf.keras.applications.xception.Xception(). Then, based on what we learned from the previous models, we took away some layers to get the model ready for better performance. We only needed 12.21 minutes. After that, we made a graph of training (accuracy) and testing (val_accuracy) accuracy per epoch. In the same way, we've made a graph of loss per epoch for both training (loss) and testing (val_loss).

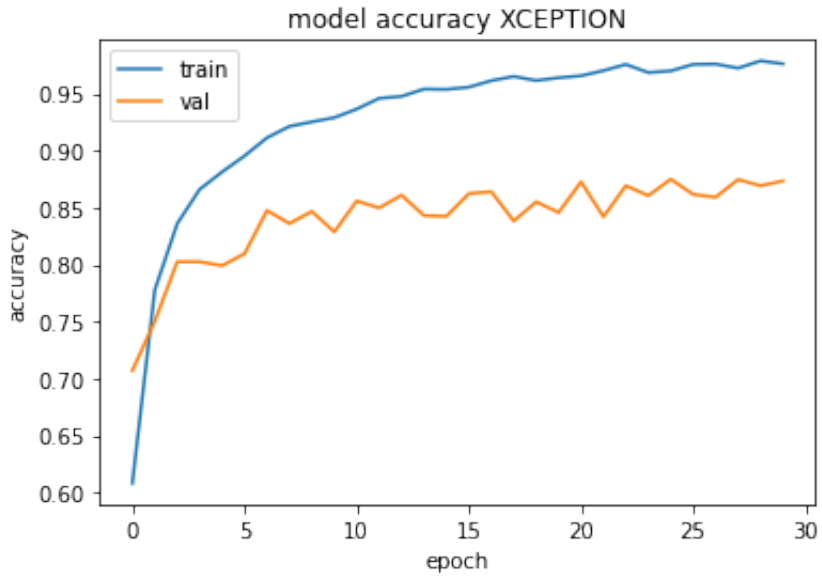


Figure 4.19: Xception Accuracy

The graph portrays how the accuracy increases in each epoch for Xception. To train, the accuracy begins with nearly 61 %. As the epoch goes the accuracy rate also moves upward. We can find the accuracy becomes 92.9%,96.41%, and 97.63% when the number of epochs is 10,20,30. Almost the same situation replicates in validation cases where the accuracy has become 70.7,82.88% and 87.35 % in the gap of every 10 epochs.

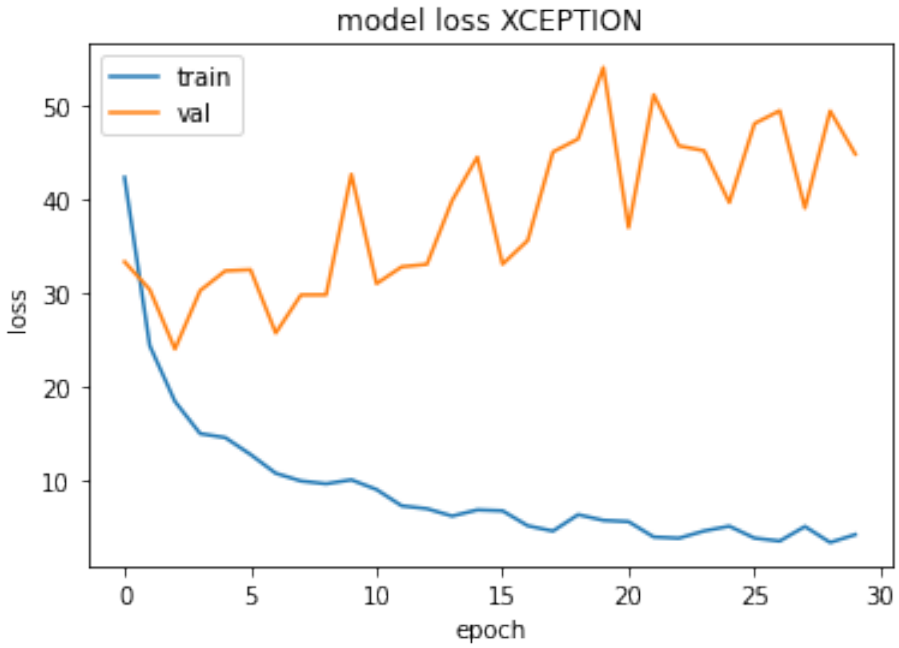


Figure 4.20: Xception Loss

The graph visualizes the loss in each epoch for Xception. In training, the loss is 42 in the 1st epoch then the loss is falling down drastically. On the 10th, and 20th epoch it decreases to 9.9 . 5.5 Lastly, it turned down to 4.1. For validation loss it was 33.26 initially, then it increased, and lastly, it turned up to 44.7.

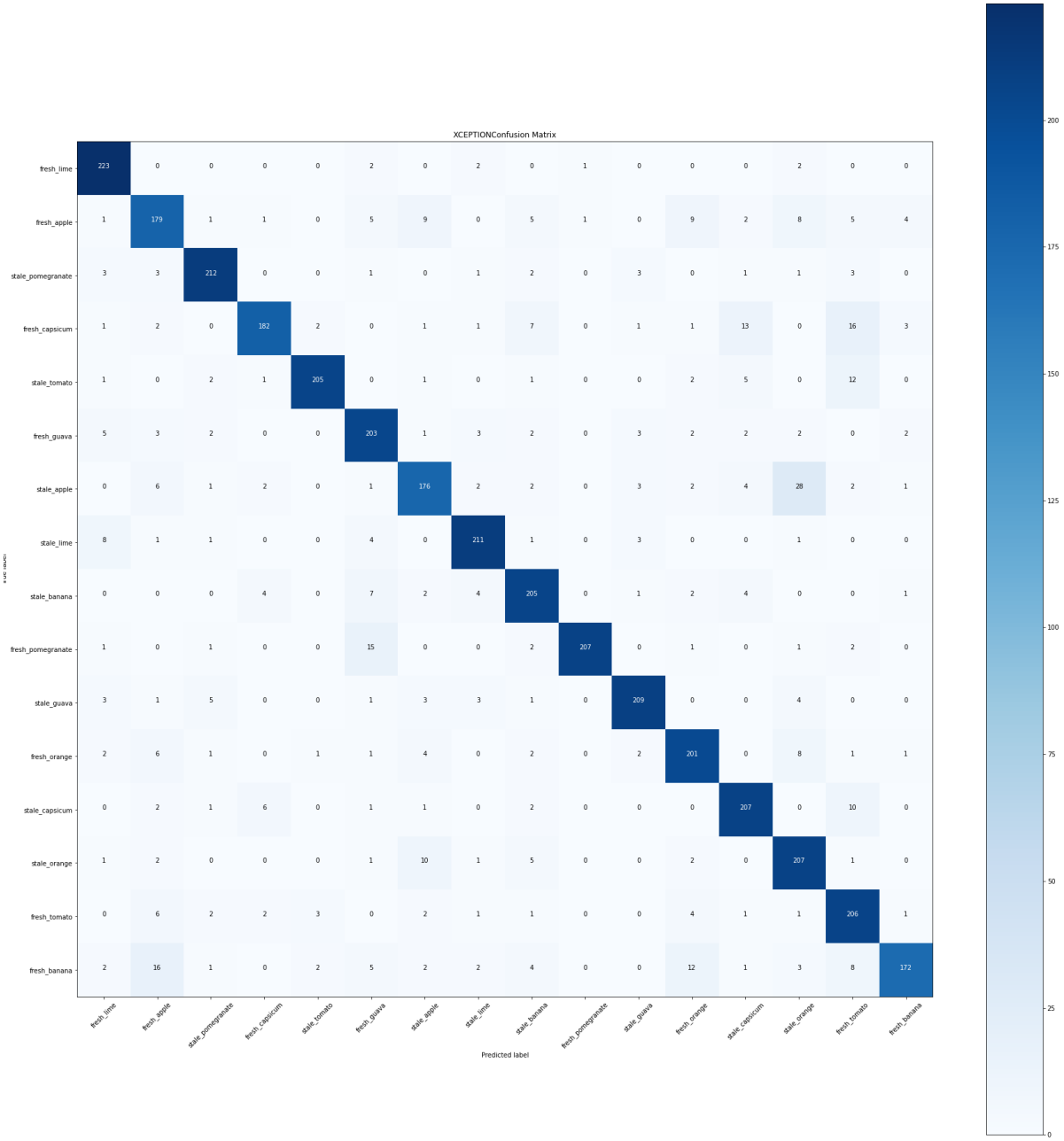


Figure 4.21: Xception Confusion Matrix

This model has predicted fresh_capsicum as stale_capsicum 13 times and stale_tomato as fresh_tomato 12 times. On the other hand, It predicted fresh_capsicum as fresh_tomato 16 times, stale_apple as stale_orange 28 times, fresh_pomegranate as fresh_guava 15 times and fresh_banana as fresh_apple 16 times. This model performed well while detecting the fruits but made significant errors in detecting their state.

Precision, Recall, F1 score: For Xception the precision is not satisfactory for fresh apples, it is 0.79,0.78,0.78 in terms of precision, recall, and f1 score. For stale bananas it is 0.89,0.85,0.87. Besides, for fresh tomatoes, the precision(0.77) is not satisfactory. Apart from these other classes classification reports are ok.

ROC-AUC score: The ROC-AUC score of the Xception model is 0.946.

