Declaration

I certify that this research work titled *"Food Quality Assessment Based on Deep Learning Models"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources has been properly acknowledged/referred to.

Signature of Student
MARYUM SANDHU
00000278117

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for plagiarism. Turnitin report endorsed by the supervisor is attached.

Signature of Student

MARYUM SANDHU

Registration Number 278117

Signature of Supervisor

Copyright Statement

Acknowledgements

I am extremely thankful to Allah (SWT) for guiding me throughout this work and blessing me with innovative thoughts and ideas. Without Your will and guidance, I would not be able to achieve this and whosoever guided me during my work was Your will. Truly all praise belongs to You.

I am profoundly thankful to my parents who raised me and taught me my first lessons of life and made me who I am today. It was their unconditional support that gave me the courage and helped me achieve every goal of my academic career. I am especially thankful to my mother for helping me in taking care of my daughter while doing my work. Without their resolute support, I would not have been able to do my research.

I would like to express my regards and thanks to my supervisor Dr. M. Jawad Khan for his able guidance. I am also thankful to him for Rehabilitation and Assistive Robotics, Computer Vision, and Deep Learning courses that he has taught me. His way of teaching has helped me grasp the concepts easily and his constant guidance helped me complete my thesis.

I would also like to thank Dr. Hasan Sajid, Dr. Usman Bhutta, and Dr. Karamdad for being on my thesis GEC committee.

Finally, I would like to extend my respect to everyone who has been of any help to my research.

Dedicated to my beloved parents, encouraging husband and affectionate siblings whose unconditional support and encouragement led me to this exceptional accomplishment.

Abstract

Objective. In this paper, a novel dataset has been collected in accordance with Pakistani needs and is used to develop an architecture for the quality assessment of fruits and vegetables. *Approach*. The dataset contains images captured under uncontrolled conditions with respect to illumination, temperature, humidity, image resolution, image aspect ratio, angle of capturing images and background. Images captured contain items individually as well as in groups. To the best of the knowledge gathered, this is the first of its kind dataset. This dataset is then preprocessed. Among usual preprocessing techniques, an aspect ratio adjustment algorithm has been introduced. After preprocessing, the data is used to train multiple models (AlexNet, VGG-16, ResNet-50, Fruits-360 Model and a proposed model with relatively lesser depth). This performs recognition of fruits and vegetables and endorse the validity of the dataset. Going further, the dataset is then prepared for quality assessment with three quality labels for each fruit/vegetable: Eatable, Partially Rotten and Rotten. Quality assessment is then performed using pre-trained VGG-16 through transfer learning, adding a fully connected network and fine-tuning the model. *Main Results.* The highest recognition accuracy on the validation set is 98.9% and the highest validation accuracy for quality assessment is 92.9%. *Significance.* Outcomes of this research demonstrate that dataset collected under an uncontrolled environment can be used for recognition of fruits/vegetables with remarkable accuracies. Moreover, quality assessment of fruits/vegetables is performed accurately with the same dataset using deep learning and three quality labels.

Key Words: *Quality Assessment of Fruits/Vegetables, Preprocessing for Aspect Ratio Adjustment, VGG-16, Transfer Learning*

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

The following research work is focused on quality assessment of fruits and vegetables using deep learning models. Among other important foods, fruits and vegetables are an essential part of the human diet on daily basis. When it comes to consuming fruits and vegetables it is highly important to assess their quality. From being picked up from farms and orchards, packaging, transportation, and displaying in grocery stores to reaching its consumers, there are lots of factors that deteriorate the quality of fruits and vegetables. And it is highly important to ensure a good quality check at every stage of this cycle. This calls for a method that is both less tedious, less error-prone and less time-consuming than the conventional and subjective methods that include experience of observer, past knowledge, cultural practices, etc. and thus is more error-prone as well and dependent on so many factors like observer's memory, some particular experience, and the decision may be biased as well. Moreover, all the chemical and microbiological methods of quality assessment of fruits and vegetables are destructive in nature and laborious. On the other hand, deep learning-based methods are eco-friendly and less laborious and offer a time-efficient process. Deep learning has been a core subject of machine learning and has helped in achieving many milestones of research in image classification problems. Deep learning offers state-of-the-art models to accomplish classification tasks with considerably good accuracy with the ease of automatically extracting features combined with a classifier.

Convolutional Neural Networks (CNNs) are an integral class of deep learning that has been recently used in agriculture and food production. The main purpose has been to achieve some sort of image classification. In this research deep learning models have been used and a new dataset has been introduced for performing the following tasks: Fruits and vegetables recognition followed by the quality assessment of the corresponding fruit/vegetable.

1.1 Food Quality Assessment and Deep Learning:

The literature survey shows that little work has been done in this field. Sorting of healthy and defective dates has been done using deep learning models (Nasiri et al, 2019). The research has been done on the Shahani date fruit found mostly in Iran and is very much liked there. A VGG16 model has been fine-tuned and the data set collected contains images that had equal aspect ratios and background of all images was kept constant. The model got an overall accuracy of 96.9%.

In another attempt, Fresh and rotten fruits have been classified using different deep learning models with pre-collected Kaggle dataset (Chakraborty et al., 2021). The images in the data set are taken in a highly conditioned environment. All images have constant white background, same illumination, same resolution, same aspect ratio and covering almost same angle of the fruit. There is little to no variation observed in the dataset as per the literature survey.

1.2 Transfer Learning:

Training models with a greater number of layers i.e., deeper networks offer more complexity and thus take more time to train. Many deeper networks such as VGG16, RESNET50, and AlexNet take longer when developed from scratch and trained. To overcome this, transfer learning can be used to perform the training without having to train the whole model. The concept of transfer learning belongs to machine learning area where a pretrained model is finetuned and used for a new task thus saving training time and achieving better accuracy in less time and epochs. The pretrained model is trained on a larger dataset and weights and biases are frozen and are used as starting point for the new task at hand.

Moreover, in the domains like bioinformatics and robotics collecting a large-scale well-annotated dataset is very difficult due to the expense of data acquisition and costly annotation. In such cases, transfer learning can be used to obtain good accuracy for insufficient data. (Tan, C. et al., 2018).

1.3 Previous Work:

Agriculture has always been an important sector for research. Among other food products, there has been a recent development in research work regarding the recognition of different fruits using deep learning models. A new dataset namely Fruits-360 was used for this purpose (Moresan and Oltean, 2018). The fruits were mounted on a shaft of a low-speed motor and a video was captured followed by frame extraction. CNN was used for recognition. A hybrid model can also prove to be an effective framework for fruit recognition (Xue et al., 2020).

Table 1.1: Literature Review

Reference	Title of Paper	Dataset	Model	Accuracy
Moresan and Oltean, 2018	Fruit recognition from images using deep learning	Fruits-360 dataset (82213 images of 120 fruits)	CNN	Train: 99.5 % Test: 95.2 %
Ren et al., 2019	Machine Learning Driven Approach Towards the Quality Assessment of Fresh Fruits Using Non-Invasive Sensing	Fresh Fruit slices of Apple and Mango for Moisture Content	SVM, KNN, Decision Tree	SVM: 90.9 % KNN: 82.9 % DT: 95.4 %
Pathmanaban et al., 2019	Recent Application of Imaging Techniques for Fruit Quality Assessment	-	Hyperspectral imaging, Raman imaging, MRI imaging and Laser backscattering imaging	-
Nasiri et al., 2019	Image Based Deep Learning Automated Sorting of Date Fruit	1300 images of healthy and defective dates	VGG16	Train: 97.9 % Validation: 98.4 %
Xue et al., 2020	A hybrid deep learning-based fruit classification using attention model and convolution autoencoder	Fruit 26 (Muresan et al.)	Attention-based DenseNet	95.8 %
Naranjo-Torres et al., 2020	A Review of Convolutional Neural Network Applied to Fruit Image Processing	Six categories from fruits-360 dataset	CNN	Train: 100 % Test: 95.45 %
Bhole and Kumar, 2020	Mango Quality Grading using Deep Learning Technique: Perspectives from Agriculture and Food Industry	RGB and Thermal Mango Dataset (1500 Images)	Fine-tuned SqueezeNet	Validation:93.3 %
Nayak et al., 2020	Intelligent food processing: Journey from artificial neural network to deep learning	-	ANN and CNN	Comparison Study
Maria and Darwin, 2021	Deep Learning Approach for Food Quality Inspection and Improvement on Hyper Spectral Fruit Images	Hyperspectral images	ANN, SVM	Proposed: 60 %
Chakraborty et al., 2021	Implementation of Deep Learning Methods to Identify Rotten Fruits	Fruits fresh and rotten for classification (13599 images) Kaggle.com	MobileNetV2 (53 layers)	Train: 94.5 % Test: 94.9 %
Zhang et al., 2021	Food and agro-product quality evaluation based on spectroscopy and deep learning: A review	-	-	-
Antony and Satheesh, 2021	A Comparative Study on Predicting Food Quality using Machine Learning Techniques	Collected dataset for 10 categories	SVM, ANN	Mango: 87 % Guava:90 % Meat: 81.5 % Milk: 98.7 %
Bhargava and Bansal, 2021	Fruits and vegetables quality evaluation using computer vision: A review	-	-	-
Bhargava et al., 2022	Machine Learning–Based Detection and Sorting of Multiple Vegetables and Fruits	healthy and defective vegetables and fruits (a total of 25,988 datasets of images)	Logistic Regression, SRC, ANN, SVM	LR: 85.5 % SRC: 87.6 SVM: 97.6 % ANN: 92.6 %

For the past few years, use of CNN for fruit recognition has greatly increased either by developing new models or using pre-trained models (transfer learning) (Naranjo-Torres et al., 2020). As a step forward, attempts are being made to achieve fruit quality grading using Hyperspectral imaging, Raman imaging, MRI imaging, laser backscattering imaging (Pathmanban, P. et al., 2019) and thermal imaging (Bhole and Kumar, 2020). Techniques used for food quality assessment range from K nearest neighbor (KNN), Decision Tree (Ren et al., 2019), Artificial Neural Network (ANN) (Antony and Satheesh, 2021), Support Vector Machines (Maria and Darwin, 2021) to CNNs.

1.4 Problem Statement:

Food quality assessment is one of the crucial steps to meet the consumers demand for high quality and safe food products. Now-a-days when online shopping is the preference of all buyers, shopping food items like fruits and vegetables online is still a doubtful task to perform. The food quality (freshness and rottenness) cannot be determined by simply looking at the provided pictures/videos. Moreover, even while making an offline purchase subjective methods are not suitable being time-consuming and more error-prone.

1.5 Approach Used:

In this research, a dataset has been collected for six categories of fruits and vegetables. The collected data set contains temperature and humidity recordings and is arranged as per the sequence of days starting from purchase of the item to its rotten stage. Next step is to arrange this data set in a folder hierarchy with proper categorical names of classes. This data preparation step is very much required for easy acquisition of images to perform training and validation. The next step is to perform the usual image pre-processing required to get optimum results. In addition to the usual pre-processing, an algorithm has been developed to perform aspect ratio adjustment to all images. After data preprocessing the next step is to develop models for recognition into six categories of the corresponding fruits and vegetables. Then the main task is to perform quality assessment on the recognized fruits/vegetables and model development for the task. The final result is the quality label of the corresponding item. Figure 1.1 shows the workflow of the research.

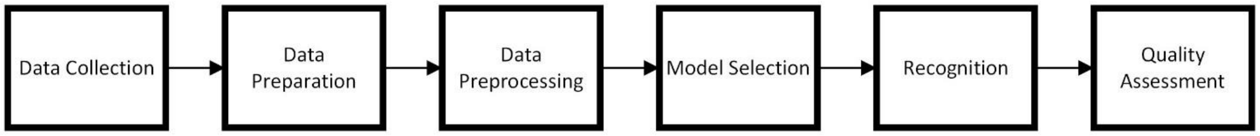


Figure 1.1: Workflow for this research

1.6 Objectives:

Following are the objectives of this research:

- Collection of data samples of fruits and vegetables in accordance with Pakistan
- Use of transfer learning on the collected data of fruits and vegetables
- Development of architecture for quality assessment of fruits and vegetables

1.7 Thesis Overview:

The research work in this thesis is defined as: Chapter 2 contains the theory of all the methods used in the approaches used for the Recognition and quality assessment of fruits and vegetables. It includes all the theoretical concepts for understanding the proposed scheme. Chapter 3 consists of the approach that has been used to achieve our objectives. It also includes the details of models used, data collection, features of the dataset collected, and the algorithms used to achieve the desired results. Chapter 4 contains the results acquired. Chapter 5 includes a discussion of the approaches used. Chapter 6 contains the conclusion of the thesis and Chapter 7 explains the future work briefly.

CHAPTER 2: THEORY

Some of the concepts necessary to understand this research are briefly explained in the following sections.

