PES UNIVERSITY

**(Established under Karnataka Act No. 16 of 2013) 100-ft Ring Road, Bengaluru – 560 085, Karnataka, India**

*Capstone Project Phase-2 Report*
*on*

# Fruits Quality Detection

***Submitted by***

**R Ashwin (PES1UG19EC223)**
**R V N D Parthasarathy (PES1UG19EC224)**
**Shreepoorna D Purohit (PES1UG19EC286)**
**Sneha Bhat (PES1UG19EC303)**

**Aug - Dec 2022**

***under the guidance of***
**Dr. Shruthi M L J**
**Department of ECE**
**PES University**

**FACULTY OF ENGINEERING**
**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING PROGRAM B.TECH**

# Table Of Contents

# List of Figures

| | |
|---|---|
| **Fig 6.1** | Industrial Visit |
| **Fig 7.1** | Final Conveyor Belt System |

# List of Tables

# Chapter 1
# 1 Introduction

Fruits are an essential item for our daily life. Fresh fruits are not only delicious to eat but also a good source of many important vitamins or minerals. The fruits and vegetables have to pass through various stages from harvesting to reach the customer. The stages are harvesting, sorting, classification, and grading. Although sorting and grading can be done by humans, it is inconsistent, time consuming, variable, subjective, onerous, expensive and easily influenced by surroundings. The manual execution of those tasks requires lots of expert resources and a long time. Automation is needed in every aspect of the processing of fruits. A fast, accurate and automated system is essential for industrial applications.


**Figure 1.1:** Manual Separation of Fresh and Rotten Fruits in Industries.

Computer vision and machine learning have earned huge success in solving various automation problems in different industries. The recent approaches in computer vision, especially in the fields of machine learning and deep learning have improved the

efficiency of image classification tasks. Deep learning is a type of machine learning that has several layers and can predict and solve a variety of problems. Convolutional Neural Networks have made significant advances in the field of image processing as they have progressed over time. Deep Neural Networks outperform all other methods and algorithms for recognizing, classifying, and distinguishing between various groups of fruits. Thus our aim is to perform Quality detection of fruits based on Convolutional Neural Networks.

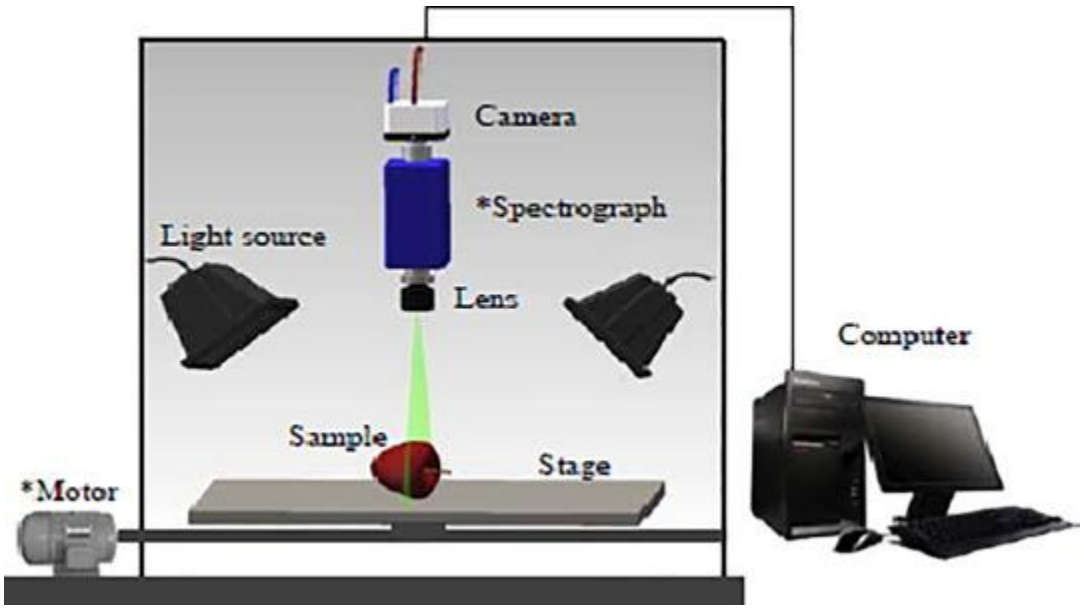



**Figure 1.2:** Usage of Automation for Fruit Quality Detection.

A deep learning based CNN model is used to extract the characteristics or attributes from the dataset created. Further results are compared with transfer learning models such as Inception V3, ResNet-50 and VGG-16.

## 1.1 Motivation

- A huge quantity of fruits are wasted after being purchased every year.
- This is because many people unknowingly buy fruits which are rotten along

with fresh fruits.

- On an average a person discards $520 worth of spoiled fruit per year.
- In a survey of 2,000 Americans, it was discovered that the average respondent wastes three pieces of fruit each week, costing $10 in total.


**Figure 1.3:** Rotten Fruits.

- This is because today's fruit industry performs manual check on fruits and classifies them as rotten and fresh, before distribution.
- This may cause inaccuracies in the classification.
- But, on the other hand, usage of automation in this field helps in increasing the accuracy of classification and at the same time reducing the amount of expert resources and time required.
- Hence, it also helps in reducing the wastage.