ROC-AUC curve:

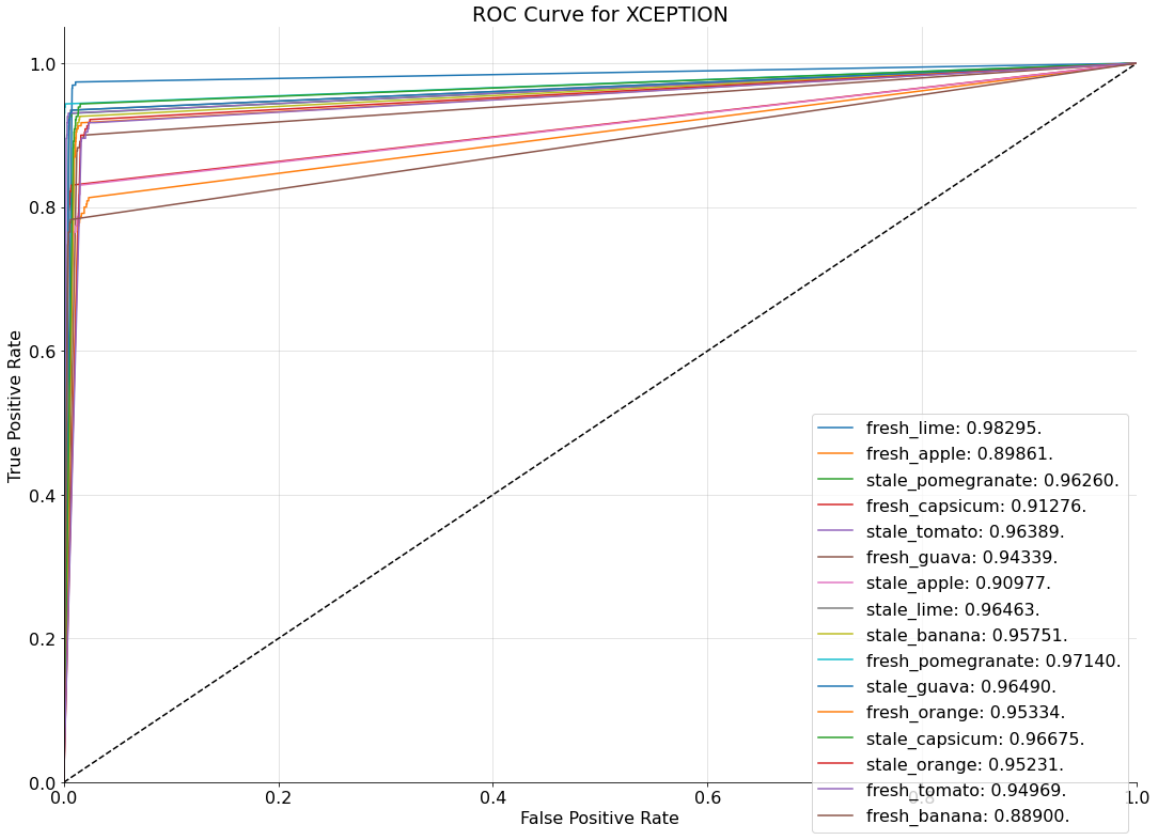


Figure 4.22: Xception ROC-AUC Curve

The figure explains, the ROC curve area of Xception. For the label fresh_lime it is 0.98 but for the label fresh_banana it is 0.889. Other, labels have the ROC curve area value in between.

4.0.8 DenseNet201

Based on what we had learned from earlier models, it was then decided to remove a few layers to improve performance. 1 hours and 7 minutes passed before we were done with the training. 2 graphs were plotted like previous models. We took help from [29] to implement the code.

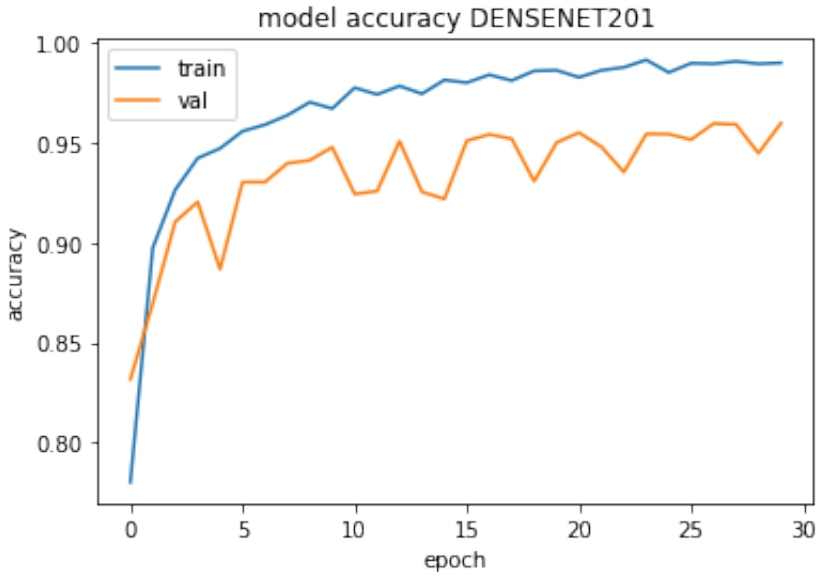
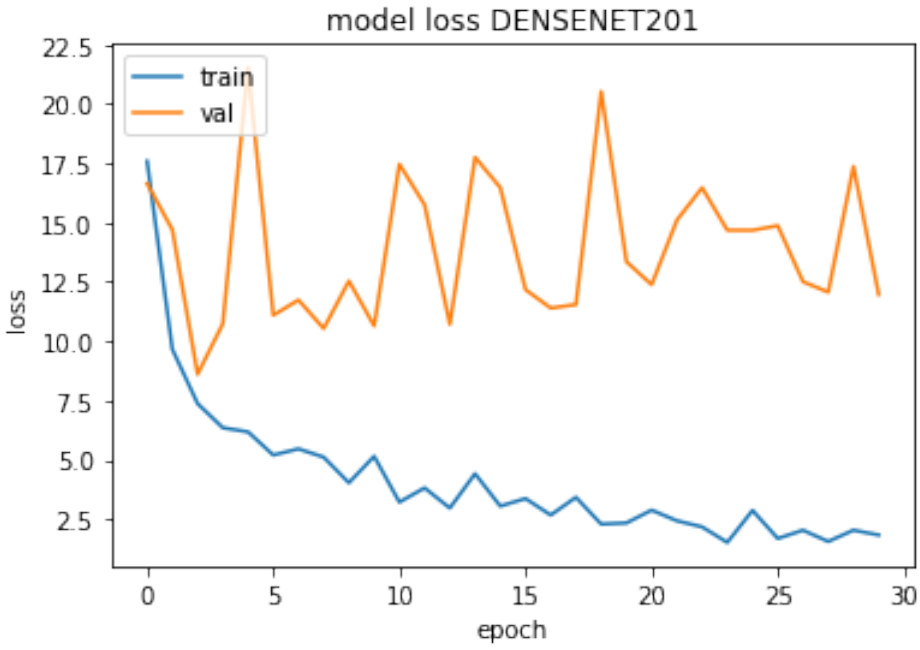


Figure 4.23: DenseNet201 Accuracy

In the above graph, we have plotted the epoch on the x-axis and accuracy on the y-axis. This graph shows the increase in accuracy with the increase of epoch over time. At the first epoch, the initial training accuracy was 78%. The training accuracy increased moderately till 10 epochs and it was 97% at 10 epochs. Then the training accuracy rose slightly and at the end of 30 epochs, it was close to 99%. The initial validation accuracy was 83%. it fluctuate up and down till 15 epochs and at the end of 15 epochs, the validation accuracy was below 93%. The validation accuracy changed a bit and it was near 96% at the end of 30 epochs.



N

Figure 4.24: DenseNet201 Loss

Then the loss graph, the x-axis, and the y-axis represent epoch and loss respectively. The graph explains the decrease of loss per epoch over time. Initially, the amount of train loss was 17.59 and the validation loss was 16.64. Then the training loss declined sharply and it was 5.148 at 10 epoch. The training loss changed slightly between 20 to 30 epochs and it was 1.8 at the end of 30 epochs.The validation loss fluctuate up and down between 15 to 25 epochs and it was 14.67 at 25 epochs and it decreased moderately , at the end of 30 epochs the validation loss was near 12%.

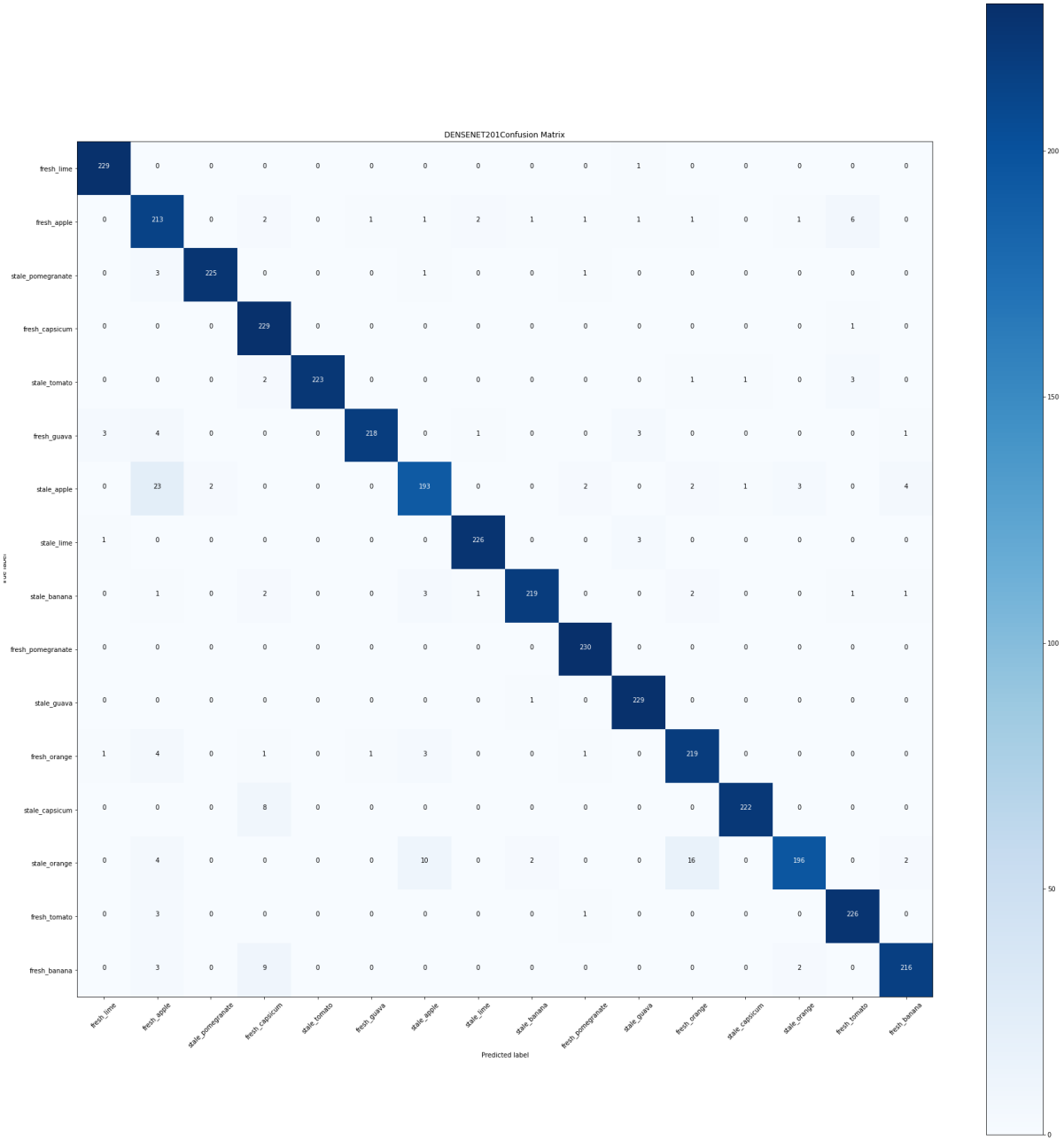


Figure 4.25: DenseNet201 Confusion Matrix

This model has predicted stale_apple as fresh_apple for 23 times and stale_capsicum as fresh_capsicum for 8 times. In both cases, the model detected the fruits correctly but got confused in determining the freshness state. The model hardly failed to detect the fruit. A mention-worthy case was it considered stale_orange as stale_apple for 10 times.