2.1 Deep Learning

Deep learning is a subfield of machine learning that uses structures and functions which are inspired by the human neural network. The word deep corresponds to the greater depth of deep neural networks. And greater depth corresponds to a greater number of layers in the network. When talking about neural networks deep learning algorithms would be deemed at the pedestal. Deep learning has greatly influenced the task that involves image processing.

2.1.1 Deep Learning and Image processing

Convolutional Neural Networks (CNNs) are the deep learning networks that have revolutionized the concept of image processing. CNNs have the advantage that the image shape is maintained while feeding it as input to the networks, unlike others. Also, one major advantage of CNNs is that they have automatic feature extraction that saves from the cumbersome process of extracting features including the decisions to extract what kind of features are important. These advantages have rendered these networks very useful and thus in the past few years, they have been widely used for a number of applications regarding image processing. Figure 2.1 shows a general architecture of a CNN.

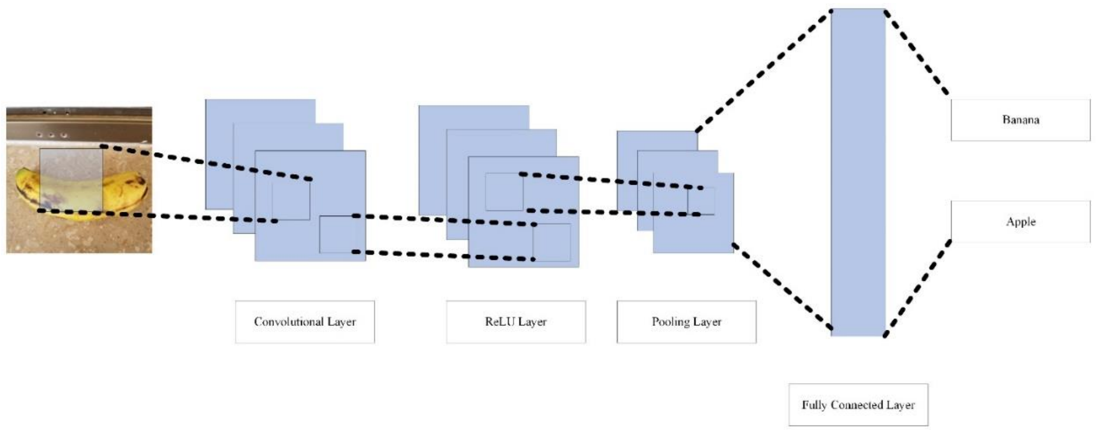


Figure 2.1: A general CNN architecture

The CNN consists of mainly the convolutional layers, it may or may not have pooling layers. In each convolutional layer there is an activation function that is applied to the result of convolution. In figure 2.1 this is labeled as the ReLU layer.

The input has 3 dimensions: height, width, and depth. Where the height is kept equal to the width and the depth refers to as the number of channels. Like in RGB images the depth is 3. So, the input would be e.g., 224x224x3 (figure 3.7).

Each bunch of feature maps (**Convolutional Layer**) is a hidden layer. These convolutional layers are the result of applying filters (kernels) to the input. After this, some activation function is applied to add some nonlinearity. Next is the **pooling** layer which corresponds to the operation of subsampling. Pooling can be max pooling, min pooling, or average pooling.

At the end there is a **fully connected layer**. This fully connected layer corresponds to the usual fully connected network that consists of dense layers, an activation function which in most cases is **softmax** and some loss function which is now-a-days mostly **cross entropy loss**. Size before and after the convolution can be denoted by:

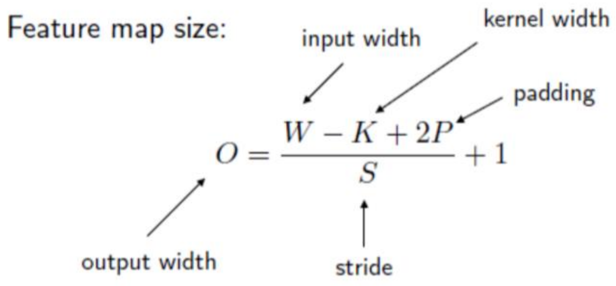


Figure 2.2: Feature Map Size

2.1.1.1 Hyperparameters:

Hyperparameters are parameters whose values are set before the learning process begins and whose values cannot be deduced from the training data. Some hyperparameters are learning rate, number of filters in each convolutional layer, size of filters, the batch size for training, number of convolutional layers, choice of activation function, pooling size, number of dense layers, regularization, dropout, etc.

2.1.1.2 Cost Function

The cost function is used to evaluate the performance of our model. It takes the predicted output and the actual output and calculates how far we are from the actual output. There are many choices for the cost functions, the one used in this research is **sparse categorical cross-entropy loss**.

2.1.1.3 Optimizer

While training the CNN, there is a need to update the weights at each epoch and minimize the cost function. Optimizer is a function that changes the attributes such as weights and learning rate so as to minimize the loss and maximize the accuracy.

2.1.1.4 Regularization

Regularization is a technique that is used to avoid the problem of overfitting while training a model. As a result of overfitting, the model fits very tightly to the training data with a high training accuracy but does not predict well on test data. There are several techniques used for regularization including L1 regularization, L2 regularization, dropout, data augmentation, early stopping, etc. Figure 2.3 shows how early stopping can be used to avoid overfitting.

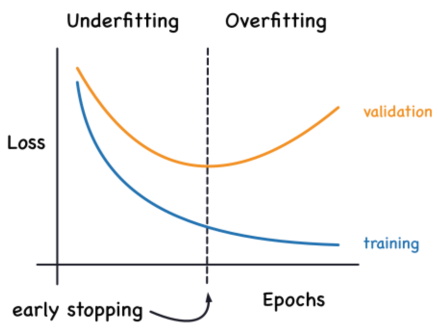


Figure 2.3: Early stopping for regularization

2.1.1.5 Batch Normalization

Batch normalization can be used as a regularization technique. It basically standardizes the inputs to a layer for each mini batch. This greatly impacts reducing the number of training epochs required.

CHAPTER 3: PROPOSED METHODOLOGY

3.1 Data Collection:

3.1.1 Experimental Setup:

The setup was established outdoors. All the items were placed on a surface in groups and individually. Keeping in mind the fact that fruits and vegetables are not always placed individually in real life. Images were taken from smartphone camera. Images were taken at 5 times of the day in the following order: 0900 hrs., 1200 hrs., 1500 hrs., 1800 hrs., and 2100 hrs. At each time of the day, 10 images were captured for each category of fruits and vegetables. 5 of which were taken individually and 5 were taken in groups of the same fruit/vegetable. Other images were also captured under categories of fruits combined, vegetables combined, common salient feature detection, and all of the fruits and vegetables combined. At daytime, natural illumination was used but at night, a little illumination was required to be able to avoid all dark images. The surface was changed with days to avoid a constant background for the images. The image resolution and aspect ratio settings were also changed to incorporate a diversity in images. There was no camera holding setup rather all images were taken from different angles. Figure 1.2 shows one of the settings made during data collection.

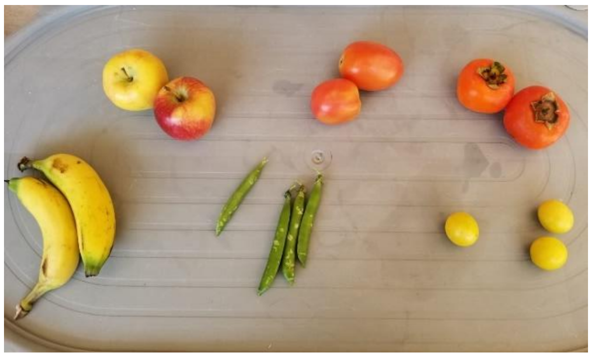


Figure 3.1: Experimental setup for data collection

3.1.2 Features of the Dataset:

Following are the main features of the data set:

- All pictures are RGB images taken from smartphone camera

- The dataset comprises of 5051 images
- Images taken are from different times of the day from morning to night (9:00 am, 12:00 pm, 3:00 pm, 6:00 pm and 9:00 pm)
- Images are captured at different image resolutions
- All images have different aspect ratios
- Images are captured from different angles thus capturing almost all sides of the fruit/vegetable
- Some images have one object (a fruit or a vegetable) while others have more than one objects belonging to the same class
- Temperature readings have been recorded
- Humidity percentage for the time of the day has been recorded
- Each time reading has a corresponding quality label for each fruit/vegetable separately
- Three quality labels have been set: Eatable (Fresh + Eatable), Partially Rotten and Rotten
- Data set also contains images for common salient feature detection for future usage

Figure 3.2 shows some sample images from the dataset.

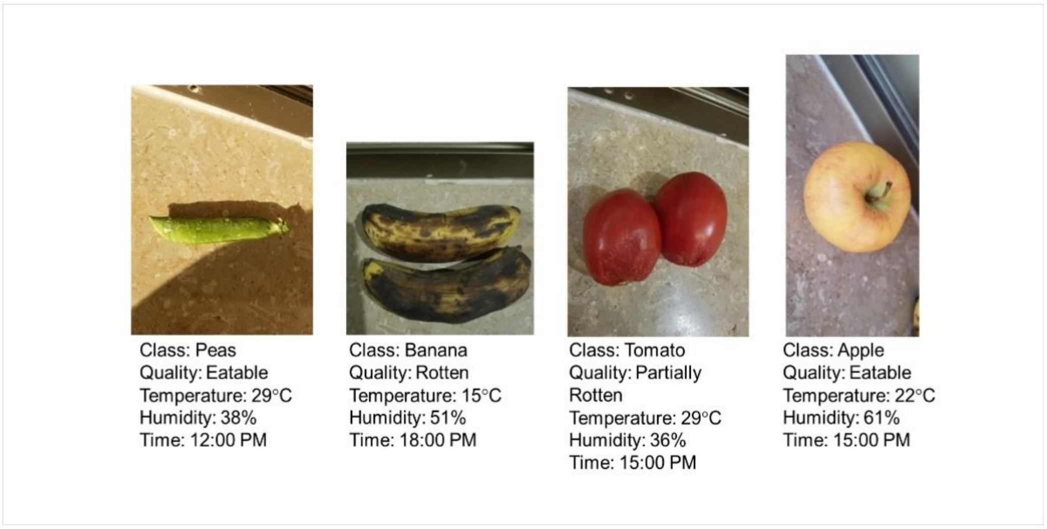


Figure 3.2: Sample images from dataset

Table 3.1 shows the number of images for each class of fruits and vegetables.

Table 3.1: Number of images for each class for recognition

Sr. No.	**Classes**	**No. of Images**
1	Apple	1110
2	Banana	630
3	Persimmon	160
4	Lemon	660
5	Peas	570
6	Tomato	670

To give an insight about how the fruits/vegetables looked in the three states of quality figure 3.3 shows some images from each quality label of different categories.

3.2 Data Preparation and Preprocessing:

All the data has been arranged in a folder hierarchy and all the folder names are named as the quality labels in the case of quality assessment and for the case of recognition, the folders are named as the names of the fruits and vegetables. After data preparation and loading the dataset for training some preprocessing needs to be done depending upon the nature of the data. Preprocessing is a very crucial step in any image classification problem.

The data was loaded using OpenCV-Python. Using the module of OpenCV the images imported are in the BGR (Blue Green Red) colorspace, so we need to change its colorspace to RGB (Red Green Blue) to avoid any confusion in the rest of our work. After solving this problem, the next one is aspect ratio adjustment. Looking at the features of the dataset as mentioned earlier in section 3.1.2 and observing the sample images from the data in figure 3.2 and figure 3.6 it can clearly be deduced that the images required some aspect ratio adjustment before resizing them to a suitable size for training a model. Figure 3.4 shows the different aspect ratios in the dataset.

3.2.1 Aspect Ratio Adjustment

A novel algorithm has been presented here to overcome the problem mentioned above in section 3.2. The algorithm assumes that the object is at the center of the images taken. Since images have different aspect ratios, so when they are resized to a smaller size as 224 x 224 (1:1 aspect ratio) the images become squashed and distorted. The algorithm is shown in figure 3.5.