# Chapter 2
# Problem Statement

Quality detection of fruits based on Convolutional Neural Networks consisting of a sequential model and further implementation using transfer learning InceptionV3.

## 2.1 The Goal

- Design of CNN based sequential model and transfer learning InceptionV3 model.
- Calculation of performance parameters for both models.
- Comparative analysis of the results obtained from the sequential model and transfer learning model using InceptionV3.
- Development of a hardware prototype for Fruits quality detection based on the transfer learning model InceptionV3.

# Chapter 3
# Literature Survey

In this chapter, we analyze and discuss various literature reviews and survey papers by highlighting the current progress of the field that results in the enhancement and advancement of our research.

Fruits and vegetable quality assessment using computer vision is covered in [1]. The five steps of the methodology include image acquisition, pre-processing, image segmentation, feature extraction, and classification. It is used to inspect the quality of fruits and vegetables. The results showed that computer vision systems should be used in place of manual food inspection since they produce results that are reliable, fair, and non-destructive. The dataset used in this study was "Fruits fresh and rotten for classification," which primarily classified three fruits: apple, oranges, and banana. The limitations of this study were that traditional computer vision systems had difficulty identifying a few defects in the fruit and also the less number of classes for classification(ie apple, oranges, and banana). In order to improve this, we should use multispectral and hyperspectral computer vision systems.

In [2] the author has addressed the Detection of rotten fruits and vegetables using deep learning. The proposed method will be very effective for the automatic detection of rotten fruits and vegetables. The proposed method is completely based on the deep learning technique. Convolutional neural network architecture is designed here for performing the task of classification into a rotten or fresh category of a fruit or vegetable. The proposed CNN model is trained with the images of fresh as well as rotten fruits and vegetables of various types with the corresponding labels. The trained CNN model will detect the rotten fruits and vegetables from an unknown image. Results obtained where Convolutional neural network architecture is proposed here to classify fresh and rotten fruits and vegetables of any type . Performance better than the previous approaches. Performs better for other sets of data that are not present in the dataset . The limitations were that it required a very large amount of data in order to perform better than

other techniques. It is extremely expensive to train due to complex data models. Deep learning requires expensive GPUs and hundreds of machines. The range of fruits which can be classified is limited to apples, bananas and oranges. The dataset used was" Fruits fresh and rotten for classification " which mainly classified 3 fruits namely apple, oranges and banana.

In [3] the author has addressed the Detection of Rotten Fruit Using a One Stage Object Detector. The methodology adopted is the proposed dataset is curated, labeled and augmented. This visual information is processed by RetinaNet to obtain the learning model for inference. The output of inference provides the class and location of fresh and rotten fruits. The results obtained in this paper are a RetinaNet based approach for fresh and rotten fruit detection and classification. To determine the most accurate backbone for RetinaNet, they evaluated ResNet-50, ResNet-101, and VGG-19. From this evaluation they concluded that the best results are obtained using ResNet-50 as the backbone for fresh and rotten fruit detection, attaining an accuracy above 0.98 per class.Limitations of this paper are that most of the classes present have high confidence scores, above 0.9. But in the case of rotten oranges, although the object is well classified and detected, the confidence score is low compared to other classes. The range of fruits which can be classified is limited to apples, bananas and oranges. The dataset used is" Fruits fresh and rotten for classification".

In [4] the author has addressed CNN based rotten fruit detection using ResNet50. Methodology adopted is the CNN algorithm was studied appropriately in order to detect the rotten fruits. Transfer learning technique was used where the model was retrained from a pre-trained CNN network ResNet50. This trained model was then used to perform classification on the rotten and fresh fruits. A total of 2100 fruit images with monotonous backgrounds were trained. The results obtained were the proposed method applied is to use transfer learning to detect the rotten fruits. A good accuracy of 98.89% is obtained in detecting the rotten apple, orange and banana. The limitations were that the range of fruits which can be classified is limited to apples, bananas and oranges. Due to the similarities in the color of apple and orange ,rotten orange was misclassified as rotten apple.The dataset used is "Fruits fresh and rotten for classification" which mainly classified 3 fruits namely apple, oranges and banana.

In [5] the author has addressed Segmentation Techniques for Rotten Fruit Detection. The original images of the fruits are captured and passed through several image segmentation techniques. The images are subjected to RGB2GRAY conversion. After noise filtration and thresholding, the grayscale images undergo several image segmentation processes such as edge detection, marker-based segmentation, and color-based segmentation. Markers are used to identify and highlight the rotten and affected regions in the output image. The drawback is that marker-based segmentation and color-based segmentation techniques depend on pixel values and position of the camera. The size of the marker changes with the variation in pixel values.