Precision, Recall, F1 score: For this model, the precision for fresh apples is 0.83 which is the lowest among all classes, the recall, and f1 scores are 0.93 and 0.87. Besides, for stale apples, it is 0.94, 0.84, 0.88 in terms of precision,recall, and f1 score. For fresh tomatoes, the precision is 0.95. Apart from these other classes have satisfactory classification reports.

ROC-AUC score: For the DenseNet201 model, we can find the ROC-AUC score is 0.981.

ROC-AUC curve:

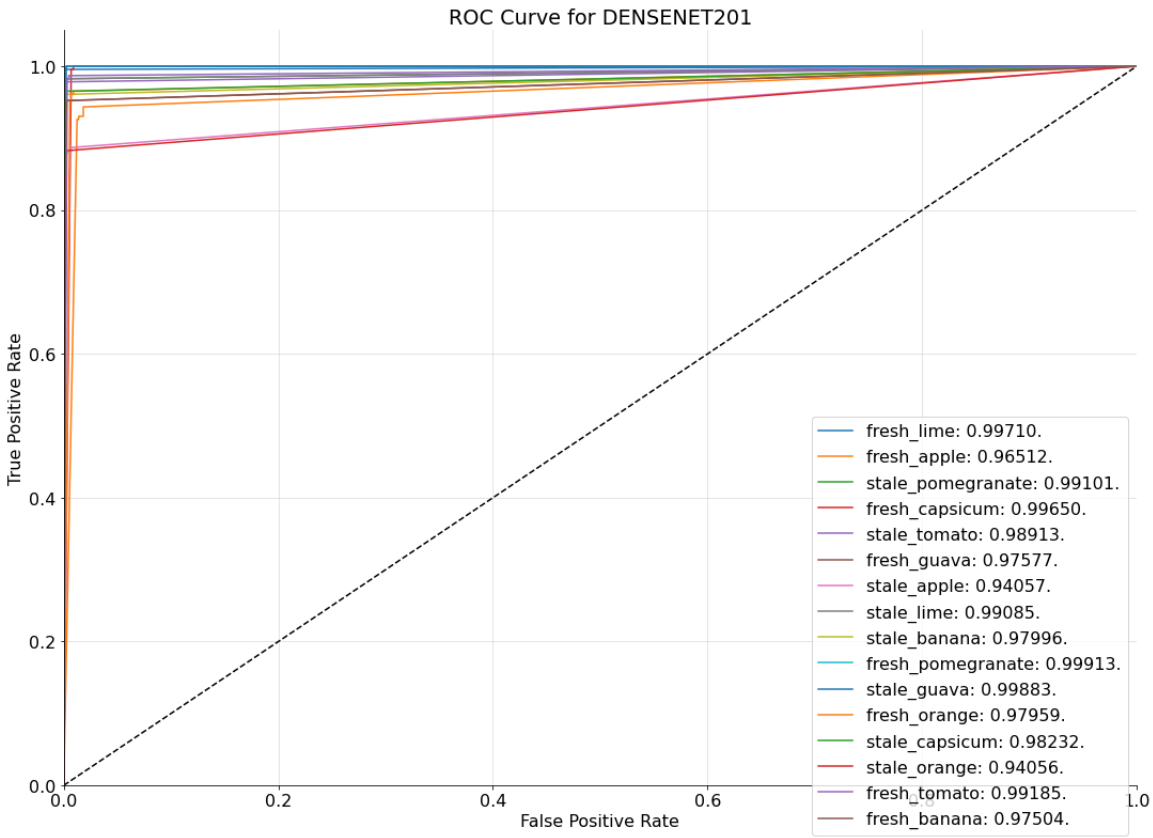


Figure 4.26: DenseNet201 ROC-AUC Curve

The figure portrays the ROC curve area value for DenseNet201. The ROC score is 0.94 for the label stale_orange (lowest) and it is 0.9970 for fresh_pomegranate(highest). And other labels' scores are in between.

4.0.9 MobileNetV2

Similar to MobileNetV1, we imported the model from tf.keras.applications.mobilenet_v2.MobileNetV2(). In the same way as the MobileNet version 1, we have updated the model such that it would only include the first layer to 2nd last layer of the original MobileNet version 2. We decided not to add the last layer after reviewing the summary of the original model. Instead, we'll include everything up to the "final global average pooling layer," which has improved the model's performance on our dataset. After eliminating the last layer, we observed that training the remaining 23 layers yields reasonably decent results, therefore we trained our dataset using just the remaining last 23 layers. We finished the 30 epochs in 1 hours and 6 minutes. We can examine how the model's accuracy varies throughout epochs and how much it loses at each epoch of training using these graphs. Orange indicates test accuracy or loss (accuracy or loss), whereas blue represents train accuracy (val_accuracy) or loss (val_loss).

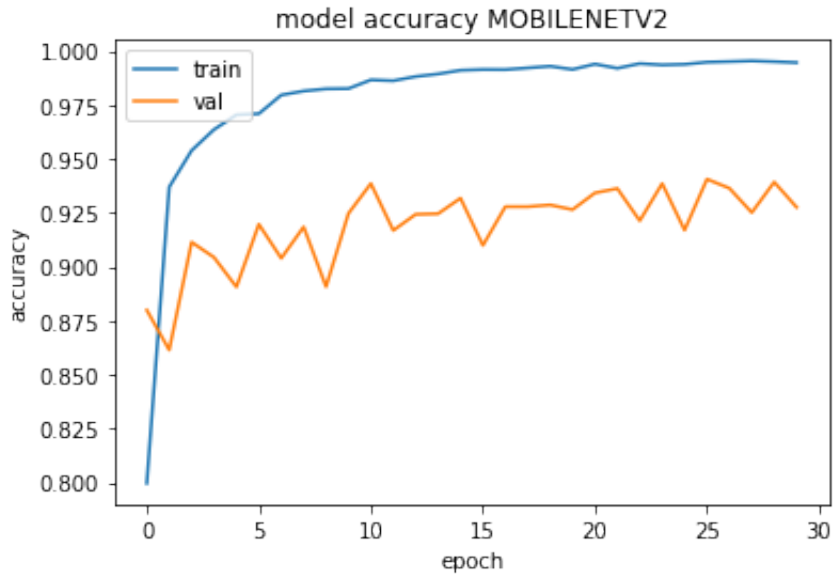


Figure 4.27: MobileNetV2 Accuracy

The graph shows the accuracy of training and validation data over time. In the 1st epoch, the training accuracy is just about 80&. In the next epoch, the accuracy climbed up to 93&. Then it is increased to moderate epoch by epoch and it turns to 98&,99& when it is the 10th and 20th epoch. Lastly, it stayed at 99.4&. Again, for validation accuracy initially, it is just about 88&. Then as the epoch goes up the validation accuracy has increased moderately. It has turned to 92& lastly.

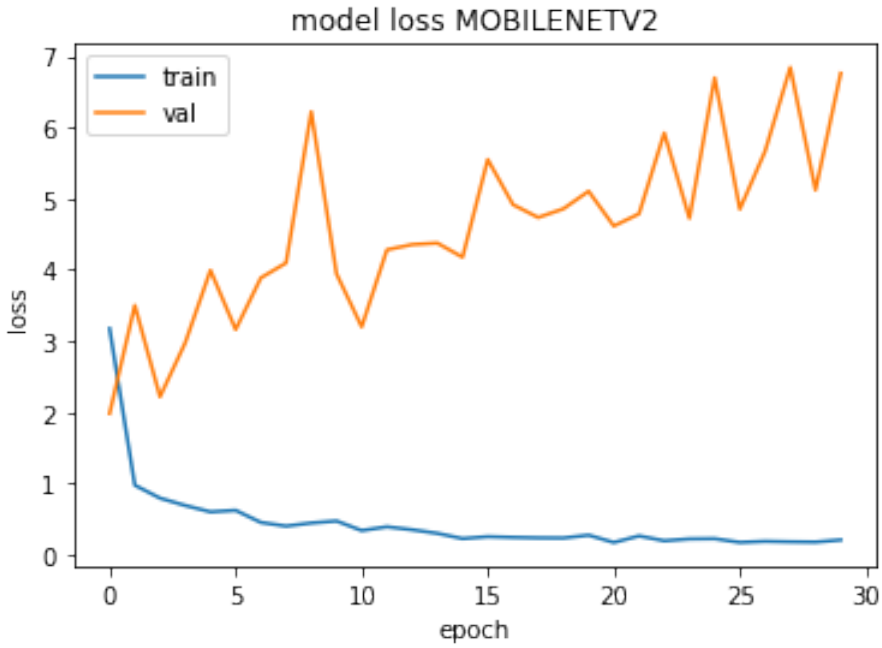


Figure 4.28: MobileNetV2 Loss

The graph demonstrates the relationship between loss over time. Initially, the loss was 3.17. Then, it decreased rapidly and turned down to 0.97 in the second epoch. In the graph it is shown that as the epoch goes on, the loss decreases and turns into 0.44 and 0.27 when it is the 10 and 20 epochs respectively. Lastly, it becomes 0.2. For validation loss, the trend is the opposite: it has turned up to 6.75(last epoch) from 1.98 (first epoch).