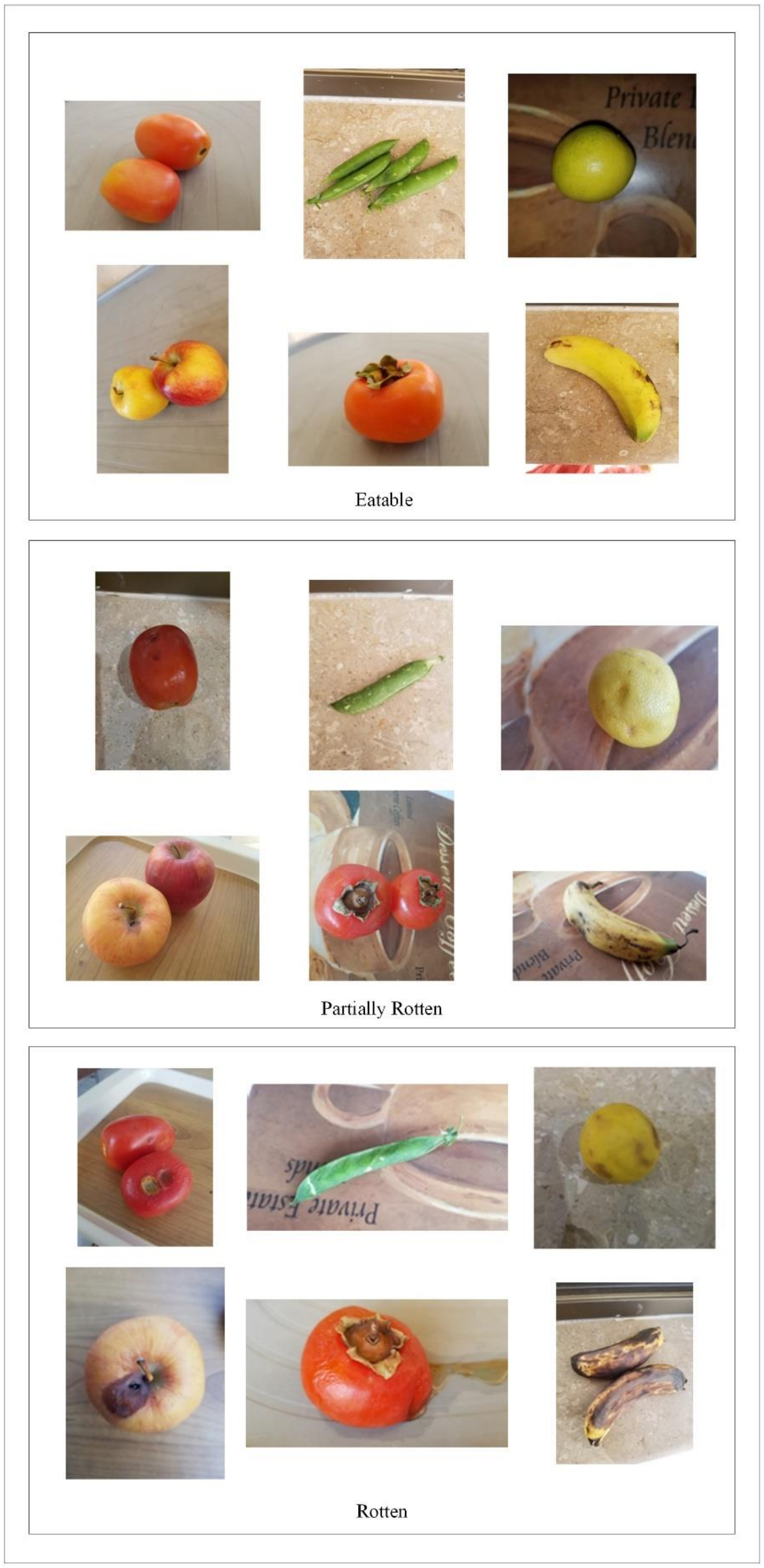


Figure 3.3: Quality stages of fruits/vegetables

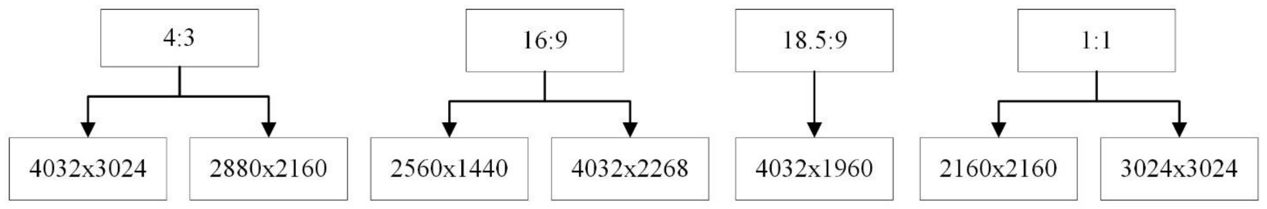
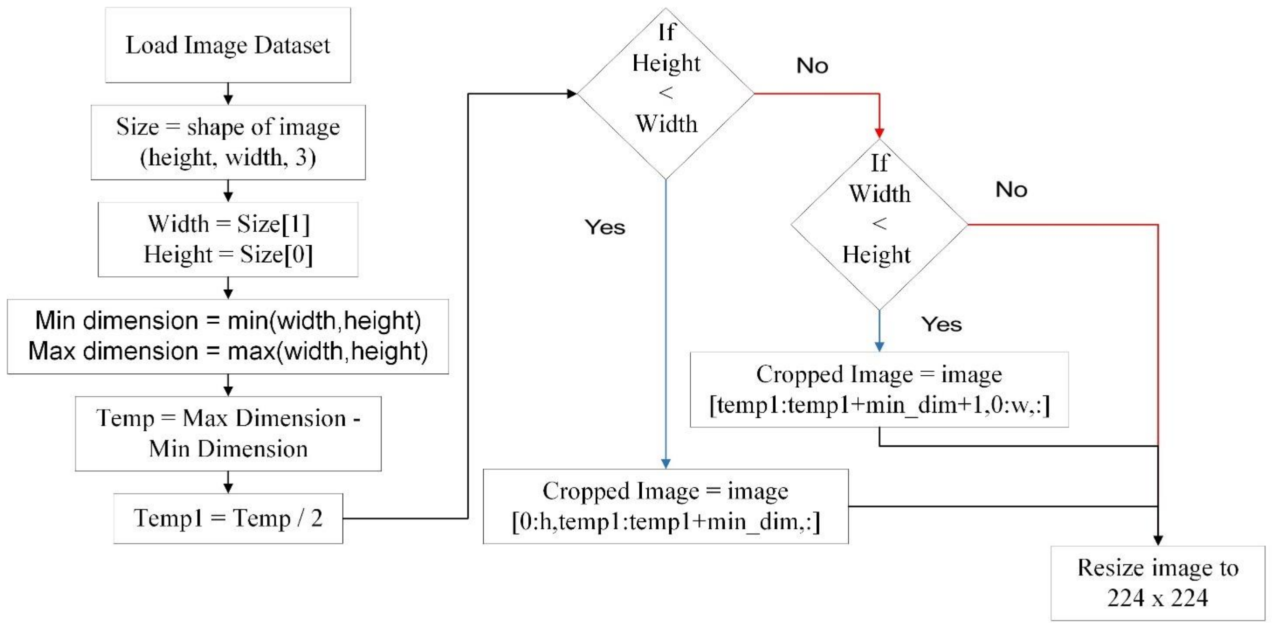


Figure 3.4: Aspect ratios of images in the dataset



Load Image Dataset
Size = shape of image (height, width, 3)
Width = Size[1]
Height = Size[0]
Min dimension = min(width,height)
Max dimension = max(width,height)
Temp = Max Dimension - Min Dimension
Temp1 = Temp / 2
If Height < Width
No
Yes
If Width < Height
No
Yes
Cropped Image = image [temp1:temp1+min_dim+1,0:w,:]
Cropped Image = image [0:h,temp1:temp1+min_dim,:]
Resize image to 224 x 224

Figure 3.5: Aspect ratio adjustment

The algorithm works by finding the size of the image. Then it follows to save the maximum and minimum dimension may it be height or width. The longer dimension is reduced to be equal to the smaller one by performing corresponding image slicing/cropping. This results in a good shift to 1:1 ratio for the images in this dataset since all the objects are at the center of the image. After transforming to 1:1 resizing to a small dimension e.g., 100 x 100, 224x224 etc. does not cause distortion in the image and thus makes it easier for the model to train. The advantage of using this algorithm can clearly be observed in figure 3.6. After this adjustment, all images are resized to 224 x 224 and rescaled by 1/255 so that all pixel values are between 0 and 1.

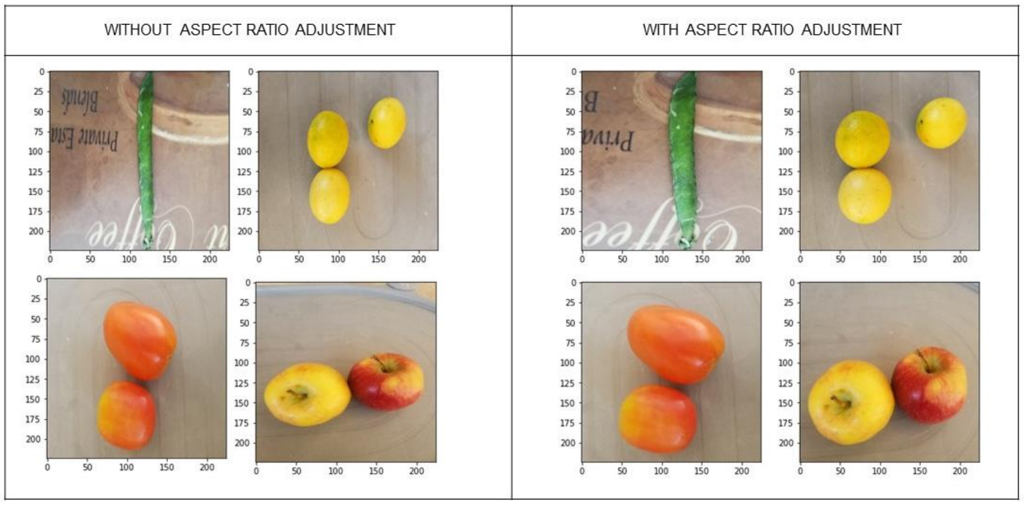


Figure 3.6: Aspect ratios of images in the dataset

3.3 Recognition of Fruits/Vegetables:

In this module, different models have been trained for the classification of fruits and vegetables. The research used the models which are popular for fruit image processing (Naranjo-Torres et al., 2020). Following are the models along with the details that have been trained.

3.3.1 AlexNet:

AlexNet is a popular convolutional neural network that is 8 layers deep. It has five convolutional layers, three pooling layers, one flattening layer, three fully connected layers. The pooling layers are performing max pooling. Batch Normalization layers and dropout layers are also used to avoid overfitting. All the activation functions used in convolution and fully connected layers are ReLU except for the output layer that has softmax activation function. The model is trained for 15 epochs. The training is being monitored for minimum validation loss, if the loss does not improve in an epoch the learning rate is reduced by a factor of 0.2. Still if the validation loss does not decrease for 3 consecutive epochs the training stops and the best model is picked up and saved. Figure 3.7 shows the model.

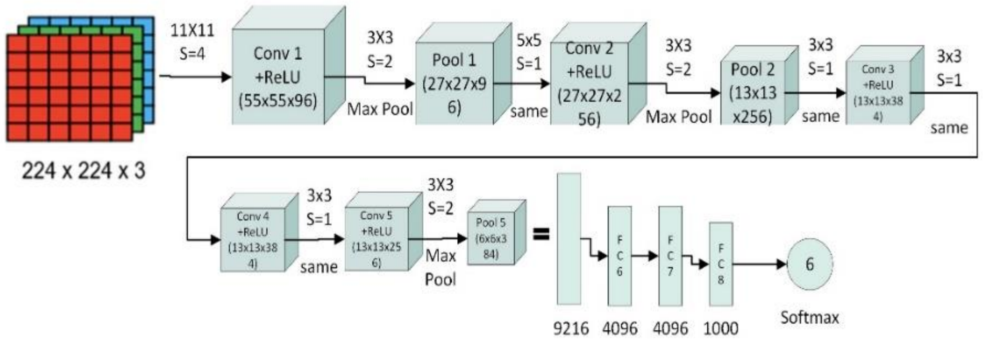


Figure 3.7: AlexNet

3.3.2 Fruits-360 Model

The research also has used Fruits-360 model (Moresan and Oltean, 2018) to see how it performs on the data. The model is 6 layers deep and looks like a modification of the AlexNet model (figure 3.7). The training process is monitored for maximum validation accuracy and if the accuracy does not improve for 5 consecutive epochs, training stops, and the model is saved. Figure 3.8 depicts the model.