In [6] the author has addressed Improving the Prediction of Rotten Fruit Using Convolutional Neural Network. The methodology proposed is the use of VGG16 CNN architecture. Training, validation, and testing are performed on 3000 images of both fresh and rotten fruits. The framework developed from the RGB images dataset is compared with the framework developed from the concatenated images dataset which incorporates the RGB, Laplacian of Gaussian and GraphCut algorithm. Thirteen convolutional layers with Rectified Linear Unit (RELU) activation function, five max-pooling layers, one flatten layer, and three dense layers comprise the generated model. Dense layer uses softmax activation function for classification. After model evaluation, it is observed that concatenated images dataset has higher values for accuracy, precision, recall, F-measure and lower validation loss. The dataset used is "Fruits fresh and rotten for classification" [15].

An Advanced Method of Identification of Fresh and Rotten Fruits Using Different Convolutional Neural Networks is addressed by the author in [7]. The fruits considered here are apple, mango, banana, plum and orange. 80/20 split is applied to raw images. Additionally, the images are reshaped to 224x224x3. The Tensorflow version of Inception V3 is trained using the original ImageNet dataset, comprising more than 1 million training images. The dataset is divided into two portions, with 20% for testing and 80% for training. Validation and preparation happen simultaneously. Inception V3 outperforms the other four models in terms of accuracy, recall, precision, and F1 score. The accuracy obtained is 97.34%. The Xception model comes in second

with a 97.16% accuracy. However, the drawback is that the dataset has lesser data. The accuracy level may be reduced if images include too much noise. The dataset was manually developed.

In [8] the author has put forth a review of Convolutional Neural Network Applied to Fruit Image Processing. Due to the novelty of the usage of CNNs in the examined field, this research contains current literature assessments on CNNs from 2015 to the present. It presents the key elements, properties, and outcomes of the collected works on three major sectors of the agri-food business relating to fruit categorization, quality control, and detection, with the goal of providing a better knowledge of how CNN models are applied. Two practical examples of CNN models for fruit classification are presented. CNN architectures utilized in this research are divided into two broad categories: own and pre-trained networks. Own networks comprise networks built from scratch and pre-trained networks employ well-known models such as AlexNet and GoogLeNet. CNN-based techniques showed exceptional results according to evaluation results. CNNs with fewer than 13 layers produce outstanding results of around 99%. Although CNN models outperform other traditional approaches for fruit classification, they have certain drawbacks. The dataset must be adequately labeled in order to train CNN, handle overfitting issues, and execute the intended task quickly. When proposing a CNN architecture for a specific issue, determining the number of layers and filters, as well as the model's parameters and hyperparameters is an issue that has to be solved by trial-and-error tuning until the best results are obtained. This is time-consuming for very deep models. There has been no study carried out with multiple types of fruit in the same image. The objective is to create a CNN model that can recognise and classify various types of fruit at the same time.

In [9] Fruit Identification and Quality Detection by Means of DAG-CNN has been addressed. A CNN variant, the Directed Acyclic Graph (DAG) structure, has been proposed in recent years. Eight distinct fruits are chosen for database acquisition: banana, lemon, lulo, mango, orange, strawberry, tamarillo, and tomato. Two thousand images of each variety of fruit are captured. DAG CNN comprises two identical branches, with different sizes of filters used to learn additional features in the network. Each branch is composed of four convolution layers, three down sampling layers and two fully connected layers. Finally, a Softmax function is utilised to

determine which category the input image belongs to. The model is later trained and tested. The DAG structure of a CNN allows each of the branches to learn features of varying sizes, improving the network's depth without affecting processing speeds. The proposed approach is more vigorous against disturbances and provides a higher accuracy. Based on the processing speeds, the technique might be incorporated in a real-time machine vision system. However, it is critical to determine if the problem is a disease or physical injury.

[10] illustrates A Comparative Analysis on Fruit Freshness Classification. This research provides a comparison of several feature extraction methods and algorithms for the categorization of fresh and rotten fruits. A publicly available dataset featuring images of fresh and rotten apples, bananas, and oranges is utilized. Segmentation, feature extraction, and classification are among the methodologies deployed. Histogram Features (Hist), Gray Level Co-Occurrence Matrix (GLCM), Bag of Features (BoF), and Convolutional Neural Networks (CNNs) are the four different feature extraction approaches used. To reduce computational time, image features are retrieved using CNNs with ResNet-50 as the pretrained network. After observing that larger batch sizes result in lower accuracy rates and longer computing periods, a mini-batch size of 32 is selected. Hist, GLCM, BoF, and CNNs are used for feature extraction. Classification is carried out using SVMs . CNNs give superior performance and higher success rates when multi-class SVMs are employed for classification. Three sets of trials are conducted, and it is discovered that when it pertains to feature extraction, CNNs outperform other approaches. BoFs demand a significant processing time, and their complexity is cost prohibitive for this operation. Results suggest that without knowing the type of the fruit (i.e., without utilizing another algorithm for fruit classification) it is feasible to identify fruit freshness using CNNs.