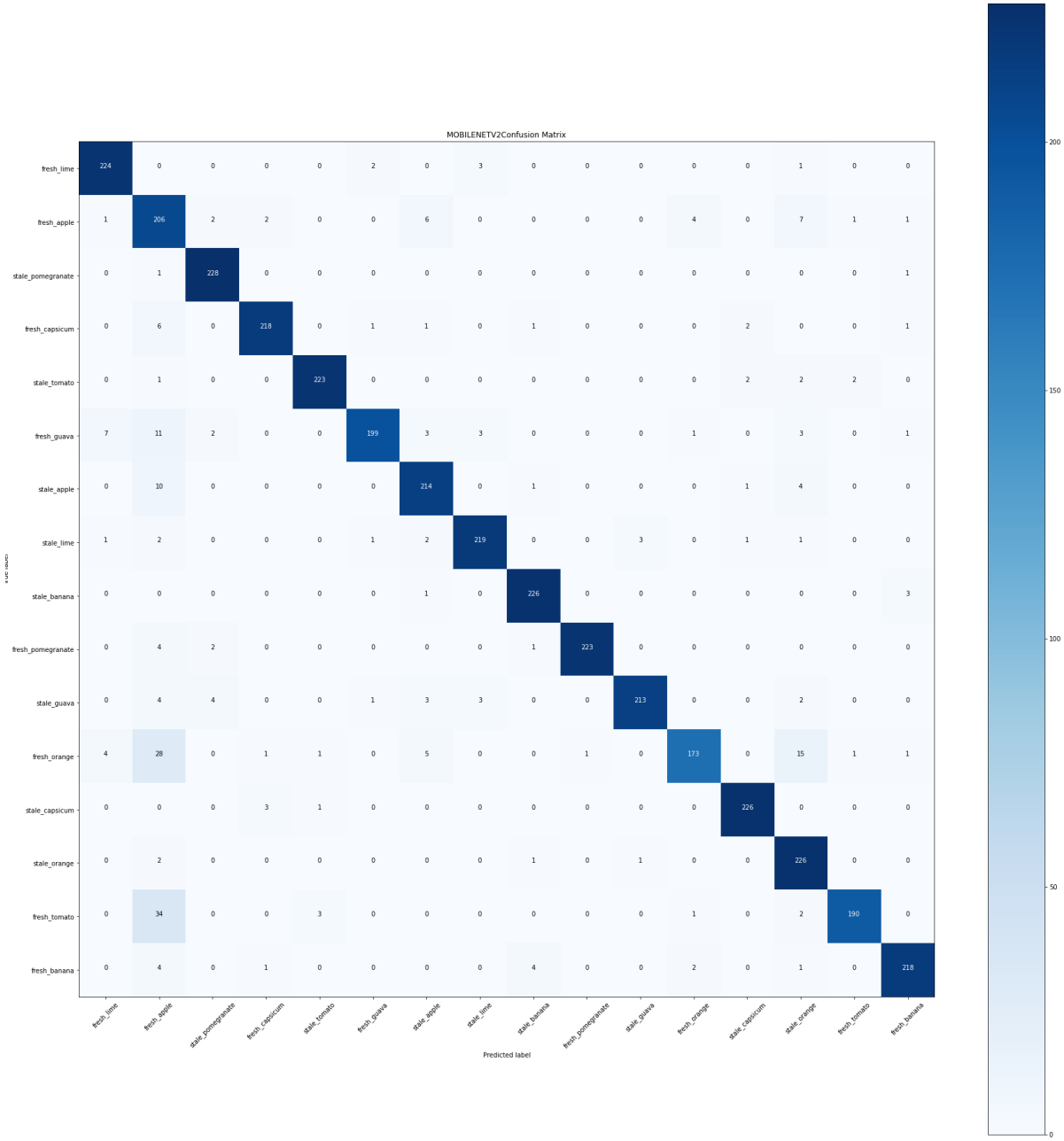


Figure 4.29: MobileNetV2 Confusion Matrix

This model has predicted fresh_guava as fresh_apple 11 times, fresh_orange as fresh_apple for 28 times and fresh_tomato as fresh_apple for 34 times. In these cases, the model made mistakes to detect the fruit properly. However, the model has predicted stale_apple as fresh_apple for 10 times and fresh-orange as stale_orange 15 times. This model has performed poorly in detecting the fruit properly than its state.

Precision, Recall, F1 score: For fresh apples 0.66 is the precision which is the lowest among all classes .0.83 is the recall for fresh tomato which is the lowest among all classes. For fresh lime, stale lime, and fresh banana the precision is 0.95,0.96,0.96. For fresh banana, and fresh tomato the recall is 0.95,0.83 and for fresh orange, it is 0.75. The f1 scores are 0.76 and 0.84 for fresh apple and fresh orange.

ROC-AUC score: We find the ROC-AUC score for the model MobileNetV2 is 0.981.

ROC-AUC curve:

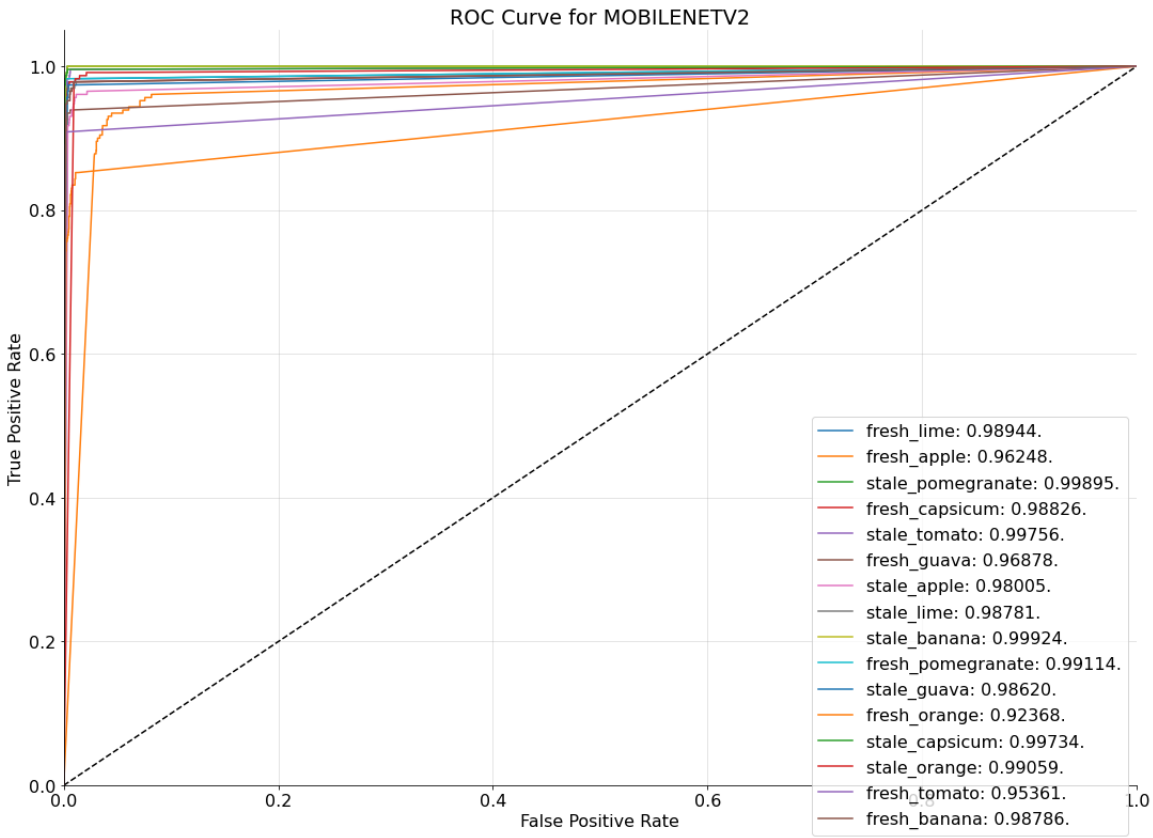


Figure 4.30: MobileNetV2 ROC-AUC Curve

The figure depicts the ROC curve area value of MobileNetV2, For the labels stale_pomagranate, and stale_banana it is just near 1 but for the label fresh_orange it is 0.92. Other, labels have also very good ROC curve area value.

4.1 Comparison of Different Models

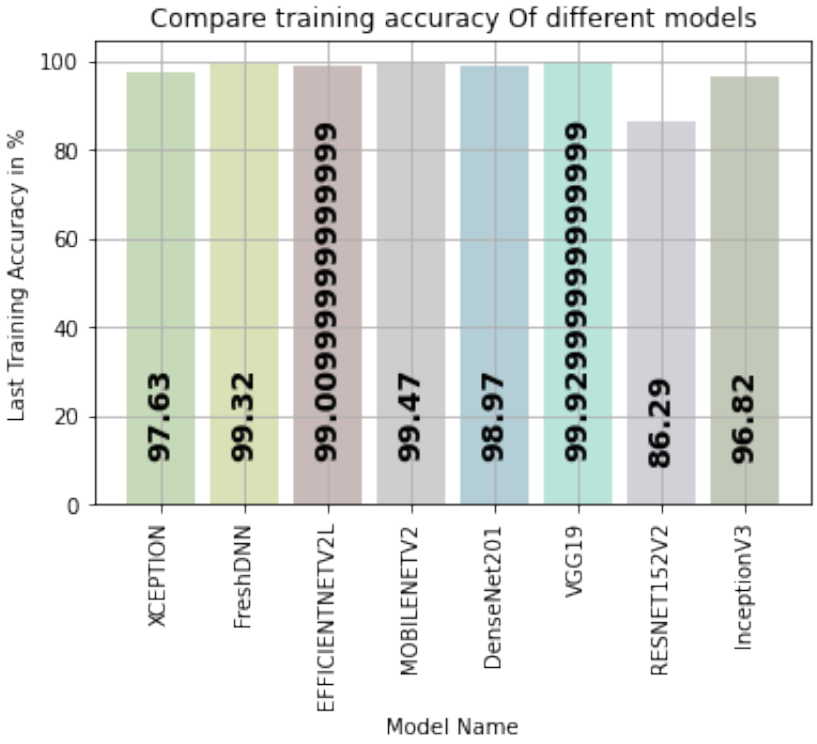


Figure 4.31: Comparison of Training Accuracy of Different Models

The bar graph compares the training accuracy of different CNN-based models applied in the research for fruit and vegetable detection. Among all models, the training accuracy of VGG19 is the highest (99.93%) whereas ResNet152V2 is the lowest(86.29%). Besides, in terms of training accuracy, MobileNetV2 is the second highest(99.01%), and our custom model FreshDNN is the 3rd highest at 99.32%. On top of that, the rest of the models performed well according to the training accuracy and it is 99.01%, 98.97%, 97.63%, 96.82% for EfficientNetV2L, DenseNet201, Xception, InceptionV3.