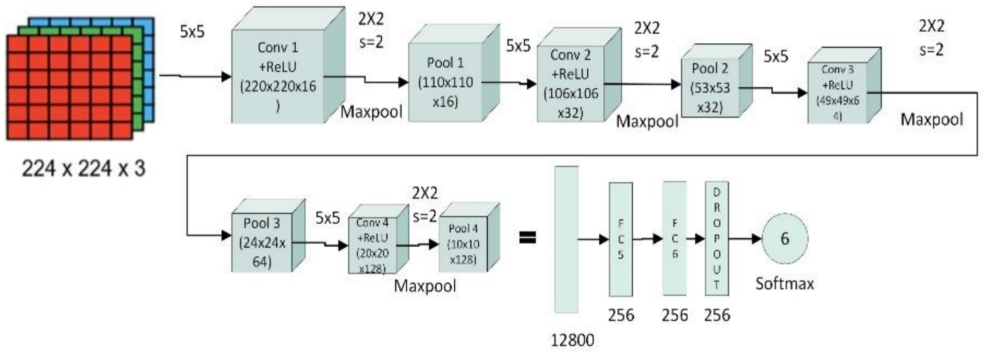


Figure 3.8: Fruits-360 Model

3.3.3 Own Model

An attempt has been made to make a simpler model and see its performance on the data. It is 5 layers deep and has more pooling and dropout layers. All activation functions are ReLU except for the output layer. The training process is monitored for maximum validation accuracy and if the accuracy does not improve for 5 consecutive epochs, training stops, and the model is saved. Figure 3.9 shows the model architecture.

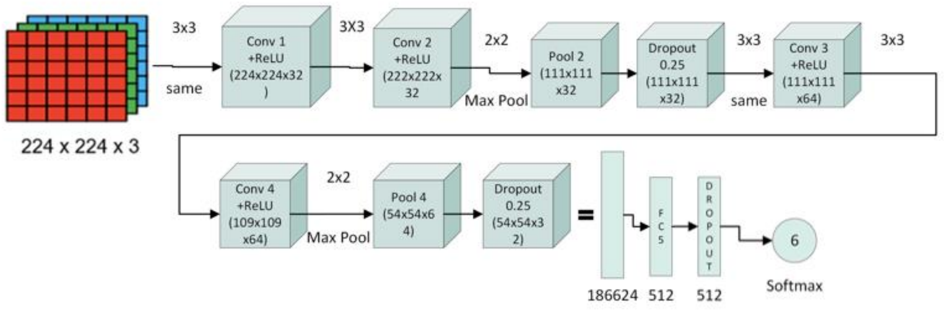


Figure 3.9: Own Model

3.3.4 VGG-16

This model has been used as a pretrained version. The training has been done using transfer learning. Since the dataset was not a very large one so transfer learning yielded better performance with smaller datasets (Tan et al., 2018). As depicted in figure 3.10 a fully connected network is designed and connected to the pretrained (trained on Image-Net dataset, (Deng et al., 2009)) vgg-16 base. The trainable layers are set to false. So, the model trained really fast with a good accuracy of 97.7%. The training was monitored for maximum validation accuracy for 5 epochs. If the accuracy does not increase further the training stops.

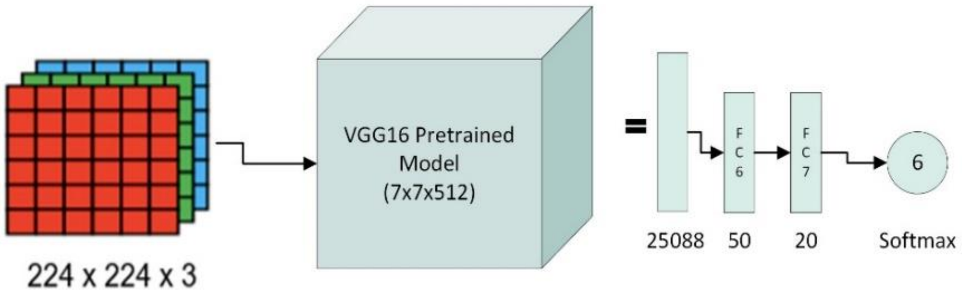


Figure 3.10: VGG-16 through transfer learning

3.3.5 ResNet-50

ResNet-50 model was also used as a transfer learning model with trainable layers set to false. The model's convolutional layers were pretrained on the Image-Net database, (Deng et al., 2009). The training was done on the fully connected layers with a good enough accuracy of 98.9%. The training was monitored for maximum validation accuracy for 5 consecutive epochs, if the accuracy does not improve the training stops. Figure 3.11 shows the model as a pre-trained base with a new fully connected network. The training is being monitored for minimum validation loss, if the loss does not improve in an epoch, the learning rate is reduced by a factor of 0.2. And if the loss does not decrease for 3 consecutive epochs the training stops and the best model is saved.

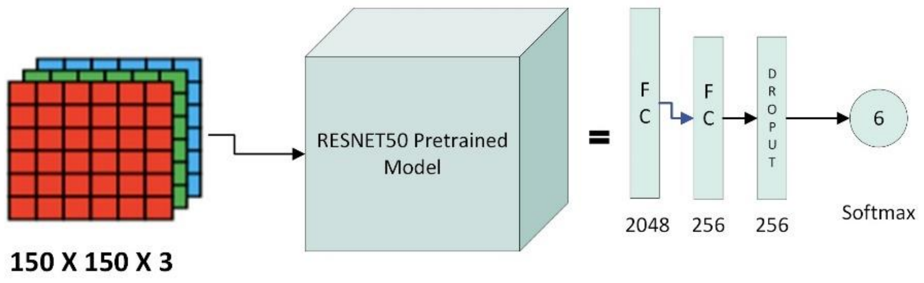


Figure 3.11: ResNet-50 through transfer learning

3.4 Quality Assessment of Fruits/Vegetables:

After the fruits and vegetables are recognized, the next step is to evaluate the quality of the recognized fruit/vegetable. Quality assessment, as emphasized earlier in section 1.4, is a significant need. The quality assessment problem has three categories or 3 classes namely: **Eatable**, **Partially Rotten** (when rottenness has started and is just in the initial stages), and **Rotten.** Though the classes are 3 but the partially rotten stage is equally similar to the eatable stage and rotten stage. But at the same time, it is important to identify this class because the fruit/vegetable quality has been compromised, and hence is necessary to be identified. The model used is a VGG-16 pretrained model through transfer learning. The model is trained separately for all the fruits/vegetables and saved separately for each of them. The training process is monitored for maximum validation accuracy. Total no of epochs for training is 15 but if the accuracy does not improve further for five consecutive epochs the training stops and the model is saved (Figure 3.12). The results for all the fruits and vegetables along with the accuracy curves, loss curves and confusion matrices are shared in the next chapter. The model uses sparse categorical cross-entropy as the loss function.

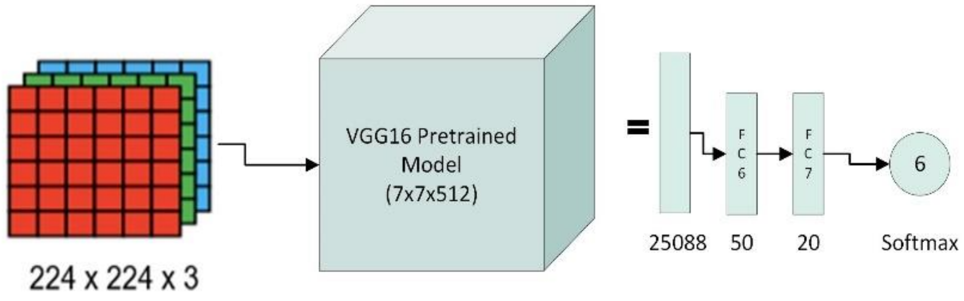


Figure 3.12: VGG-16 through transfer learning for quality assessment

Figure 3.13 shows the entire proposed methodology as a workflow diagram.

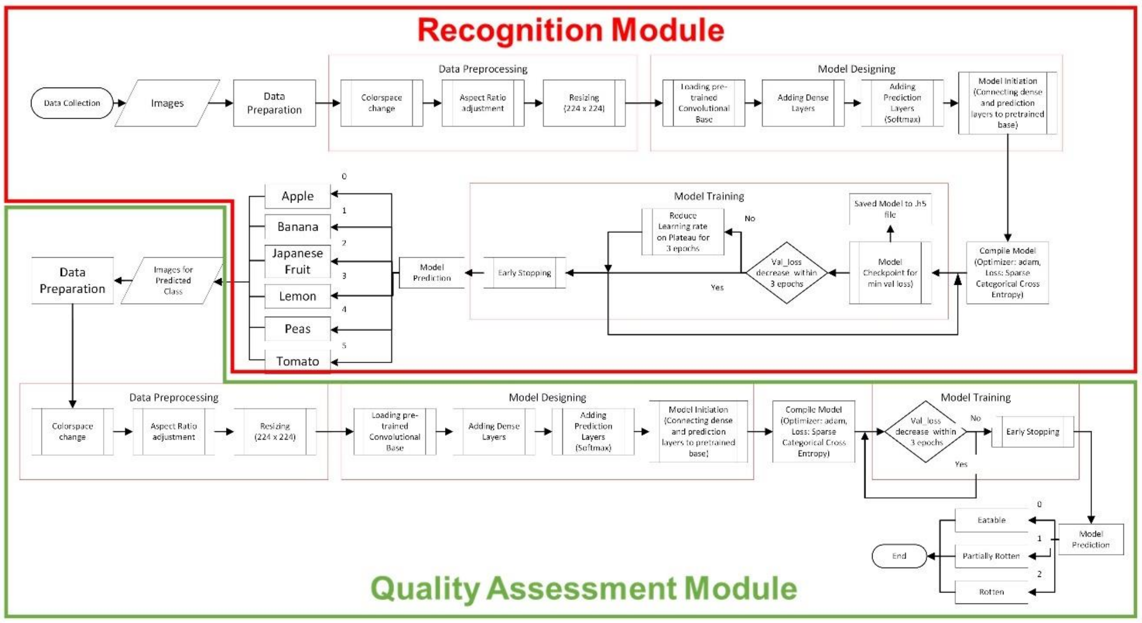

Recognition Module
Data Collection
Images
Data Preparation
Data Preprocessing
Colorspace change
Aspect Ratio adjustment
Resizing (224 x 224)
Model Designing
Loading pre-trained Convolutional Base
Adding Dense Layers
Adding Prediction Layers (Softmax)
Model Initiation (Connecting dense and prediction layers to pretrained base)
Compile Model (Optimizer: adam, Loss: Sparse Categorical Cross Entropy)
Model Training
Model Checkpoint for min val loss)
Saved Model to .h5 file
Val_loss decrease within 3 epochs
No
Yes
Reduce Learning rate on Plateau for 3 epochs
Early Stopping
Model Prediction
0
Apple
1
Banana
2
Japanese Fruit
3
Lemon
4
Peas
5
Tomato
Images for Predicted Class
Data Preparation
Data Preprocessing
Colorspace change
Aspect Ratio adjustment
Resizing (224 x 224)
Model Designing
Loading pre-trained Convolutional Base
Adding Dense Layers
Adding Prediction Layers (Softmax)
Model Initiation (Connecting dense and prediction layers to pretrained base)
Compile Model (Optimizer: adam, Loss: Sparse Categorical Cross Entropy)
Model Training
Val_loss decrease within 3 epochs
No
Yes
Early Stopping
Model Prediction
0
Eatable
1
Partially Rotten
2
Rotten
End
Quality Assessment Module

Figure 3.13: Workflow Diagram of Proposed Methodology

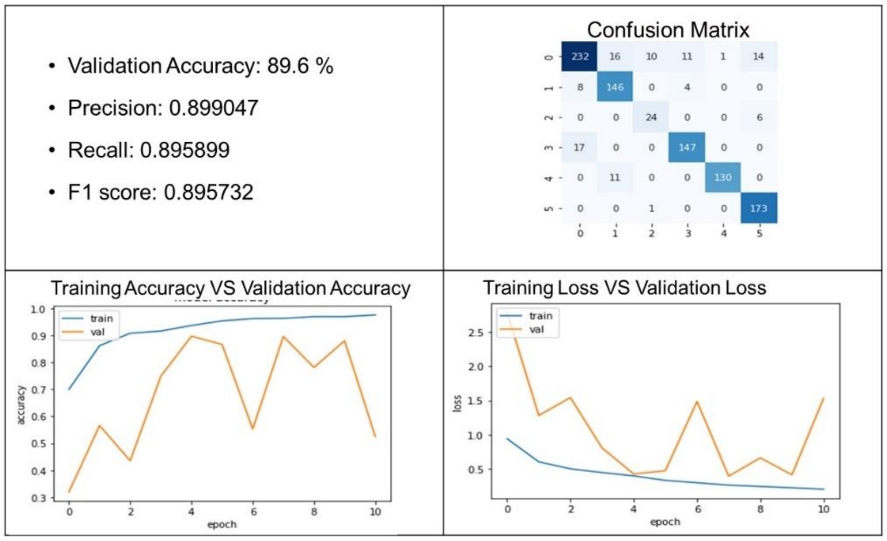
CHAPTER 4: RESULTS

All results obtained after training the models depicted in chapter 3 are displayed in the following sections.