[11] presents a thorough analysis of various DL models used for fruit detection and classification. The benchmark datasets and assessment indices are presented in the paper. Large datasets are needed to train DL classifiers. The Fruit 360 dataset was used in this experiment since it is currently the dataset that researchers most frequently use to categorize fruits. For the training and testing sets, the set value remained the same as it had been originally established by the owner. Modified pretrained models are heavily used in the detection job while

trained-from-scratch is heavily used in classification tasks. The choice of hyperparameters has been a significant issue in DL. Here, a thorough analysis of fruit detection and classification techniques, feature description, detection, and classification algorithms, as well as various datasets, is presented.

[12] describes an ensemble model that integrates the bottleneck features of two multi-task deep convolutional neural networks with various designs (ResNet-50 and ResNet-101). A kind of artificial neural networks known as convolutional neural networks (CNNs) use convolutional layers to filter inputs for relevant data. When it is not possible to create a big training dataset, transfer learning can be employed. Since there isn't a workable dataset that has real-world photographs suited for the purpose of classifying fruit and freshness, experiments are conducted here on the new customized dataset, which combines the previous dataset comprising simple images with real-world images crawled from the internet. The dataset is divided into two sections: the simple dataset, which only includes images with white backgrounds, and the real-world dataset, which includes images with all different types of backgrounds. Seven different fruit categories are represented by fresh and rotten fruit images in each partition (apple, banana, lemon, orange, pear, strawberry, and others). Pictures of various fruit varieties that aren't organized into separate categories can be found in the "others" category. The outcomes show that the proposed ensemble method outperforms the schemes without and with fruit type information, as well as the other transfer learning techniques, in terms of classifying the freshness of fruits. To deploy on real-time applications, the model's size should be decreased.

[13] provides a thorough examination of the freshness grading system utilizing computer vision and deep learning. This grading system is based on the visual examination of digital photos. The categorization and location-oriented model (YOLO) and a series of regression CNNs that are specifically targeted at each fruit variety make up the hierarchically created model. The six fruit classes represented in the dataset—apple, banana, dragon fruit, orange, pear, and Kiwi fruits—were drawn from a wide range of places in the photos that included different noises, pointless surrounding objects, and lighting conditions. A total of about 4,000 photos were gathered, with roughly 700 images being obtained for each class of fruit. In a ratio of 1 to 9, we

divided the dataset into training and validation sets, with 90% of the data used for training and 10% for validating. The performance of the four deep learning models is comparable. While VGG targets the lowest error with the validation set, AlexNet displays the best MSE using the training set. Four well-known models, namely GoogLeNet, ResNet, AlexNet, and VGG-11, were independently trained on a convolutional neural network.

[14] concerns with the creation of a system that facilitates fruit identification. Here, the system explicitly relies on convolutional neural networks (CNNs). Convolutional layers, batch normalization layers, pooling layers, ReLU layers, and fully connected layers are all parts of the CNN. Convolutional layers are made up of kernels of various sizes. The photographs in the dataset utilized here fall into five main groups. Apple, Banana, Grapes, Litchi, and Mango are the five different categories. The dataset uses RGB pictures, which have the three color channels R, G, and B. The collection has 4,760 photos in total, spread over the five classes. The photos in the collection have already been cleaned up, scaled to 224x224x3, and cropped to exclude any extraneous data. 90% of the photos in the training dataset are trained, and 10% of the images in the validation dataset are validated. Stochastic gradient descent (SGD) with piecewise learning is the optimizer employed. There are 128 in each minibatch. The starting rate of learning is 0.01. The decline in learning rate is 0.02 percent. 40 epochs make up the learning rate drop phase. The training dataset's photos have been enhanced. The classification accuracy of the suggested CNN model in this study is 92.23%. But the restriction is that only five fruit groups were taken into account here for classification (They are apple, banana, grapes, litchi and mango).

## 3.1 Outcome of the Survey

- Most widely used dataset is “fruits fresh and rotten classification" , which mainly focused on classifying three classes of fruits resulting in the classification of a very small number of fruits.
- Traditional computer vision systems lack external defect detection techniques.
- Instead of utilizing the same weight value for all characteristics, adjusting the weight value may enhance performance.
- Images of fruits and vegetables are generally captured from one angle. Consider the images of fruits and vegetables captured from various angles to optimize system performance.
- Including images from various regions to ensure that the system is regional bias-free.
- A single fruit is used for grading, sorting, and disease identification. A generic system may also be created to grade or sort various fruits and vegetables at the same time.
- During the training phase, a monotonous background should be employed. If a non-monotonous background is employed, the model will learn undesired characteristics during the training phase. This may result in incorrect fruit classification.

# 4 Implementation

This section speaks about the structure and framework of the sequential model and implementation of conveyor belt along with servo mechanism for segregation .

The sequential model consists of 9 layers and is implemented using Google Collab . The components used for building the conveyor belt are wooden chassis , motors , servo's , belt and camera modules .