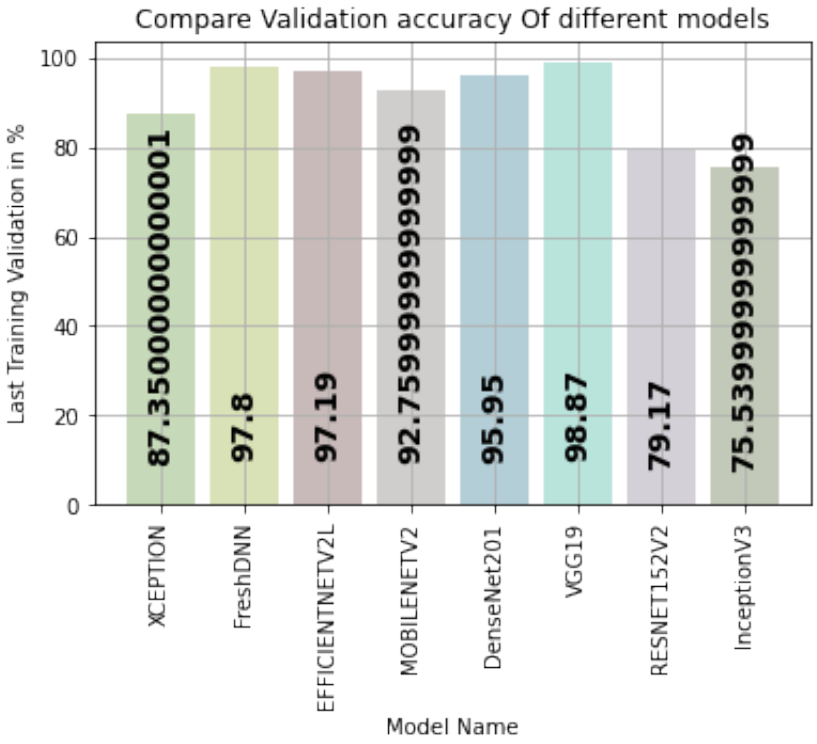


Figure 4.32: Comparison of Validation Accuracy of Different Models

The Bar graph shows the validation accuracy of different CNN-based models. It is visible in the graph that VGG19 has the highest validation accuracy and it is 98.87%, on the other hand, InceptionV3 has the lowest accuracy(75.54%). The custom model, FreshDNN performs 2nd in terms of validation accuracy (97.8%). The difference between VGG19 and FreshDNN is 1.07. Other models, for example, EfficientNetV2L, DenseNet201, MobileNetV2, Xception, ResNet152V2 performed 97.19%, 95.95%, 92.76%, 87.35%, 79.17% consecutively.

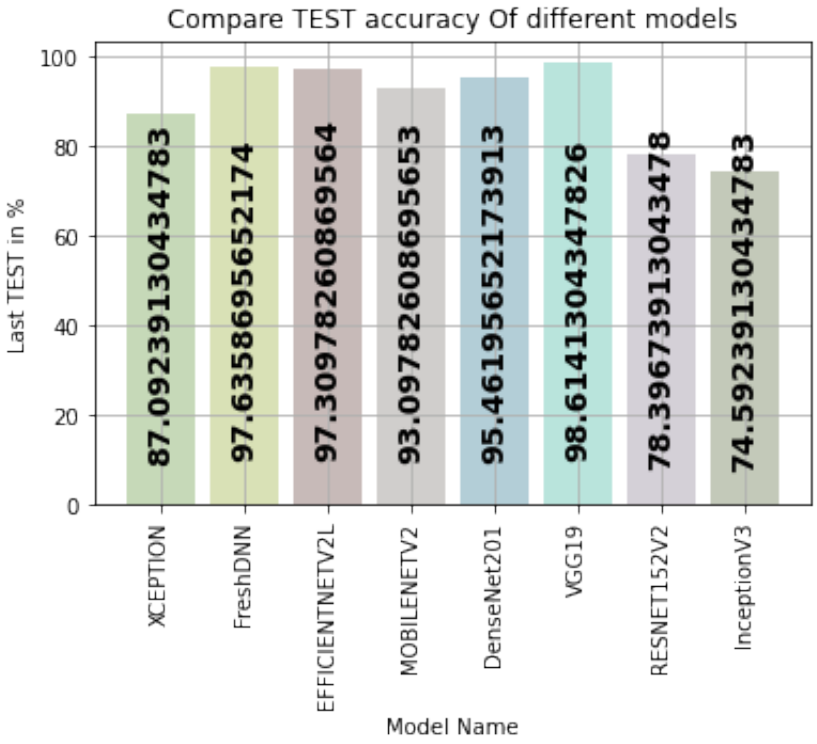


Figure 4.33: Comparison of Test Accuracy of Different Models

The Bar graph shows the test accuracy of different CNN-based models. It is visible in the graph that VGG19 has the highest test accuracy and it is 98.61%, on the other hand, InceptionV3 has the lowest accuracy(75.59%). The custom model, FreshDNN performs 2nd in terms of test accuracy (97.63%). The difference between VGG19 and FreshDNN is 0.98. Other models, for example, EfficientNetV2L, DenseNet201, MobileNetV2, Xception, ResNet152V2 performed 97.30%, 95.46%, 93.09%, 87.09%, 78.39% consecutively.

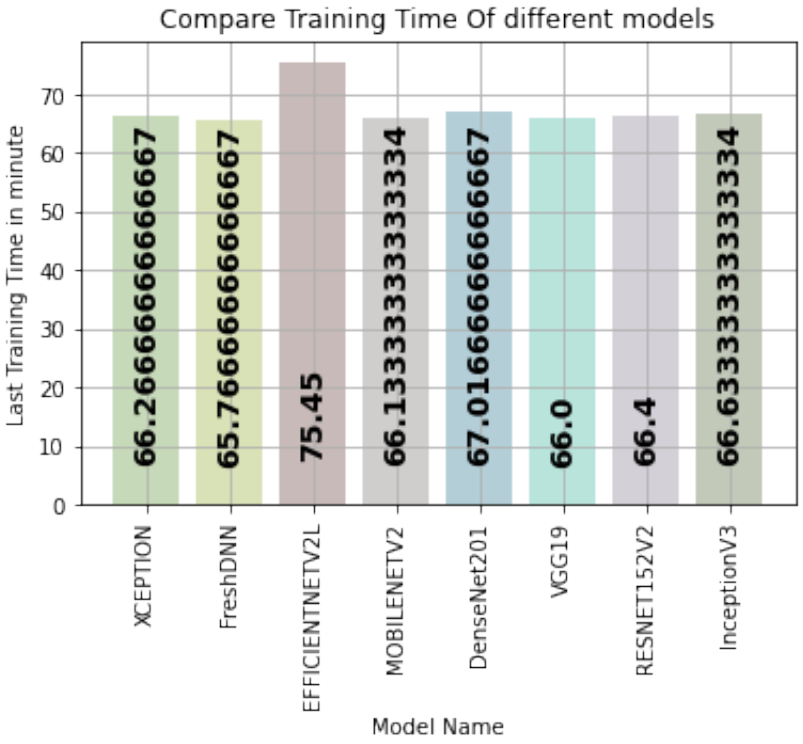


Figure 4.34: Comparison of Training Time of Different Models

The Bar graph shows the total training time of the different training models. The custom model FreshDNN has taken the lowest time 65.77 minutes whereas the model EfficientNetV2L has taken the highest time(75.45 minutes) to train. Apart from this rest of the models have taken almost a similar time to train for example it takes 66.0, 66.12, 66.27, 66.4, 66.63, 67.01 minutes to train VGG19, MobileNetV2, Xception, ResNet152V2, InceptionV3, DenseNet201.

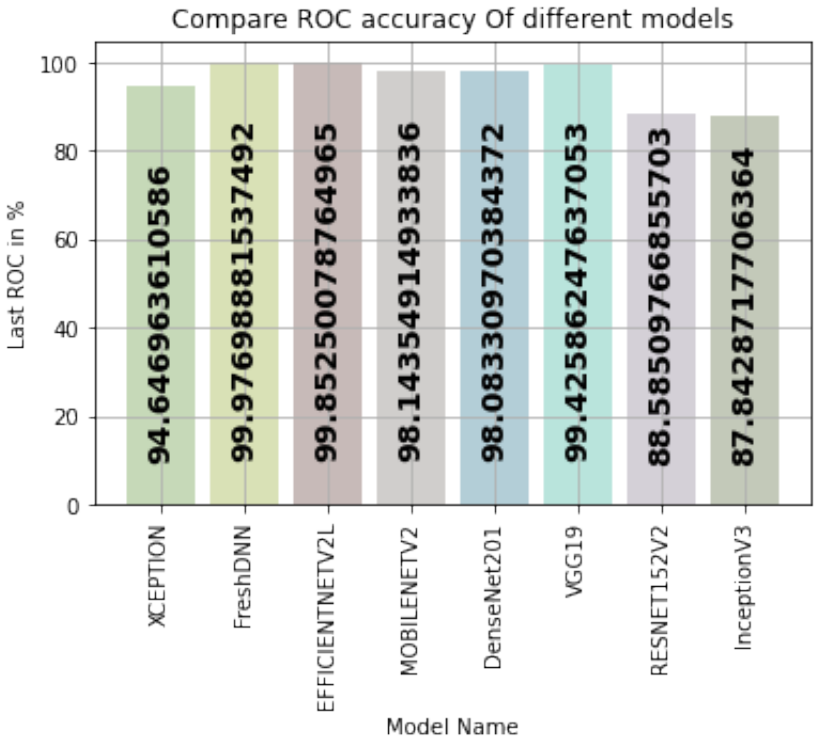


Figure 4.35: Comparison of ROC Accuracy of Different Models

The bar graph portrays the ROC accuracy of different CNN-based models. The ROC score of FreshDNN is the highest at 99.98 and it is better than any pre-trained models applied in the research. Among all pre-trained models, the EfficientNetV2L model has a better ROC score, it is 99.85 whereas Inception V3 has the lowest ROC score of 87.84, and 2nd lowest ROC score is 88.59 for ResNet152V2. Apart from these other pre-trained models have satisfactory ROC scores.

The overall performance of different CNN-based models is demonstrated below.