4.1 Results for Recognition of Fruits/Vegetables:

4.1.1 AlexNet

The model performed well on the training with a training accuracy of 96.3%. The validation accuracy was a bit lesser (89.6%). Figure 4.1 shows the performance metrics along with the confusion matrix. The training accuracy vs validation accuracy and training loss vs validation loss curves are also shown in figure 4.1.



- Validation Accuracy: 89.6 %
- Precision: 0.899047
- Recall: 0.895899
- F1 score: 0.895732

Confusion Matrix

	0	1	2	3	4	5
0	232	16	10	11	1	14
1	8	146	0	4	0	0
2	0	0	24	0	0	6
3	17	0	0	147	0	0
4	0	11	0	0	130	0
5	0	0	1	0	0	173

Figure 4.1: Results – Recognition using AlexNet

4.1.2 Fruits-360 Model:

The fruits-360 model (Moresan and Oltean, 2018) was used to train for recognition of fruits and vegetables yielding surprisingly good results with 94.9% training accuracy and validation accuracy of 92.01%. Figure 4.2 shows the results graphically as well.

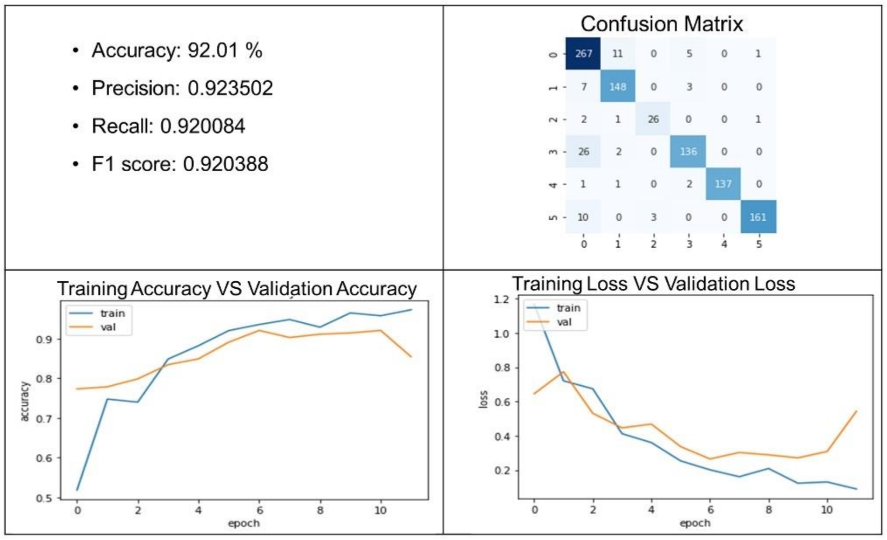

- Accuracy: 92.01 %
- Precision: 0.923502
- Recall: 0.920084
- F1 score: 0.920388

Figure 4.2: Results – Recognition using Fruits-360 Model

4.1.3 Own Model:

This is the simplest model used for training with 5 layers and gives 88.4% accuracy which is fine enough compared to the complexity of remaining models.

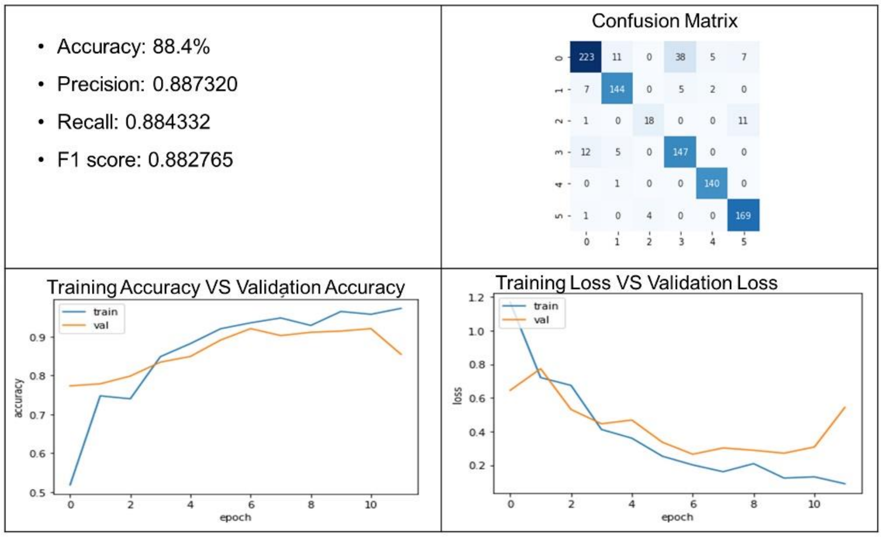

- Accuracy: 88.4%
- Precision: 0.887320
- Recall: 0.884332
- F1 score: 0.882765

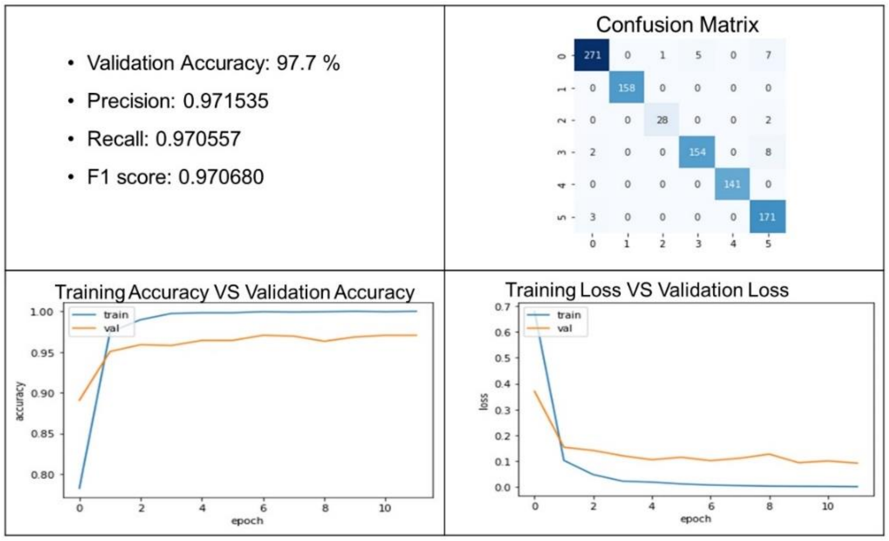
Figure 4.3: Results – Recognition using Own Model

4.1.4 VGG-16:

VGG-16 was used as a pretrained base using transfer learning. A fully connected network was then attached to the pretrained base. The resulting model got training accuracy of 99.5% which is the best training accuracy out of all the models used. Results are depicted in figure 4.4.

4.1.5 ResNet-50:

The ResNet-50 model was used using transfer learning. The trainable layers were set to false and they were used as a pretrained base as shown in figure 3.11, with a new fully connected network.



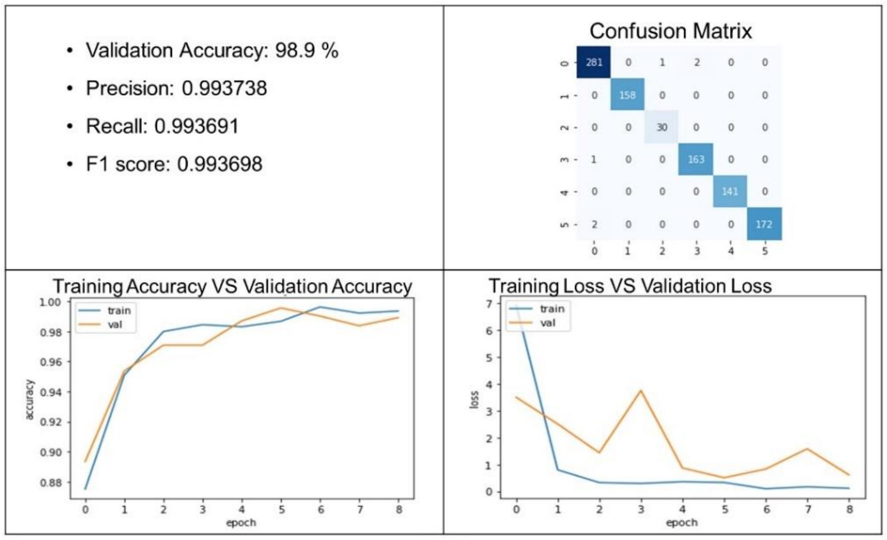
- Validation Accuracy: 97.7 %
- Precision: 0.971535
- Recall: 0.970557
- F1 score: 0.970680

Confusion Matrix

	0	1	2	3	4	5
0	271	0	1	5	0	7
1	0	158	0	0	0	0
2	0	0	28	0	0	2
3	2	0	0	154	0	8
4	0	0	0	0	141	0
5	3	0	0	0	0	171

Figure 4.4: Results – Recognition using VGG-16

The model achieved a very good accuracy of 98.9%. A summary of results is depicted in figure 4.5.



- Validation Accuracy: 98.9 %
- Precision: 0.993738
- Recall: 0.993691
- F1 score: 0.993698

Confusion Matrix

	0	1	2	3	4	5
0	281	0	1	2	0	0
1	0	158	0	0	0	0
2	0	0	30	0	0	0
3	1	0	0	163	0	0
4	0	0	0	0	141	0
5	2	0	0	0	0	172

Figure 4.5: Results – Recognition using ResNet-50

4.1.6 Comparison of Recognition Models

Table 4.1 shows the comparison of all the models for recognition of fruits/vegetables. It can be observed that VGG-16 and ResNet-50 both obtained almost equal accuracies but in terms of both training and validation losses, VGG-16 performed better than ResNet-50.

Table 4.1: Comparison of Recognition Models

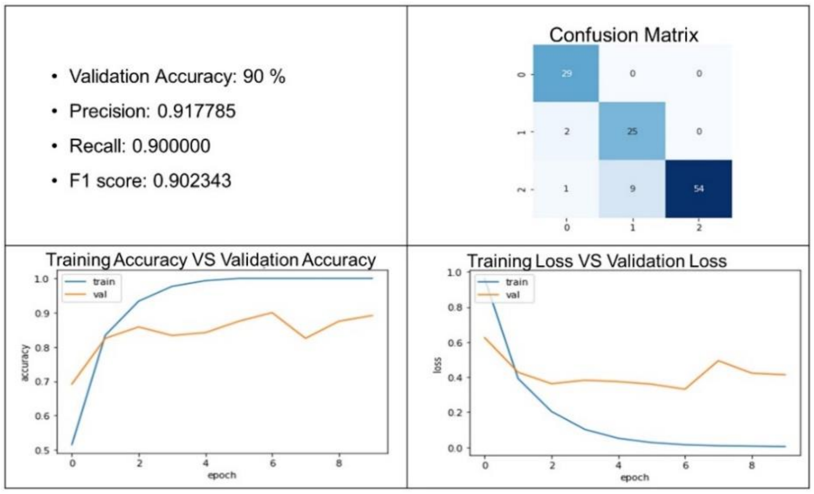
Model	Image Size	Training		Validation	
		Accuracy	Loss	Accuracy	Loss
Modified AlexNet	224 x 224	94.9 %	0.17	93.4 %	0.22
Fruits-360 Model	224 x 224	94.9 %	0.15	92.01 %	0.33
VGG16	**224 x 224**	**99.5 %**	**0.02**	**97.7 %**	**0.06**
ResNet50	**150 x 150**	**99.1 %**	**0.11**	**98.9 %**	**0.61**
AlexNet	224 x 224	96.3 %	0.26	89.6 **%**	0.39

4.2 Results for Quality Assessment of Fruits/Vegetables:

The results for all six fruits and vegetables are shown in detail in the following sections.

4.2.1 Quality Assessment of Banana:

As can be seen in figure 4.6 that VGG-16 trained well on the dataset for quality assessment of



- Validation Accuracy: 90 %
- Precision: 0.917785
- Recall: 0.900000
- F1 score: 0.902343

Figure 4.6: Results – Quality Assessment of Banana

banana. The training accuracy was 99% and the validation accuracy came to be 90%.

Since the partially rotten class is an intermediate class between eatable and rotten so there is a little confusion in partially rotten and rotten state as can be seen in the confusion matrix but overall, the model performed very well.

4.2.2 Quality Assessment of Persimmon:

Quality assessment was done using the model shown in figure 3.12. Overall, the model performed good with an accuracy of 92%. Rest of the results are displayed in figure 4.7.