## 4.1 Software Implementation

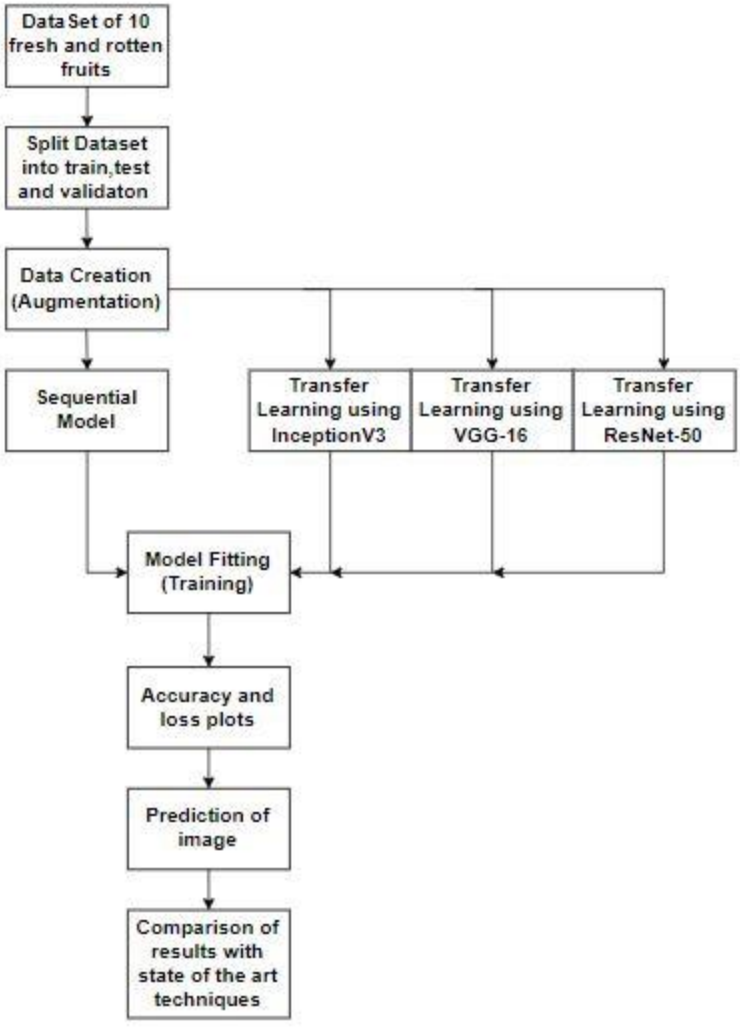



**Figure 4.1 :** Basic implementation flow chart

**Fig 4.1** shows the block diagram of the steps involved in the software implementation of the project . It consists of the creation of a custom dataset containing 10 fresh and rotten fruits . This is followed by splitting of the dataset into training , testing and validation categories .

Further steps involve data creation which include preprocessing and data augmentation . The prepared images in the dataset are fed to the sequential model and transfer learning model based on InceptionV3 and the models are trained .

Upon completion of training , accuracy and loss plots are obtained and the models are tested to predict an image . Results obtained from both the models are compared using state of the art techniques.

### 4.1.1 Data Fetch

The process of acquiring data required for training the sequential model . The custom dataset consists of images clicked using a mobile phone camera . The dataset consists of 2 folders namely train and test .There are 5000 training images and 1214 testing images making the total dataset of 6214 images . The training folder consists of 10 sub folders that are fresh and rotten folders for each class of the 5 classes of fruits . Each sub folder of fresh and rotten consists of 500 images making a total of 1000 images per class of fruits.

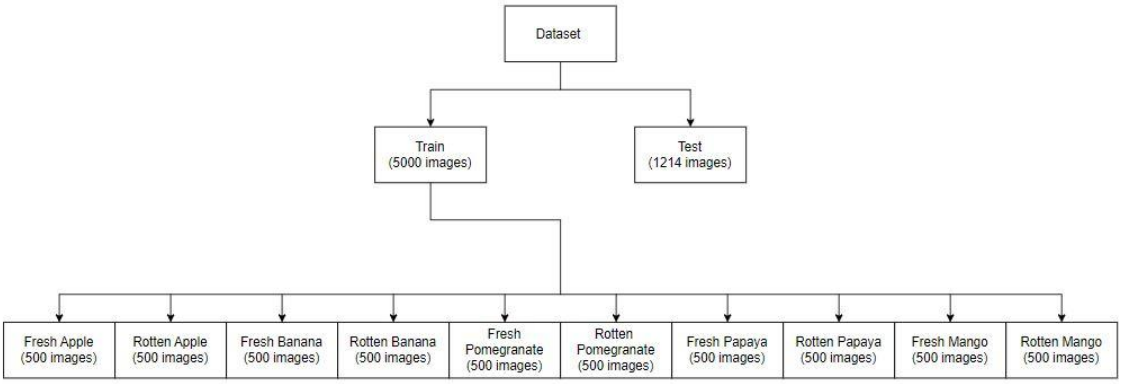



**Figure 4.2** Structure of the dataset

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

**Figure 4.3 : (a)** Fresh apple , **(b)** Rotten apple , **(c)** Fresh banana , **(d)** Rotten Banana , **(e)** Fresh mango , **(f)** Rotten mango , **(g)** Fresh papaya , **(h)** Rotten papaya , **(i)**Fresh pomegranate , **(j)** Rotten pomegranate

### 4.1.2 Preprocessing

Since the proposed methodology is based on convolutional neural networks, the preprocessing involved is very less when compared to other classification algorithms .