	Model Name	Last Training Accuracy(%)	Last Validation Accuracy(%)	roc_auc_score(%)	Testing Accuracy(%)	Total Training Time(min)
0	XCEPTION	97.63	87.35	94.646964	87.092391	66.266667
1	FreshDNN	99.32	97.80	99.976989	97.635870	65.766667
2	EFFICIENTNETV2L	99.01	97.19	99.852501	97.309783	75.450000
3	MOBILENETV2	99.47	92.76	98.143549	93.097826	66.133333
4	DenseNet201	98.97	95.95	98.083310	95.461957	67.016667
5	VGG19	99.93	98.87	99.425862	98.614130	66.000000
6	RESNET152V2	86.29	79.17	88.585098	78.396739	66.400000
7	InceptionV3	96.82	75.54	87.842872	74.592391	66.633333

Figure 4.36: Overall Performance of All Models

4.2 5-fold Cross Validation Result Analysis

k-fold is a cross-validation approach used to evaluate Machine Learning (ML) models using a small sample of data. This procedure contains a parameter called k that specifies the number of groups into which a given data sample should be split. Consequently, the term K fold cross-validation was coined.

For the research, in this procedure, firstly we shuffled the whole dataset of 16 classes (2300*16=36,800) randomly. Then we separated the 10% of data(230*16=3680) for the test set. Then, the remaining 90% of data(2070*16=33120) are split into 5 different groups to apply 5 k-fold. Out of 5 folds, 1 fold (414*16=6624 data) is considered as validation data and the rest 4 folds (414*16*4=26496) are considered as training data.
Then, on the basis of training data, the model was fit and validated using validation data and evaluated using test data.

Model fitting, validation, and evaluation were also done 4 more times where each time they considered each different fold as validation data set. Lastly, k-fold calculated the evaluated results on average.

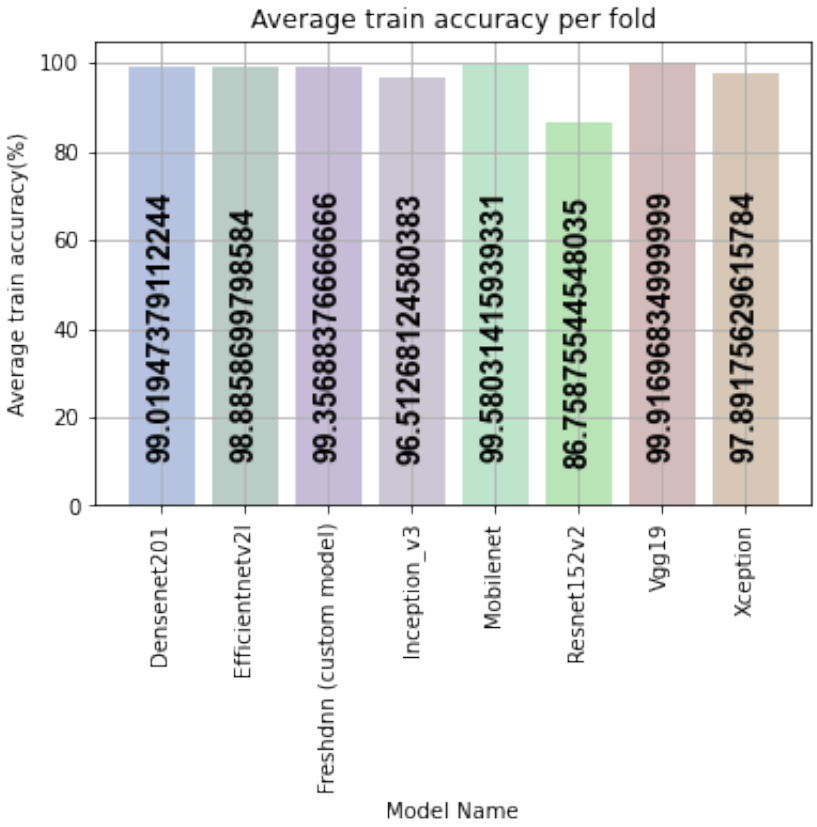


Figure 4.37: Comparison of 5 Fold Training Accuracy of Different Models Per Fold

The bar chart shows the average train accuracy per fold of the 7 pre-trained CNN models and 1 custom CNN model, "FreshDNN" to detect fruit and vegetable freshness. It is visible in the bar that VGG19 has the highest train accuracy(99.92%) and ResNet152V2 has the lowest train accuracy (86.75%). Besides, MobileNetV2 has the 2nd highest train accuracy(99.58%) and our custom model has the 3rd highest train accuracy and it is 99.36%. Apart from these other pre-trained models performed decently in terms of training accuracy.

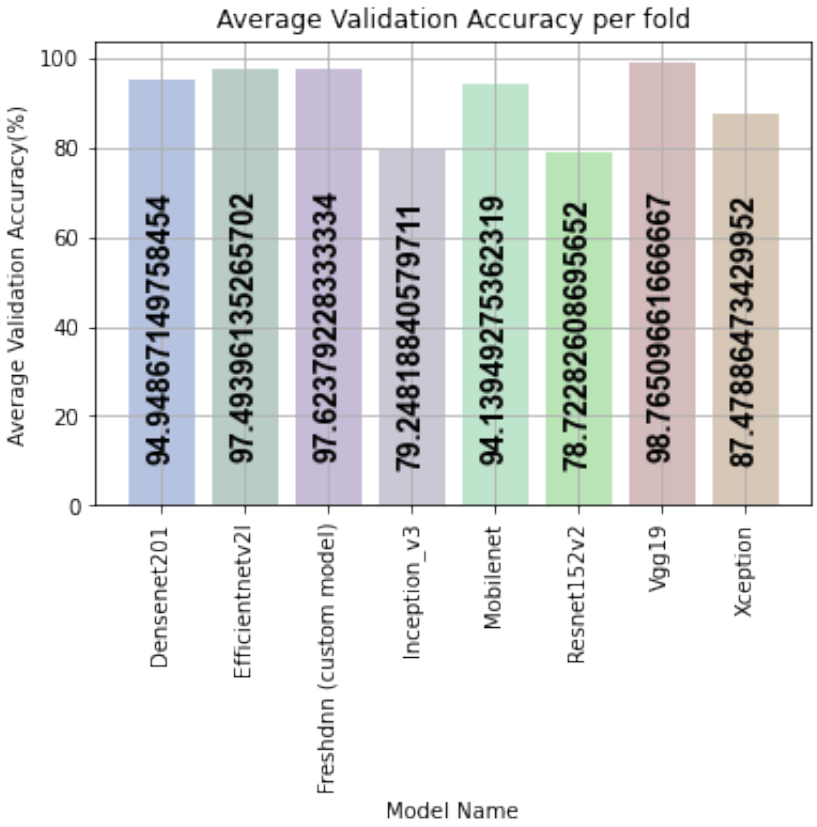


Figure 4.38: Comparison of 5 Fold Validation Accuracy of Different Models Per Fold

The figure demonstrates average validation accuracy per fold of different models. From test results we find that VGG19 was most accurate during the test. Its accuracy is about 98.76%. With an accuracy of 97.62%, FreshDNN holds the 2nd position followed by EfficientNetV2L having 97.49% accuracy. Other two models, DenseNet201 and MobileNetV2 showed accuracy more than 90%. Only one model showed accuracy in between 80-90% which is Xception having 87.48% accuracy. InceptionV3 and ResNet152V2 performed with below 80% accuracy. To summarize, 5 models out of 8 models showed precision more than 90 % and only 1 model showed below 90% but above 80% and rest 2 models performed below 80%. Among these models Vgg19 has most accuracy and Restnet152v2 showed least accuracy.

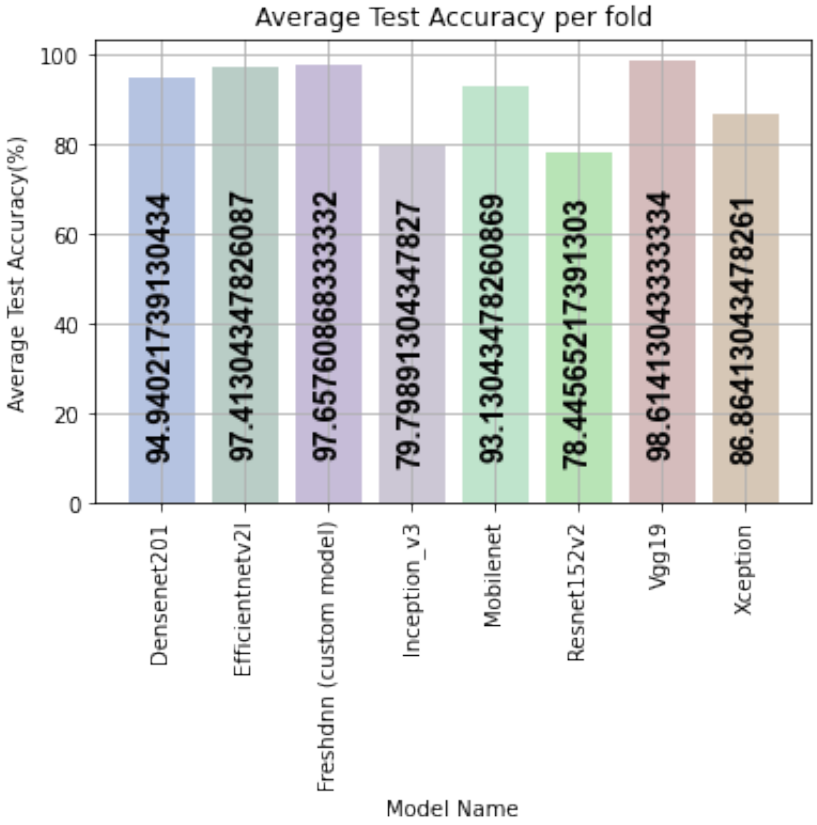


Figure 4.39: Comparison of 5 Fold Test Accuracy of Different Models Per Fold

The graph illustrates the average test accuracy per fold among different CNN-pertained models including the custom model. In the graph, it is noticed that VGG19 has the highest validation test accuracy which is above 98 %. Besides, both the FreshDNN and EfficientNetV2L have validation test accuracy above 97 %. Also, the DensNet201 and the MobileNetV2 have validation test accuracy above 92 %. The validation test accuracy of Xception is 86.8 % which is higher than Inception V3 and ResNet152V2, which have validation test accuracy of 79.7 and 78.44(lowest) respectively.