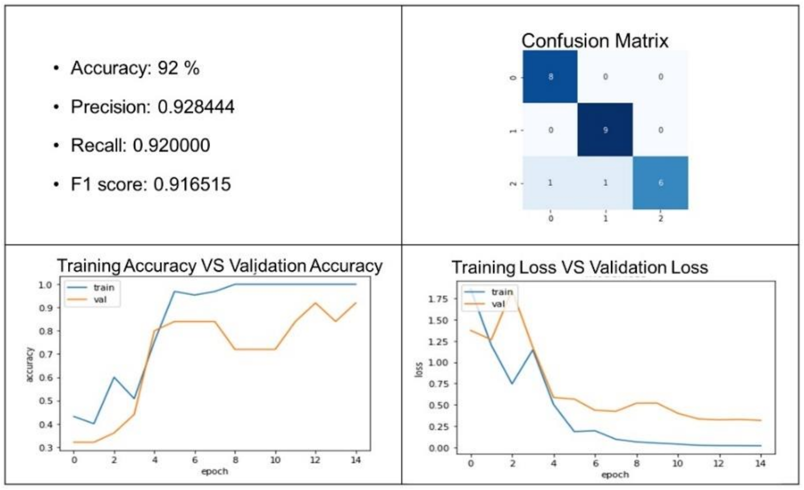


Figure 4.7: Results – Quality Assessment of Persimmon

4.2.3 Quality Assessment of Apple:

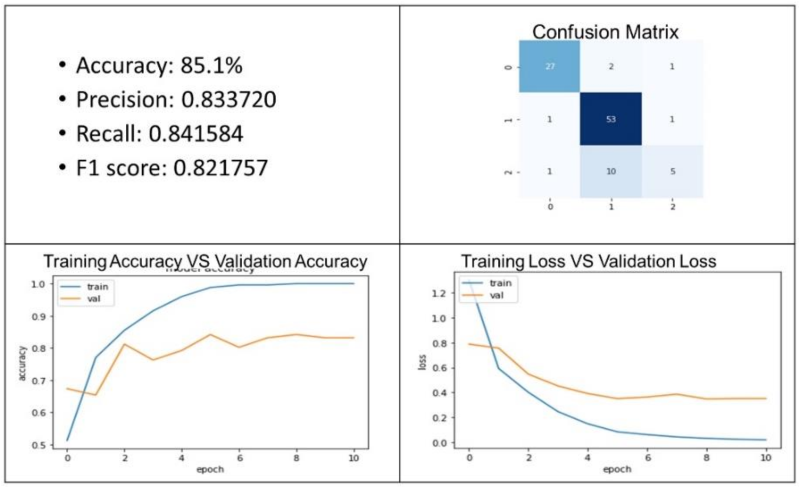
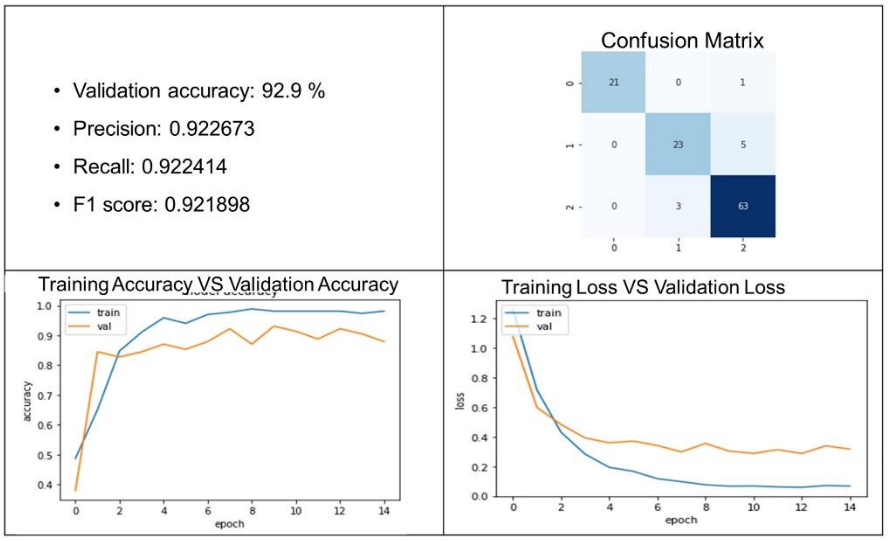


Figure 4.8: Results – Quality Assessment of Apple

The results of training VGG-16 through transfer learning for quality assessment data for apple are shown in the figure 4.8. The accuracy achieved is 85.1%.

4.2.4 Quality Assessment of Peas:

The VGG-16 model performed well on quality assessment data for peas. The accuracy achieved is 92.9% highest among other fruits and vegetables. The results can be observed in figure 4.9.



- Validation accuracy: 92.9 %
- Precision: 0.922673
- Recall: 0.922414
- F1 score: 0.921898

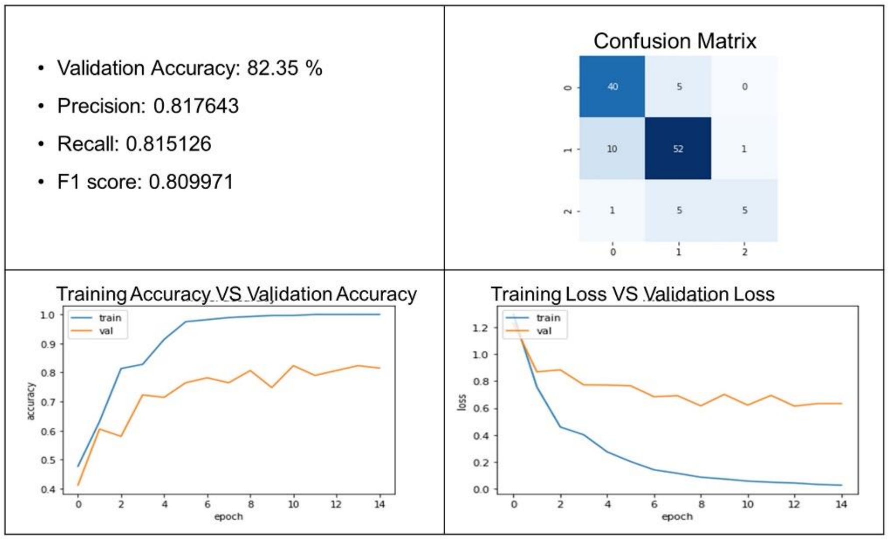
Confusion Matrix

	0	1	2
0	21	0	1
1	0	23	5
2	0	3	63

Figure 4.9: Results – Quality Assessment of Peas

4.2.5 Quality Assessment of Lemon

Figure 4.10 shows the results. The accuracy achieved was 82.4%.



- Validation Accuracy: 82.35 %
- Precision: 0.817643
- Recall: 0.815126
- F1 score: 0.809971

Confusion Matrix

	0	1	2
0	40	5	0
1	10	52	1
2	1	5	5

Figure 4.10: Results – Quality Assessment of Lemon

The model did not perform as well as it performed on other fruits and vegetables with around 90% accuracy.

4.2.6 Quality Assessment of Tomato

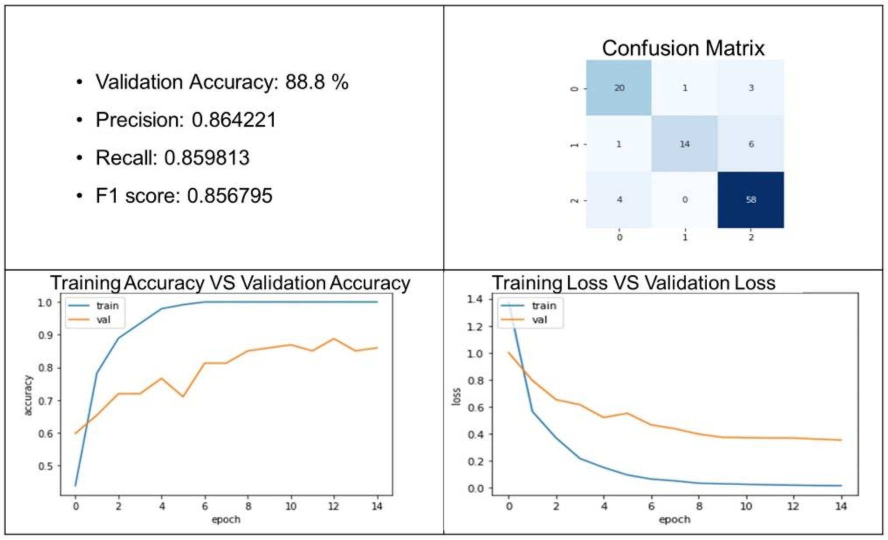


Figure 4.11: Results – Quality Assessment of Tomato

All the performance metrics of the model were good. The results are as shown in figure 4.11.

Table 4.2 gives the summary of performance of quality assessment model for all fruits and vegetables.

Table 4.2: Summary of Quality Assessment

Sr. No.	Fruit/Vegetable	Accuracy	Precision	Recall	F1-score
1	**Banana**	90.0%	0.9178	0.9000	0.9023
2	**Persimmon**	92.0%	0.9284	0.9200	0.9165
3	**Apple**	85.1%	0.8337	0.8416	0.8218
4	**Peas**	92.9%	0.9227	0.9224	0.9219
5	**Lemon**	82.4%	0.8176	0.8151	0.8099
6	**Tomato**	88.8%	0.8642	0.8598	0.8568

CHAPTER 5: DISCUSSION

This research has been successful in collecting a new dataset that is very different from the datasets already presented in other studies. Using the dataset, the research has achieved recognition of fruits/vegetables and finally has performed a successful quality assessment on fruits and vegetables.

This field of research is relatively latest and there is not much work done on the quality assessment of fruits/vegetables. The literature research reveals that though there has been a latest surge in the research regarding food recognition and quality assessment but, as far as deep learning is concerned, there is very little research available for quality assessment of fruits and vegetables. Usually, images used for food recognition are acquired through thermal imaging (Bhole and Kumar, 2020), hyper spectral imaging (Maria and Darwin, 2021), magnetic resonance imaging, electrical tomography (Bhargava and Bansal, 2021), Raman Imaging, Laser backscattering imaging (Pathmanaban et al., 2019) etc. After acquiring images through these techniques they are then subject to preprocessing, and feature extraction. The process of food image processing has been carried out using techniques of computer vision (Bhargava and Bansal, 2021), machine learning algorithms like logistic regression, artificial neural networks (ANN), support vector machine (SVM) as in (Bhargava et al., 2022) and (Antony and Saatheesh, 2021), K-nearest neighbor (KNN), Decision Tree algorithm (Ren et al, 2019).

Images captured from such sources as mentioned in the previous paragraph render it difficult to relate to daily life or use the model in any future platform to be used by humans in daily life. The data set collected is collected through smartphone camera and thus captures the details that can be captured by the smartphone camera that everyone has. In (Nasiri et al.,) an attempt has been made to classify healthy and defective dates. And images are captured through smartphone camera, but the work is limited to a type of date mostly found in middle east (Shahani). Throughout the dataset the background of the images was kept constant unlike the dataset introduced and used in this research. Furthermore, in (Chakraborty et al., 2021) they have used a dataset of three fruits from Kaggle. From the dataset it can be clearly observed that it has been taken under strictly controlled environment with respect to different features. All images are taken with almost same background and extremely good constant illumination. So, there is no chance of background clutter. And in both these researches, almost all images have constant aspect ratios.

Unlike the researches mentioned above the dataset used in this research have different aspect ratios and backgrounds are not kept constant to provide the model with a chance to encounter objects with different backgrounds because in real life all items cannot be found against the same background. One more feature that sets this dataset apart from all the researches done before is that images of fruits and vegetables are also collected in groups since all fruits and vegetables are actually present in groups and so there may not be a single object of attention in one frame. This point is kept in mind, since this research can be rendered into a more general platform with less limiting conditions.

There have been various models used to train over this dataset and despite of the images being in groups and varying backgrounds with background clutter, the models have performed remarkably well for recognition with highest accuracy on validation set to be 98.9% and very good for quality assessment with the highest of 92.9 % for peas. This is the first of its kind research that has attempted to directly classify **fruit/vegetables images in groups** into **three quality labels**: **Eatable, Partially Rotten and Rotten.**

CHAPTER 6: CONCLUSION

This research is based on the dataset consisting of fruits and vegetables. To the best of our knowledge this is the only quality assessment research carried out using a Pakistani dataset. The dataset has been collected from smartphone camera, as part of the research, in uncontrolled environment with respect to illumination, image resolution, image aspect ratios, capturing angles and background. Images contain fruit and vegetables individually as well as in groups. The dataset is then used to perform image recognition using the following models: AlexNet, Own (Modified AlexNet), Fruits-360 Model, VGG-16 and ResNet-50. The techniques of transfer learning have been used to cater a small-scale dataset. The best accuracies have been obtained using transfer learning on VGG-16 and ResNet-50 (97.7% and 98.9% respectively). After recognition, the next step was to perform quality assessment on the recognized fruit/vegetable. The model used for this is VGG-16 with the highest accuracy of 92.9% for peas.

CHAPTER 7: FUTURE WORK

The research work is based on the dataset collected for fruits and vegetables. This is a small-scale dataset of six categories of fruits and vegetables. As a future work, this dataset needs to be increased including more fruits and vegetables in accordance with Pakistani needs. Increasing the dataset will make the research more inclusive. The number of images for each category can be increased which will boost up the accuracies for quality assessment module.