### 4.1.3 Data creation

Data creation consists of methods to augment the images used in the data. Data augmentation is considered to be a very important step as it helps to improve the performance by forming new and unique examples to train the datasets . The accuracy and performance of the model is highly dependent on the dataset which must be rich and sufficient .

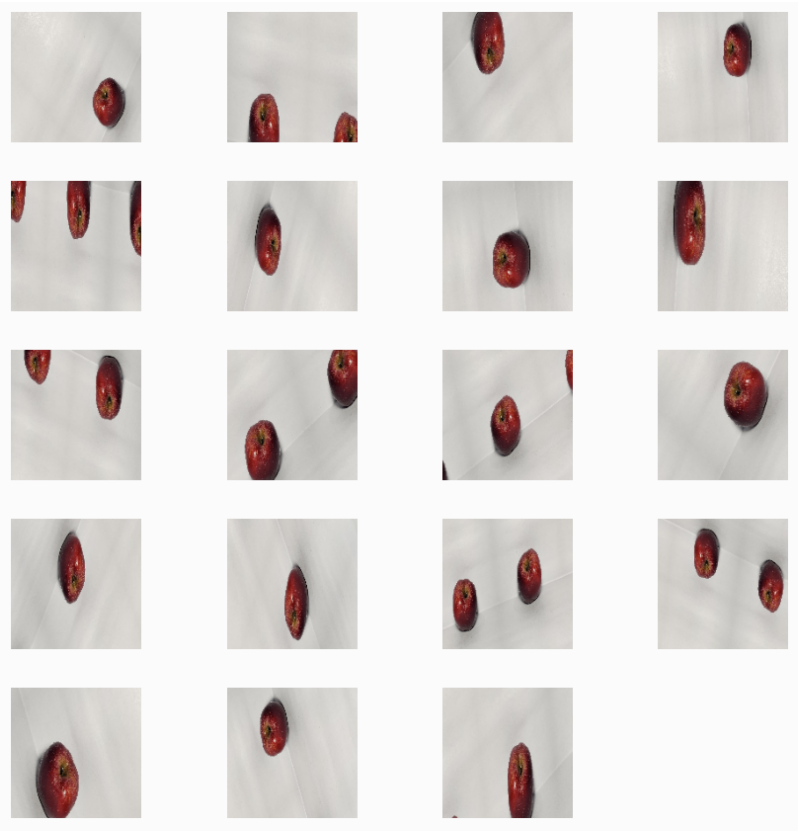


**Figure 4.4** : Augmentation and forming new and unique images for training

## 4.1.4 Convolutional Neural networks:

(ConvNet or CNN) is a deep learning algorithm that can recognise distinct characteristics and objects in an image and determine their relative importance by applying biases and learnable weights.

CNN requires fairly little preprocessing compared to other classification algorithms. The ConvNet's function is to transform the images into a more manageable format without sacrificing key elements needed for accurate predictions.

Through the use of pertinent filters, a ConvNet effectively captures the spatial and temporal dependencies in an image and because there are fewer factors to consider and the weights can be reused, the architecture provides a better fitting to the dataset. In other words, the network may be trained to better comprehend

the level of complexity in the image.

**Convolution Operation :**

Convolution Operation's goal is to take the input image's high-level characteristics, like edges, and extract them. There is no requirement that ConvNets have just one convolutional layer. Typically, low-level features like edges, color, gradient direction, etc. are captured by the first ConvLayer. With further layers, the architecture adjusts to the High-Level characteristics as well, giving us a network that understands the photos in the dataset as well as we do.

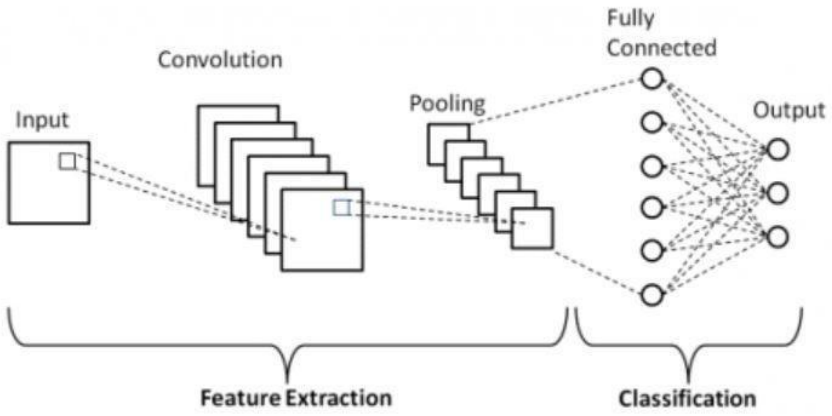



**Figure 4.5** : Basic architecture of a CNN

**Pooling Operation:**

Reduced spatial size of the Convolved Feature is a result of the Pooling layer. Through dimensionality reduction, the computational power required to process the data is reduced. Furthermore, it is useful for extracting dominant features that are rotational and positional invariant, allowing the model to be effectively trained.