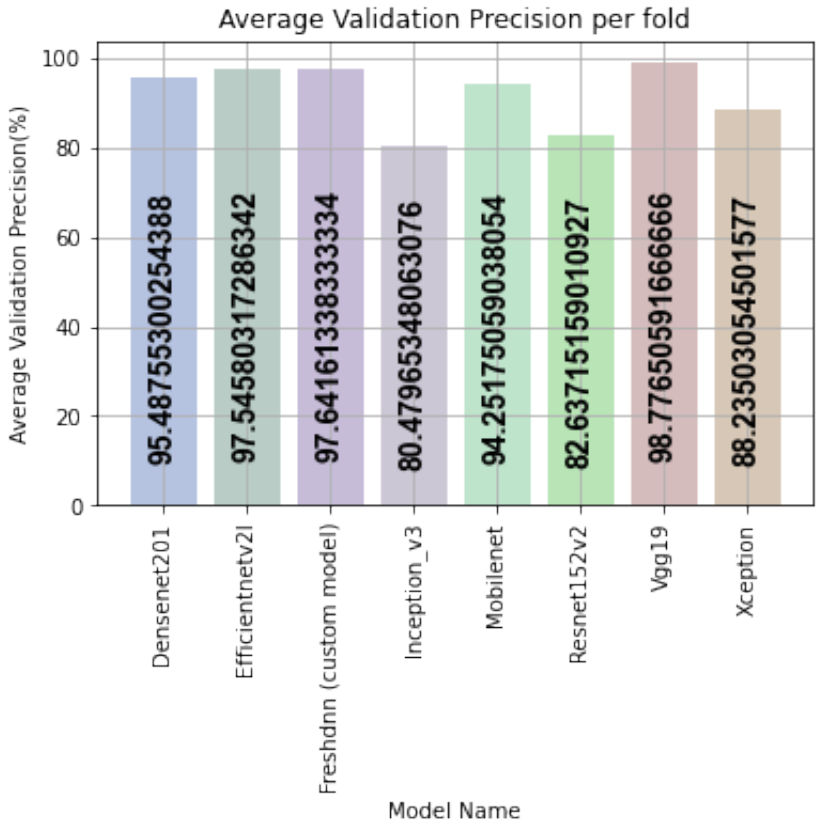


Figure 4.40: Comparison of 5 Fold Validation Precision of Different Models Per Fold

The bar chart above defines the average validation precision per fold of several CNN models for detecting fruit and vegetable freshness. Among 8 models, VGG19 (pre-trained model) has the highest precision (98.78%), whereas, the custom model obtained the 2nd highest position in terms of average validation precision and it is 97.641%. Besides, EfficientNetV2L scored 3rd highest position(97.546%). Moreover, the score of DenseNet201, MobileNetV2, and Xception performed decently and they scored 95.49%, 94.25%, and 88.24%. On the other hand, ResNet152V2 model % obtained 82.64% which is 2nd lowest and inception v3 has the lowest average validation precision (80.48%).

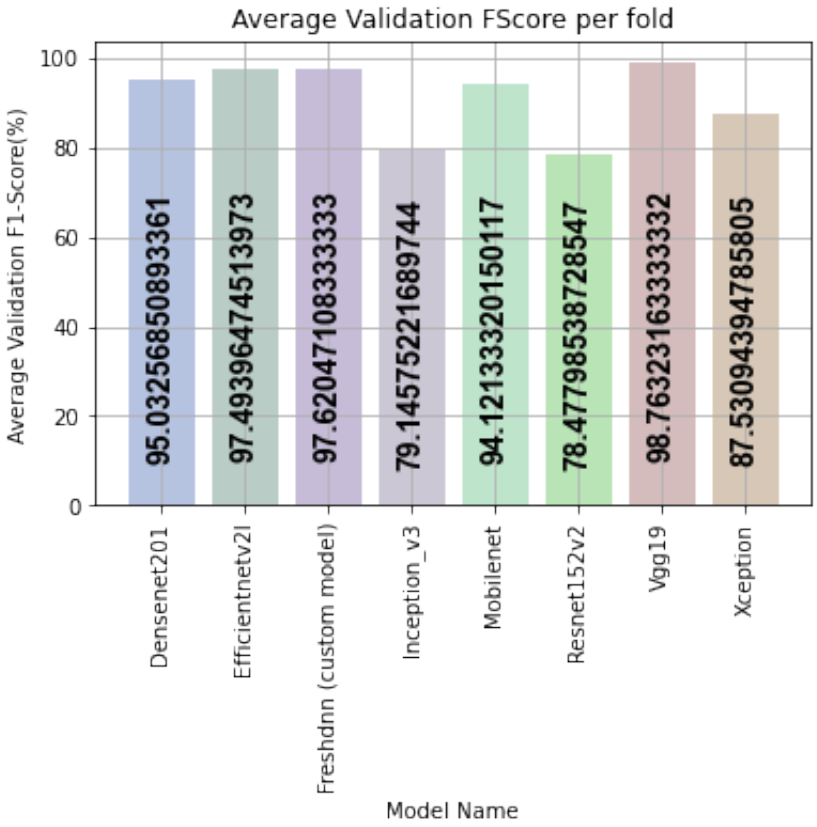


Figure 4.41: Comparison of 5 Fold Validation F1 Score of Different Models Per Fold

The graph demonstrates the average validation F1 score among various CNN pertain models including FreshDNN. VGG19 has the highest F1 score which is near 99%. Besides, both the EfficientNetV2L and the Custom model have F1 scores above 97% and the validation F1 score of DenseNet is 95%. Moreover, the validation F1 score of Xception is below 90%. Apart from these, InceptionV3 has a validation F1 score of 79% and ResNet152V2 has the lowest F1 score which is 78.47%.

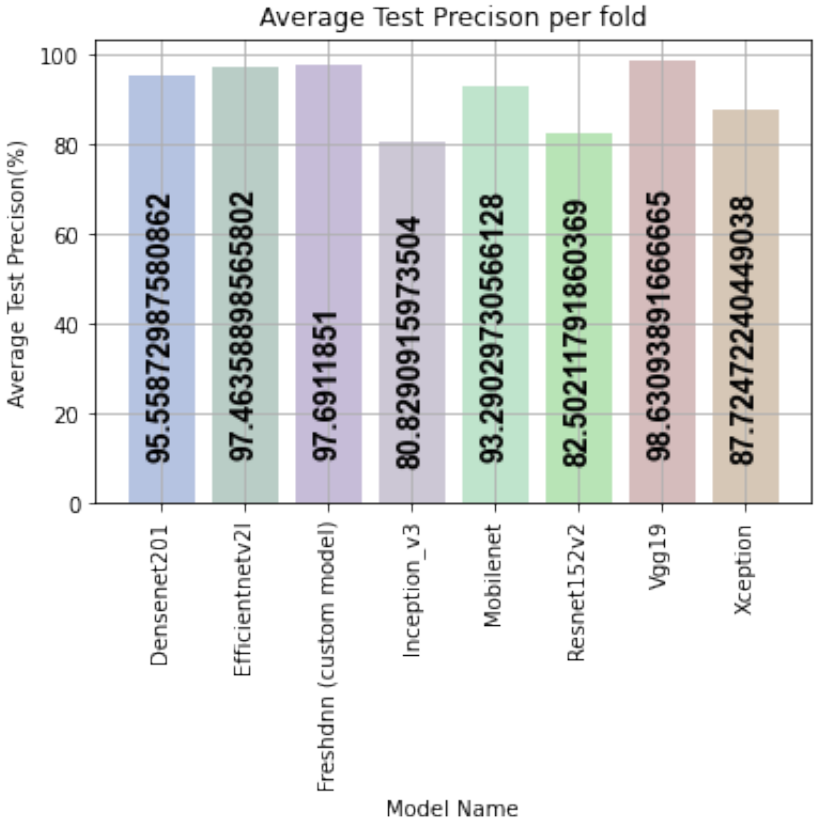


Figure 4.42: Comparison of 5 Fold Test Precision of Different Models Per Fold

The figure illustrates Average Test Precision per fold of different models. The test results show that VGG19 was most accurate during the test. Its accuracy is 98.63%. Having a reduction of about 1% in results, FreshDNN stands on the 2nd place with an accuracy of 97.69%. Its close competitor was EfficientNetV2L with 97.46% precision. After these three models only DenseNet201 and MobileNetV2 (accuracy 95.56% and 93.29% respectively) showed accuracy above 90%.
Xception, ResNet152V2 and InceptionV3 performed with accuracy more than 80%. Their accuracy was 87.72%, 82.50% and 80.83% respectively. To sum up 5 models out of 8 models showed precision more than 90% and rest of 3 models showed below 90% but above 80%. Among these models VGG19 was most accurate and InceptionV3 performed with least accuracy.

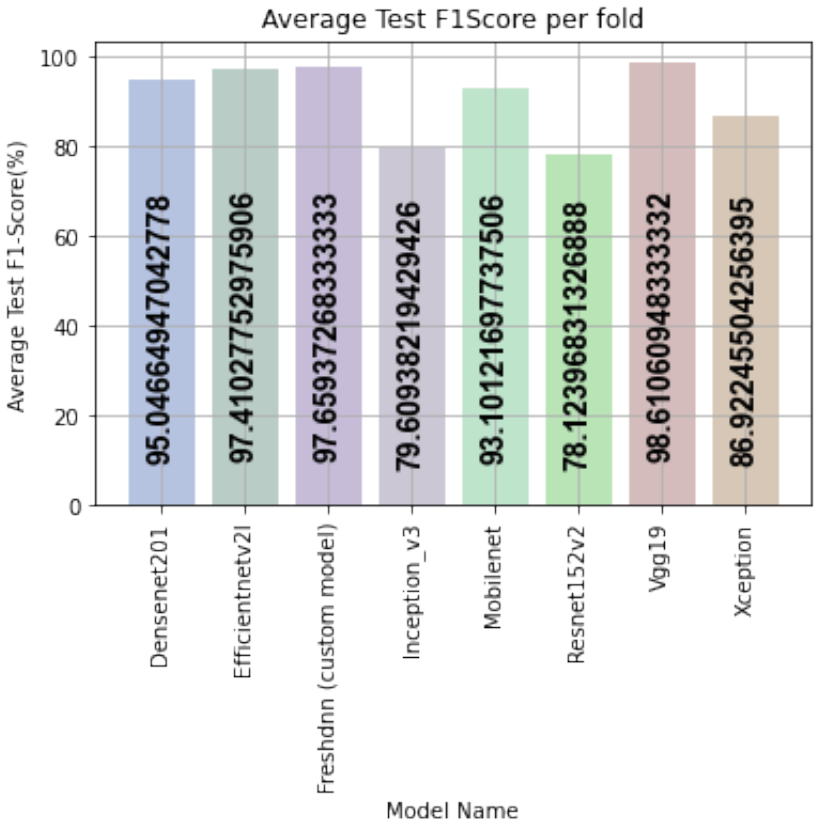


Figure 4.43: Comparison of 5 Fold Test F1Score of Different Models Per Fold

The bar chart above explains the average Test F1 score per fold of several CNN models (7 pre-trained and 1 custom). Among all models, VGG19 performed best (98.61%). Our custom model(FreshDNN) performed just below VGG19 and obtained the 2nd best position (97.66%). Besides, EfficientNetV2L (97.41%) also performed reasonably well. Moreover, DenseNet 201, MobileNetV2, and Xception performed satisfactorily and it is 95.04%, 93.1%, and 86.9% but both InceptionV3 (79.61%) and ResNet152V2(78.124%) performed less satisfactorily.