Temperature and Humidity percentages have been recorded for data samples collected at each time of the day, but they have not been incorporated into the current research. These recordings need to be incorporated for further advancement in decision-making of the quality of fruits and vegetables and their obvious dependence on these parameters. Furthermore, considering the temperature and humidity percentages can make the performance better with respect to real life.

Moreover, an online platform can be developed which can both be used at the consumer and the seller end for quality validation and assessment.

APPENDIX A

Python Code for AlexNet:

```
import numpy as np # linear algebra
import matplotlib.pyplot as plt # Data visulation
import glob # For including images
import cv2 # OpenCV
import tensorflow as tf # Machine learning lib
from tensorflow import keras #Tensorflow high-level api
import os
from PIL import Image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras import layers, models
from sklearn.metrics import precision_score, accuracy_score, recall_score,
f1_score
import seaborn as sns
from tensorflow.python.keras.utils.vis_utils import plot_model

num_classes = 6
img_rows, img_cols = 224, 224
batch_size = 32
epochs = 15
train_data= []
train_label = []
val_data= []
val_label = []
img_size=[]
new_dim=224

#Loading Data set--------------------------------------------------------------
for dir_path in glob.glob("./Train_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]
        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w,:]
        elif (w==h):
            img_crop=img

        img = cv2.resize(img_crop, (new_dim, new_dim))
        train_data.append(img)
        train_label.append(img_label)
train_data = np.array(train_data)
train_data=train_data/255
train_label = np.array(train_label)

for dir_path in glob.glob("./Val_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w]
        elif (w==h):
            img_crop=img


        img = cv2.resize(img_crop, (new_dim, new_dim))
        val_data.append(img)
        val_label.append(img_label)
val_data = np.array(val_data)
val_data=val_data/255
val_label = np.array(val_label)
l=len(np.unique(val_label))

#--------------------------------------------------------------------------------
label_to_id = {v : k for k, v in enumerate(np.unique(train_label))}
id_to_label = {v : k for k, v in label_to_id.items()}
train_label_id = np.array([label_to_id[i] for i in train_label])
test_label_id = np.array([label_to_id[i] for i in val_label])

#---------------------------------------------------------------------------------
#Model
#Instantiation
AlexNet = Sequential()

#1st Convolutional Layer
AlexNet.add(Conv2D(filters=96, input_shape=(img_rows,img_cols,3),
kernel_size=(11,11), strides=(4,4), padding='same'))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
AlexNet.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#2nd Convolutional Layer
AlexNet.add(Conv2D(filters=256, kernel_size=(5, 5), strides=(1,1),
padding='same'))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
AlexNet.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#3rd Convolutional Layer
AlexNet.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
padding='same'))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

#4th Convolutional Layer
AlexNet.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
padding='same'))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))

#5th Convolutional Layer
AlexNet.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),
padding='same'))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
AlexNet.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

#Passing it to a Fully Connected layer
AlexNet.add(Flatten())
#1st Fully Connected Layer
AlexNet.add(Dense(4096))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
# Add Dropout to prevent overfitting
AlexNet.add(Dropout(0.4))

#2nd Fully Connected Layer
AlexNet.add(Dense(4096))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
#Add Dropout
AlexNet.add(Dropout(0.4))

#3rd Fully Connected Layer
AlexNet.add(Dense(1000))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('relu'))
#Add Dropout
AlexNet.add(Dropout(0.4))

#Output Layer
AlexNet.add(Dense(num_classes))
AlexNet.add(BatchNormalization())
AlexNet.add(Activation('softmax'))
```

```
#Model Summary
AlexNet.summary()

#-----------------------------------------------------------------------------

checkpoint = ModelCheckpoint("AlexNet.h5",
                              monitor="val_loss",
                              mode="min",
                              save_best_only = True,
                              verbose=1)

earlystop = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 3,
                          verbose = 1,
                          restore_best_weights = True)

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss',
                              factor = 0.2,
                              patience = 3,
                              verbose = 1,
                              min_delta = 0.0001)

# we put our call backs into a callback list
callbacks = [earlystop, checkpoint, reduce_lr]
#--------------------------------------------------------------------------
# We use a very small learning rate
AlexNet.compile(loss = 'sparse_categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])

#Saving Model and Recovering--------------------------------------------

history=AlexNet.fit(train_data, train_label_id, batch_size = 32, epochs = 15,
callbacks = callbacks, validation_data=(val_data, test_label_id))

np.save('AlexNet_history.npy',history.history)
#--------------------------------------------------------------------
model = load_model('AlexNet.h5')

#Visualizing model
plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
history=np.load('AlexNet_history.npy',allow_pickle=True).item()
#--------------------------------------------------------------------

#PlotsAccuracy-------------------------------------------------
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#plotloss
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
#----------------------------------------------------------------------
#Conf_Matrix----------------------------------------------------------
y_pred = model.predict(val_data)
y_pred = np.argmax(y_pred, axis=1)

conf_mat = confusion_matrix(test_label_id, y_pred)
sns.heatmap(conf_mat, square=True, annot=True, cmap='Blues', fmt='d',
cbar=False)
#---------------------------------------------------------------------

# accuracy: (tp + tn) / (p + n)------------------------------------------
acc = accuracy_score(test_label_id, y_pred)
print('Accuracy: %f' % acc)

# precision tp / (tp + fp)----------------------------------------
precision = precision_score(test_label_id, y_pred,
                            average='weighted')
print('Precision: %f' % precision)

# recall: tp / (tp + fn)------------------------------------------------
recall = recall_score(test_label_id, y_pred,
                      average='weighted')
print('Recall: %f' % recall)

# f1: 2 tp / (2 tp + fp + fn)------------------------------------------
f1 = f1_score(test_label_id, y_pred,
              average='weighted')
print('F1 score: %f' % f1)

#--------------------------------------------------------------------
```

Python Code for Fruits-360 Model:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Data visulation
import glob # For including images
import cv2 # OpenCV
import tensorflow as tf # Machine learning lib
from tensorflow import keras # Tensorflow high-level api
import os
from PIL import Image
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
```

```
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras import layers, models
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score,
recall_score, f1_score
import seaborn as sns
from tensorflow.python.keras.utils.vis_utils import plot_model

#---------------------------------------------------------------------------
num_classes = 6
img_rows, img_cols = 224, 224
batch_size = 32
epochs = 15
train_data= []
train_label = []
val_data= []
val_label = []
img_size=[]
new_dim=224

#Loading Data set-------------------------------------------------------
for dir_path in glob.glob("./Train_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w,:]
        elif (w==h):
            img_crop=img

        img = cv2.resize(img_crop, (new_dim, new_dim))
        train_data.append(img)
        train_label.append(img_label)
train_data = np.array(train_data)
train_data=train_data/255
train_label = np.array(train_label)


for dir_path in glob.glob("./Val_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w,:]
        elif (w==h):
            img_crop=img

        img = cv2.resize(img_crop, (new_dim, new_dim))
        val_data.append(img)
        val_label.append(img_label)
val_data = np.array(val_data)
val_data=val_data/255
val_label = np.array(val_label)
l=len(np.unique(val_label))

  #--------------------------------------------------------------------

label_to_id = {v : k for k, v in enumerate(np.unique(train_label))}
id_to_label = {v : k for k, v in label_to_id.items()}
train_label_id = np.array([label_to_id[i] for i in train_label])
test_label_id = np.array([label_to_id[i] for i in val_label])


#Model--------------------------------------------------------------

model = Sequential()

# Padding = 'same'  results in padding the input such that
# the output has the same length as the original input
model.add(Conv2D(16,(5,5),activation='relu',input_shape=(img_rows,img_cols,3)))
model.add(MaxPooling2D(pool_size=(2, 2), strides = 2))
model.add(Conv2D(32,(5,5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))


model.add(Conv2D(64,(5,5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides = 2))
model.add(Conv2D(128,(5,5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

```
##----------------------------------------------------------------------

#initiate RMSprop optimizer and configure some parameters
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
print(model.summary())

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'],
)

es = EarlyStopping(monitor='val_accuracy', mode='max', patience=5,
restore_best_weights=True)

history=model.fit(train_data, train_label_id, batch_size = 32, epochs = 15,
callbacks = [es], validation_data=(val_data, test_label_id))
np.save('Fruit360_Model_history.npy',history.history)
model.save('Fruit360_Model.h5')

#----------------------------------------------------------------------

history=np.load('Fruit360_Model_history.npy',allow_pickle=True).item()

#PlotsAccuracy---------------------------------------------------------
plt.figure()
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#plotloss--------------------------------------------------------------
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#---------------------------------------------------------------------
# #Conf_Matrix--------------------------------------------------------

model = load_model('Fruit360_Model.h5')
y_pred = model.predict(val_data)
y_pred = np.argmax(y_pred, axis=1)

conf_mat = confusion_matrix(test_label_id, y_pred)
sns.heatmap(conf_mat, square=True, annot=True, cmap='Blues', fmt='d',
cbar=False)


# accuracy: (tp + tn) / (p + n)---------------------------------------
acc = accuracy_score(test_label_id, y_pred)
```

```
print('Accuracy: %f' % acc)

# precision tp / (tp + fp)---------------------------------------------------
precision = precision_score(test_label_id, y_pred,
                            average='weighted')
print('Precision: %f' % precision)

# recall: tp / (tp + fn)-------------------------------------------------
recall = recall_score(test_label_id, y_pred,
                      average='weighted')
print('Recall: %f' % recall)

# f1: 2 tp / (2 tp + fp + fn)----------------------------------------------
f1 = f1_score(test_label_id, y_pred,
              average='weighted')
print('F1 score: %f' % f1)

plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
```

Python Code for Own Model:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Data visulation
import glob # For including images
import cv2 # OpenCV
import tensorflow as tf # Machine learning lib
from tensorflow import keras # Tensorflow high-level api
import os
from PIL import Image
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras import layers, models
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score,
recall_score, f1_score
import seaborn as sns
from tensorflow.python.keras.utils.vis_utils import plot_model

#-------------------------------------------------------------------
num_classes = 6
img_rows, img_cols = 224, 224
batch_size = 32
epochs = 15
train_data= []
```

```
train_label = []
val_data= []
val_label = []
img_size=[]
new_dim=224

# #Loading Data set---------------------------------------------------

for dir_path in glob.glob("./Train_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w,:]
        elif (w==h):
            img_crop=img

        img = cv2.resize(img_crop, (new_dim, new_dim))
        train_data.append(img)
        train_label.append(img_label)
train_data = np.array(train_data)
train_data=train_data/255
train_label = np.array(train_label)


for dir_path in glob.glob("./Val_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w]
        elif (w==h):
            img_crop=img
        img = cv2.resize(img_crop, (new_dim, new_dim))
        val_data.append(img)
        val_label.append(img_label)
val_data = np.array(val_data)
val_data=val_data/255
val_label = np.array(val_label)
l=len(np.unique(val_label))

  #------------------------------------------------------------------

label_to_id = {v : k for k, v in enumerate(np.unique(train_label))}
id_to_label = {v : k for k, v in label_to_id.items()}
train_label_id = np.array([label_to_id[i] for i in train_label])
test_label_id = np.array([label_to_id[i] for i in val_label])


model = Sequential()

# Padding = 'same'  results in padding the input such that
# the output has the same length as the original input
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape= (img_rows, img_cols, 3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))

#--------------------------------------------------------------------

# initiate RMSprop optimizer and configure some parameters
#opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
print(model.summary())

model.compile(
    optimizer=RMSprop(lr = 0.001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'],
)
```

```
es = EarlyStopping(monitor='val_accuracy', mode='max', patience=5,
restore_best_weights=True)

history=model.fit(train_data, train_label_id, batch_size = 32, epochs = 15,
callbacks = [es], validation_data=(val_data, test_label_id))
np.save('Own_history.npy',history.history)
model.save('Own.h5')

#--------------------------------------------------------------------------
history=np.load('Own_history.npy',allow_pickle=True).item()
print(history['accuracy'])
model = load_model('Own.h5')
res=model.score()
#PlotsAccuracy-------------------------------------------------------
plt.figure()
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#plotloss-----------------------------------------------------------------
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#--------------------------------------------------------------------------

#Conf_Matrix---------------------------------------------------------

model = load_model('Own.h5')
y_pred = model.predict(val_data)
y_pred = np.argmax(y_pred, axis=1)

conf_mat = confusion_matrix(test_label_id, y_pred)
sns.heatmap(conf_mat, square=True, annot=True, cmap='Blues', fmt='d',
cbar=False)


# accuracy: (tp + tn) / (p + n)----------------------------------------
acc = accuracy_score(test_label_id, y_pred)
print('Accuracy: %f' % acc)