Pooling consists of two types namely , max pooling and average pooling .

Max Pooling returns the maximum value from the Kernel-covered portion of the image. Average Pooling, on the other hand, returns the average of all the values from the portion of the image covered by the Kernel.

**Fully Connected layer :**

The Fully-Connected layer is usually a low-cost method of learning non-linear combinations of the high-level features represented by the convolution layer's output .

## 4.1.5 Sequential Model :

This subsection speaks about the structure and framework of the sequential model .

Sequential is the simplest way to build a model . Sequential models are basically the custom built models used to serve a specific task .

In a sequential model , each layer has exactly one input and one output and are placed one above the other to form the entire network .

The custom built sequential model for the quality detection of fruits consists of 9 layers .

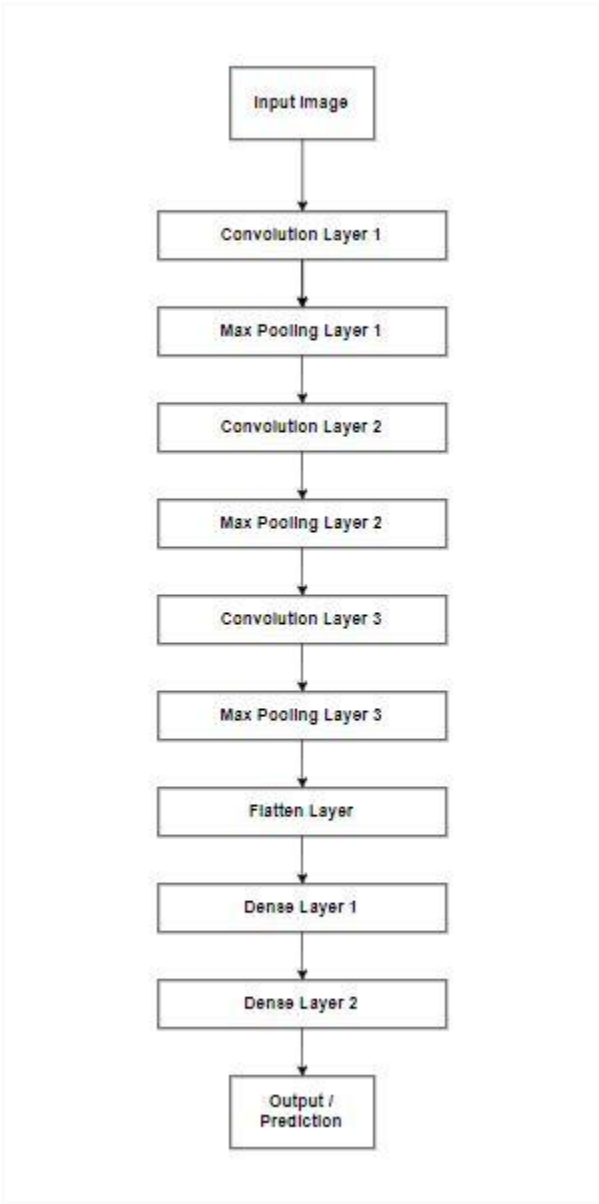



**Figure 4.6 :** Layers in Custom Sequential Model

As shown in **Fig 4.6 ,** the custom sequential model consists of 9 layers which include 3 convolution layers , 3 max pooling layers , 1 Flatten layer and 2 Dense layers .

The input image provided is of size 150 x 150 .

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 148, 148, 32)      896

 max_pooling2d_3 (MaxPooling  (None, 74, 74, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_5 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 36992)             0