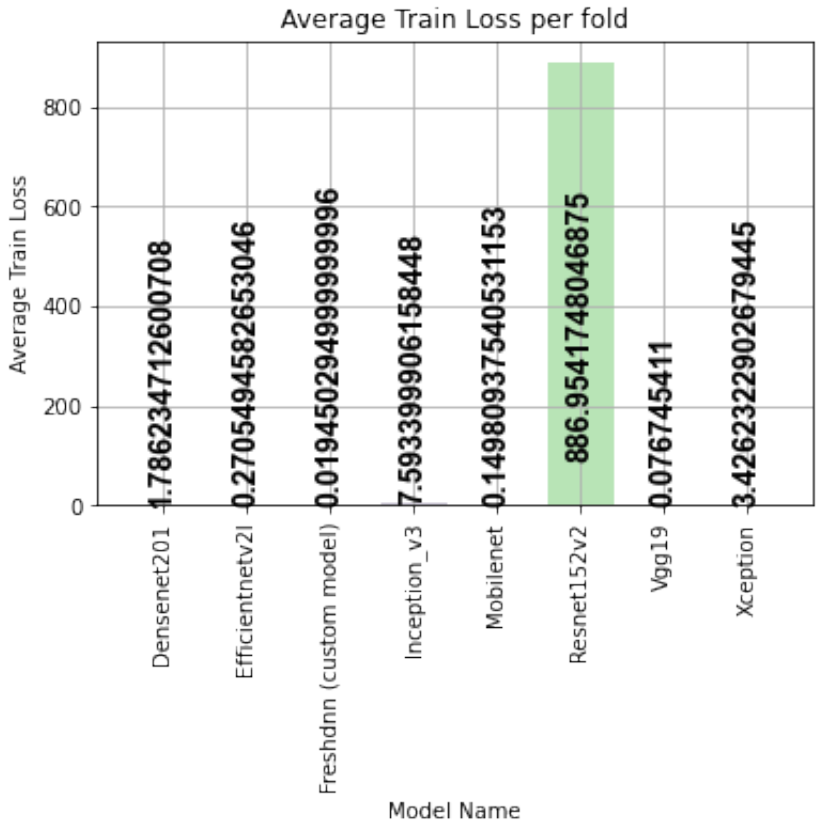


Figure 4.44: Comparison of 5 Fold Train Loss of Different Models Per Fold

The figure demonstrates average train Loss per fold of different models. The results showed that Custom model FreshDNN has the lowest train loss which is about 0.019. Low train loss indicates better accuracy. Hence FreshDNN has the greatest accuracy. After that VGG19 has a train loss of 0.076 which in turn makes it 2nd most accurate model. Only these two models have train loss less than 0.01. There are two models having train loss in range of 0.1 to 1. The models are MobileNetV2 and EfficientNetV2L which train loss 0.15 and 0.27 respectively. Rest of the models have train loss more than 1. DenseNet201, Exception, InceptionV3 and RestNet152V2 have train loss 1.786, 3.426, 7.593 and 886.954 respectively. These models are of lower accuracy then other models mentioned earlier. Among these models, RestNet152V2 has the most train loss which indicating it to be the most inaccurate model. To conclude it can be said, 2 models out of 8 models have train loss less than 0.01 in which Custom model was the most accurate during test. RestNet152V2 showed highest train loss which makes it less accurate model.

Chapter 5

Conclusion and Future Work

The main purpose of this research is to propose a novel CNN based model FreshDNN to detect if a fruit or vegetable is fresh or stale. To conduct this research, we have collected necessary data from various sources and prepared, preprocessed, and fed it into CNN models. We compared our custom model to pre-trained CNN models and assessed overall performance in order to improve it further. So far, seven pre-trained models have been applied to our dataset. Our focus of the research was to build a CNN model that would be faster in computation, smaller in size and perform as good as the pre-trained CNN models in terms of various performance metrics. This model can be deployed in a microcontroller-based embedded hand held device (using tensorflow micro) and in this task, our model is more feasible to apply than the aforementioned heavy pre-trained models like VGG19, MobileNet, ResNet152V2, and so on. Besides, for cross validation purpose, we used 5 fold technique and observed our proposed model performed reasonably well. In a nutshell, our custom model performed well overall for the collected dataset. In the criteria of parameter and size, it performed reasonably well. We hope, the agricultural industry, farming sector, and general public can benefit from this model in the automation of fruit and vegetable sorting and quality assessment improvement. In the future, we are eager to improve the model further by using Explainable AI (XAI) to explain our model more precisely and identify the behavioral pattern and exploring the possibility to merge with Vision Transformer (ViT) technology.

Bibliography

[1] D. Karakaya, O. Ulucan, and M. Turkan, "A comparative analysis on fruit freshness classification," in *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*, IEEE, 2019, pp. 1–4.

[2] L. Zhou, C. Zhang, F. Liu, Z. Qiu, and Y. He, "Application of deep learning in food: A review," *Comprehensive reviews in food science and food safety*, vol. 18, no. 6, pp. 1793–1811, 2019.

[3] M. A. Hedjazi, I. Kourbane, and Y. Genc, "On identifying leaves: A comparison of cnn with classical ml methods," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2017, pp. 1–4.

[4] S. Chan, V. Reddy, B. Myers, Q. Thibodeaux, N. Brownstone, and W. Liao, "Machine learning in dermatology: Current applications, opportunities, and limitations," *Dermatology and therapy*, vol. 10, no. 3, pp. 365–386, 2020.

[5] F. Valentino, T. W. Cenggoro, and B. Pardamean, "A design of deep learning experimentation for fruit freshness detection," in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing, vol. 794, 2021, p. 012 110.

[6] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," *sensors*, vol. 16, no. 8, p. 1222, 2016.

[7] W. Song, N. Jiang, H. Wang, and G. Guo, "Evaluation of machine learning methods for organic apple authentication based on diffraction grating and image processing," *Journal of Food Composition and Analysis*, vol. 88, p. 103 437, 2020.

[8] M. Kamruzzaman, Y. Makino, and S. Oshita, "Rapid and non-destructive detection of chicken adulteration in minced beef using visible near-infrared hyperspectral imaging and machine learning," *Journal of Food Engineering*, vol. 170, pp. 8–15, 2016.

[9] F. Femling, A. Olsson, and F. Alonso-Fernandez, "Fruit and vegetable identification using machine learning for retail applications," in *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, IEEE, 2018, pp. 9–15.

[10] H. Kagaya, K. Aizawa, and M. Ogawa, "Food detection and recognition using convolutional neural network," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1085–1088.

[11] P. Pouladzadeh and S. Shirmohammadi, "Mobile multi-food recognition using deep learning," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 3s, pp. 1–21, 2017.

[12] A. Taheri-Garavand, A. Nasiri, A. Banan, and Y.-D. Zhang, "Smart deep learning-based approach for non-destructive freshness diagnosis of common carp fish," *Journal of Food Engineering*, vol. 278, p. 109 930, 2020.

[13] H. Parastar, G. van Kollenburg, Y. Weesepoel, A. van den Doel, L. Buydens, and J. Jansen, "Integration of handheld nir and machine learning to "measure & monitor" chicken meat authenticity," *Food Control*, vol. 112, p. 107 149, 2020.

[14] A. Ren, A. Zahid, A. Zoha, *et al.*, "Machine learning driven approach towards the quality assessment of fresh fruits using non-invasive sensing," *IEEE Sensors Journal*, vol. 20, no. 4, pp. 2075–2083, 2019.

[15] N. U. Hasan, N. Ejaz, W. Ejaz, and H. S. Kim, "Meat and fish freshness inspection system based on odor sensing," *Sensors*, vol. 12, no. 11, pp. 15 542–15 557, 2012.

[16] N. Ismail and O. A. Malik, "Real-time visual inspection system for grading fruits using computer vision and deep learning techniques," *Information Processing in Agriculture*, vol. 9, no. 1, pp. 24–37, 2022.

[17] S. Phiphiphatphaisit and O. Surinta, "Food image classification with improved mobilenet architecture and data augmentation," in *Proceedings of the 2020 The 3rd International Conference on Information Science and System*, 2020, pp. 51–56.

[18] W. Wang, Y. Li, T. Zou, X. Wang, J. You, and Y. Luo, "A novel image classification approach via dense-mobilenet models," *Mobile Information Systems*, vol. 2020, 2020.

[19] A. G. Howard, M. Zhu, B. Chen, *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[20] L. G. Fahad, S. F. Tahir, U. Rasheed, H. Saqib, M. Hassan, and H. Alquhayz, "Fruits and vegetables freshness categorization using deep learning," *CMC-COMPUTERS MATERIALS & CONTINUA*, vol. 71, no. 3, pp. 5083–5098, 2022.

[21] R. R. Potdar, *Fresh and stale images of fruits and vegetables*, May 2021. [Online]. Available: https://www.kaggle.com/datasets/raghavrpotdar/fresh-and-stale-images-of-fruits-and-vegetables.

[22] A. Baloch, *Vegetables amp; fruits fresh and stale*, Apr. 2022. [Online]. Available: https://www.kaggle.com/datasets/alibaloch/vegetables-fruits-fresh-and-stale.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] L. Alzubaidi *et al.*, *Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. j. big data 8 (1), 1–74 (2021).*

[25] Y. LeCun, Y. Bengio, G. Hinton, *et al.*, "Deep learning. nature, 521 (7553), 436-444," *Google Scholar Google Scholar Cross Ref Cross Ref*, 2015.

[26] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in *2017 2nd international conference on image, vision and computing (ICIVC)*, IEEE, 2017, pp. 783–787.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[29] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2752–2761.