# precision tp / (tp + fp)--------------------------------------------
precision = precision_score(test_label_id, y_pred,
                            average='weighted')
print('Precision: %f' % precision)

# recall: tp / (tp + fn)----------------------------------------------
recall = recall_score(test_label_id, y_pred,
                      average='weighted')
```

```
print('Recall: %f' % recall)

# f1: 2 tp / (2 tp + fp + fn)--------------------------------------------
f1 = f1_score(test_label_id, y_pred,
              average='weighted')
print('F1 score: %f' % f1)

plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
```

Python Code for VGG-16 Model:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Data visulation
import glob # For including images
import cv2 # OpenCV
import tensorflow as tf # Machine learning lib
from tensorflow import keras # Tensorflow high-level api
import os
from PIL import Image
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras import layers, models
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score,
recall_score, f1_score
import seaborn as sns
from tensorflow.python.keras.utils.vis_utils import plot_model

#---------------------------------------------------------------------
num_classes = 6
img_rows, img_cols = 224, 224
batch_size = 32
epochs = 15
train_data= []
train_label = []
val_data= []
val_label = []
img_size=[]
new_dim=224

#Loading Data set-----------------------------------------------------

for dir_path in glob.glob("./Train_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w,:]
        elif (w==h):
            img_crop=img

        img = cv2.resize(img_crop, (new_dim, new_dim))
        train_data.append(img)
        train_label.append(img_label)
train_data = np.array(train_data)
train_data=train_data/255
train_label = np.array(train_label)


for dir_path in glob.glob("./Val_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w,:]
        elif (w==h):
            img_crop=img


        img = cv2.resize(img_crop, (new_dim, new_dim))
        val_data.append(img)
        val_label.append(img_label)
val_data = np.array(val_data)
val_data=val_data/255
val_label = np.array(val_label)
l=len(np.unique(val_label))

  #---------------------------------------------------------------
```

```
label_to_id = {v : k for k, v in enumerate(np.unique(train_label))}
id_to_label = {v : k for k, v in label_to_id.items()}
train_label_id = np.array([label_to_id[i] for i in train_label])
test_label_id = np.array([label_to_id[i] for i in val_label])

#----------------------------------------------------------------
## Loading VGG16 model
base_model = VGG16(weights="imagenet", include_top=False,
input_shape=(img_rows, img_cols, 3))
base_model.trainable = False ## Not trainable weights
base_model.summary()
flatten_layer = layers.Flatten()
dense_layer_1 = layers.Dense(50, activation='relu')
dense_layer_2 = layers.Dense(20, activation='relu')
prediction_layer = layers.Dense(6, activation='softmax')

model = models.Sequential([
    base_model,
    flatten_layer,
    dense_layer_1,
    dense_layer_2,
    prediction_layer
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'],
)


es = EarlyStopping(monitor='val_accuracy', mode='max', patience=5,
restore_best_weights=True)

history=model.fit(train_data, train_label_id, batch_size = 32, epochs = 15,
callbacks = [es], validation_data=(val_data, test_label_id))
np.save('vgg_history.npy',history.history)
model.save('vgg.h5')

#----------------------------------------------------------------

history=np.load('vgg_history.npy',allow_pickle=True).item()

#PlotsAccuracy---------------------------------------------------
plt.figure()
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#plotloss--------------------------------------------------------
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#-------------------------------------------------------------------

#Conf_Matrix-------------------------------------------------------

model = load_model('vgg.h5')
y_pred = model.predict(val_data)
y_pred = np.argmax(y_pred, axis=1)

conf_mat = confusion_matrix(test_label_id, y_pred)
sns.heatmap(conf_mat, square=True, annot=True, cmap='Blues', fmt='d',
cbar=False)


# accuracy: (tp + tn) / (p + n)------------------------------------
acc = accuracy_score(test_label_id, y_pred)
print('Accuracy: %f' % acc)

# precision tp / (tp + fp)----------------------------------------
precision = precision_score(test_label_id, y_pred,
                            average='weighted')
print('Precision: %f' % precision)

# recall: tp / (tp + fn)------------------------------------------
recall = recall_score(test_label_id, y_pred,
                      average='weighted')
print('Recall: %f' % recall)

# f1: 2 tp / (2 tp + fp + fn)---------------------------------------
f1 = f1_score(test_label_id, y_pred,
              average='weighted')
print('F1 score: %f' % f1)

plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
```

Python Code for ResNet-50 Model:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Data visulation
import glob # For including images
import cv2 # OpenCV
import tensorflow # Machine learning lib
from tensorflow import keras # Tensorflow high-level api
import os
from PIL import Image
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import RMSprop, SGD
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, 
ReduceLROnPlateau
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras import layers, models
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score, 
recall_score, f1_score
import seaborn as sns

#------------------------------------------------------------------
num_classes = 6
img_rows, img_cols = 224, 224
batch_size = 32
epochs = 15
train_data= []
train_label = []
val_data= []
val_label = []
img_size=[]
new_dim=224

#Loading Data set--------------------------------------------------

for dir_path in glob.glob("./Train_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w,:]
        elif (w==h):
            img_crop=img

        img = cv2.resize(img_crop, (new_dim, new_dim))
        train_data.append(img)
        train_label.append(img_label)
train_data = np.array(train_data)
train_data=train_data/255
train_label = np.array(train_label)
```

```
for dir_path in glob.glob("./Val_data/*"):
    img_label = dir_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(dir_path, "*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        h=img.shape[0]
        w=img.shape[1]
        img_size=(h, w)
        min_dim=min(img_size)
        max_dim=max(img_size)
        temp=max_dim-min_dim
        temp1=int(temp/2)

        if (h<w):
            img_crop=img[0:h,temp1:temp1+min_dim,:]

        elif (w<h):
            img_crop=img[temp1:temp1+min_dim,0:w]
        elif (w==h):
            img_crop=img


        img = cv2.resize(img_crop, (new_dim, new_dim))
        val_data.append(img)
        val_label.append(img_label)
val_data = np.array(val_data)
val_data=val_data/255
val_label = np.array(val_label)
l=len(np.unique(val_label))

 #-------------------------------------------------------------------
train_data=tensorflow.keras.applications.resnet.preprocess_input(train_data)
val_data=tensorflow.keras.applications.resnet.preprocess_input(val_data)
label_to_id = {v : k for k, v in enumerate(np.unique(train_label))}
id_to_label = {v : k for k, v in label_to_id.items()}
train_label_id = np.array([label_to_id[i] for i in train_label])
test_label_id = np.array([label_to_id[i] for i in val_label])

#-------------------------------------------------------------------
model = Sequential()

pretrained_model= tensorflow.keras.applications.ResNet50(include_top=False,
                    input_shape=(img_rows,img_cols,3),
                    pooling='avg',classes=6,
                    weights='imagenet')
for layer in pretrained_model.layers[:-8]:
    layer.trainable = False

model.add(pretrained_model)


model.add(Flatten())
model.add(Dense(units = 256, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(6, activation = 'softmax'))
model.summary()
  # initiate RMSprop optimizer and configure some parameters
```

```
#opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
print(model.summary())

from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau


checkpoint = ModelCheckpoint("ResNet50.h5",
                              monitor="val_loss",
                              mode="min",
                              save_best_only = True,
                              verbose=1)

earlystop = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 3,
                          verbose = 1,
                          restore_best_weights = True)

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss',
                              factor = 0.2,
                              patience = 3,
                              verbose = 1,
                              min_delta = 0.0001)

# we put our call backs into a callback list
callbacks = [earlystop, checkpoint, reduce_lr]
model.compile(
    optimizer=RMSprop(lr = 0.02),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'],
)


history=model.fit(train_data, train_label_id, batch_size = 32, epochs = 15,
callbacks = callbacks, validation_data=(val_data, test_label_id))
np.save('ResNet50_history.npy',history.history)

#--------------------------------------------------------------------

history=np.load('ResNet50_history.npy',allow_pickle=True).item()

#PlotsAccuracy------------------------------------------------------
plt.figure()
plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#plotloss-----------------------------------------------------------
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

#-----------------------------------------------------------------

#Conf_Matrix-----------------------------------------------------

model = load_model('ResNet50.h5')
y_pred = model.predict(val_data)
y_pred = np.argmax(y_pred, axis=1)

conf_mat = confusion_matrix(test_label_id, y_pred)
sns.heatmap(conf_mat, square=True, annot=True, cmap='Blues', fmt='d',
cbar=False)


# accuracy: (tp + tn) / (p + n)-----------------------------------
acc = accuracy_score(test_label_id, y_pred)
print('Accuracy: %f' % acc)

# precision tp / (tp + fp)----------------------------------------
precision = precision_score(test_label_id, y_pred,
                            average='weighted')
print('Precision: %f' % precision)

# recall: tp / (tp + fn)-----------------------------------------
recall = recall_score(test_label_id, y_pred,
                      average='weighted')
print('Recall: %f' % recall)

# f1: 2 tp / (2 tp + fp + fn)-----------------------------------
f1 = f1_score(test_label_id, y_pred,
              average='weighted')
print('F1 score: %f' % f1)
```

REFERENCES

Nasiri, A., Taheri-Garavand, A. & Zhang, Y. (2019). 'Image-based deep learning automated sorting of date fruit', *Postharvest Biology and Technology*. https://doi.org/10.1016/j.postharvbio.2019.04.003

Chakraborty, S., Shamrat, F.J.M.J., Billah, Md. M., Al Jubair, Md., Alaudin, Md. & Ranjan, R. (2021). 'Implementation of Deep Learning Methods to Identify Rotten Fruits', International Conference on Trends in Electronics and Informatics (ICEI). 10.1109/ICOEI51242.2021.9453004

Muresan, H. & Oltean, M. (2021). 'Fruit recognition from images using deep learning'. Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42. https://doi.org/10.48550/arXiv.1712.00580

Xue, Z., Liu, S., Ma, Y., (2020). 'A hybrid deep learning-based fruit classification using attention model and convolution autoencoder'. Complex & Intelligent Systems. https://doi.org/10.1007/s40747-020-00192-x

Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C. & Valenzuela, A. (2020). 'A Review of Convolutional Neural Network Applied to Fruit Image Procesing', Applied Sciences. 10(10), 3443

Bhole, V. & Kumar, A. (2020). 'Mango Quality Grading using Deep Learning Technique: Perspectives from Agriculture and Food Industry'. SIGITE '20: Proceedings of the 21st Annual Conference on Information Technology Education. https://doi.org/10.1145/3368308.3415370

Ren, A., Zahid, A., Zoha, A., Shah, S. A., Imran, M. A., Alomainy, A. & Abbasi, Q. H., (2019). 'Machine Learning Driven Approach Towards the Quality Assessment of Fresh Fruits Using Non-Invasive Sensing'. IEEE Sensors Journal. 2075 - 2083. 10.1109/JSEN.2019.2949528

Nayak, J., Vakula, K., Dinesh, P., Naik, B. and Pelusi, D. (2020). 'Intelligent food processing: Journey from artificial neural network to deep learning'. Computer Science Review. https://doi.org/10.1016/j.cosrev.2020.100297

Maria, T. A. & Darwin, P. (2021). ‘Deep Learning Approach for Food Quality Inspection and Improvement on Hyper Spectral Fruit Images’. Annals of the Romanian Society for Cell Biology. 15682–15696. https://www.annalsofrscb.ro/index.php/journal/article/view/5219

Zhang, X., Yang, J., Lin, T. and Ying, Y. (2021). ‘Food and agro-product quality evaluation based on spectroscopy and deep learning: A review’. Trends in Food Science and Technology 431-441. https://doi.org/10.1016/j.tifs.2021.04.008

Antony, M. A., & Kumar, R. S. (2021). A Comparative Study on Predicting Food Quality using Machine Learning Techniques. 7th International Conference on Advanced Computing and Communication Systems (ICACCS) (Vol. 1, pp. 1771-1776). IEEE. 10.1109/ICACCS51430.2021.9441743

Bhargava, A., Bansal, A., & Goyal, V. (2022). Machine learning–based detection and sorting of multiple vegetables and fruits. Food Analytical Methods, 15(1), 228-242.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C. (2018). A Survey on Deep Transfer Learning. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds) Artificial Neural Networks and Machine Learning – ICANN 2018. Lecture Notes in Computer Science (), vol 11141. Springer, Cham. https://doi.org/10.1007/978-3-030-01424-7_27

J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

Karagiannakos, S., 2021. “Regularization techniques for training deep neural networks”. *The AI Summer*. https://theaisummer.com/regularization/

Bhargava, A., & Bansal, A. (2021). Fruits and vegetables quality evaluation using computer vision: A review. *Journal of King Saud University-Computer and Information Sciences*, 33(3), 243-257. https://doi.org/10.1016/j.jksuci.2018.06.002