 dense_2 (Dense)             (None, 512)               18940416

 dense_3 (Dense)             (None, 10)                5130

=================================================================
Total params: 19,038,794
Trainable params: 19,038,794
Non-trainable params: 0
```

**Figure 4.7:** Layer by layer Output Dimensions and Trainable Parameters

As shown in **Fig 4.7 ,** the output shape of the images after each operation can be seen . After each layer the number of trainable parameters are also calculated . Finally , the total trainable parameters are 19,038,794 with 0 non trainable parameters .

The first convolution layer and max pooling layer consists of 32 channels . The next convolution and max pooling layers have 64 channels and the final set of convolution and max pooling layers contain 128 channels .

After flattening , the size of the image goes to 36992x1 which is further reduced by dense layers to a 10 class classification .

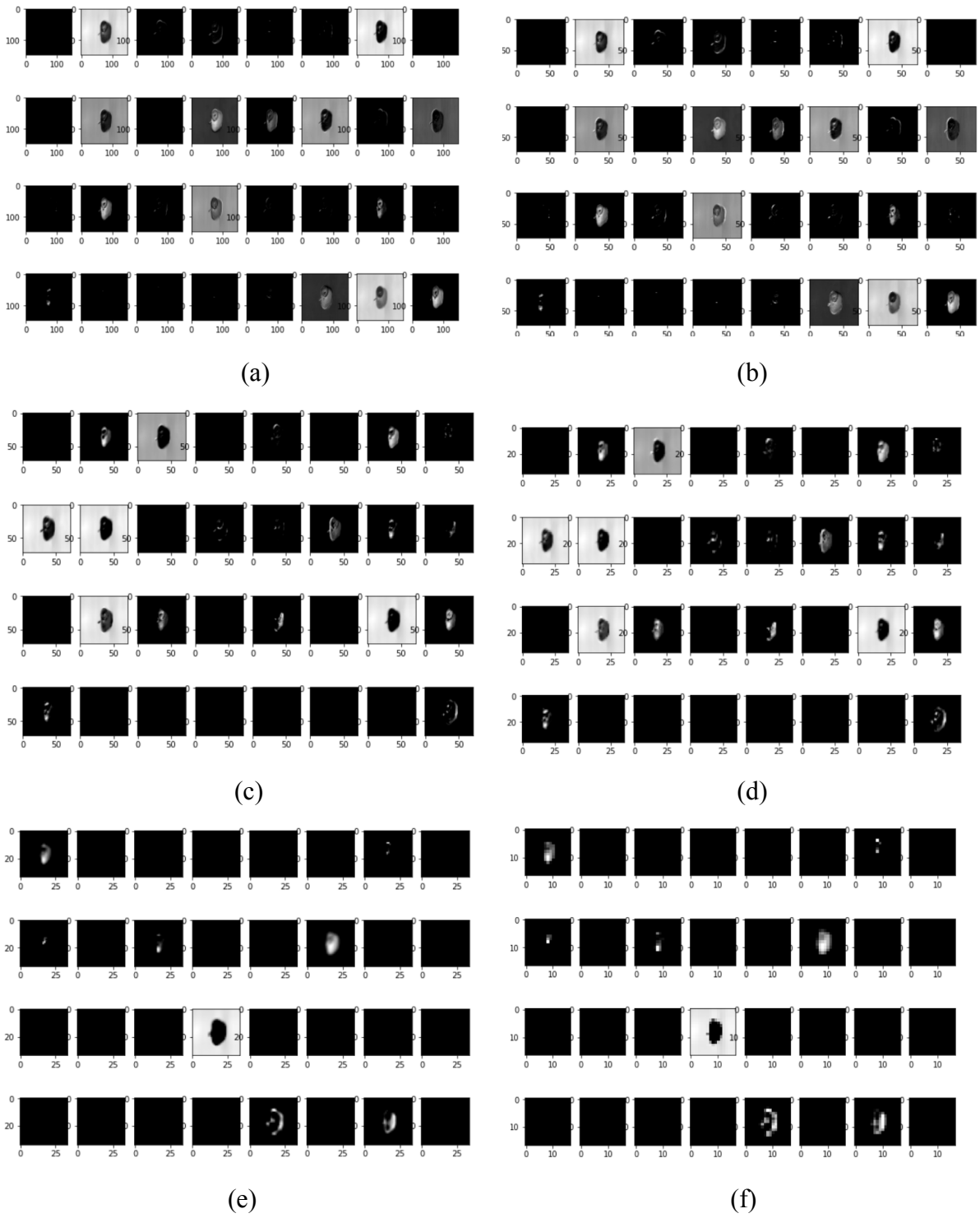


**Figure 4.8 :** Outputs of each layer on a single image , **(a)** Output of convolution 1 **(b)** Output of max pooling 1, **(c)** Output of convolution 2 , **(d)** Output of max pooling 2 **(e)** Output of convolution 3 , **(f)** Output of max pooling 3

### 4.1.6 Deep Learning Models:

This subsection deals with the different deep learning techniques used for fruit quality detection. One of the ways for computers to learn from data is deep learning. Each of the deep learning models used here have a main component i.e. the CNN'S.

Training pertaining to any CNN model requires a huge collection of data to make learning possible. Each neuron of each layer is interconnected, and it is responsible for the computations involved. This gives a computer algorithm or architecture the ability to detect patterns. CNNs can learn copious feature rendition for a broad spectrum of images. Consequently, deep learning is known to excel when it comes down to image classification especially when images belonging to multiple categories need to be classified.

The models used for classification are ResNet50, VGG-16 and InceptionV3. A dive into the structures of these models and its implementation for this problem is given in subsections that followed this discussion.

#### 4.1.6.1 ResNet50

The ResNet50 (shorthand for Residual Networks), with 50 indicating the number of layers in the CNNs, gained immense popularity in the field of computer vision because particularly the image classification in its ensemble secured the top position in the ILSVRC competition with a mere 3.57% error.

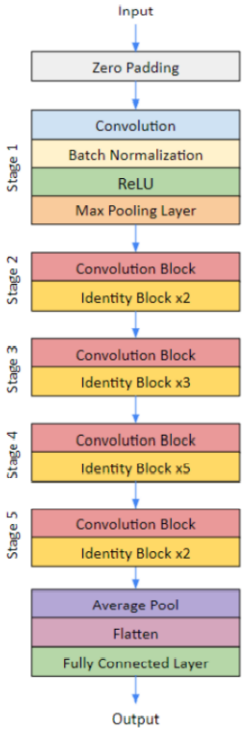



**Figure 4.9 :** Layers of ResNet50 Architecture

The layers that form the building block for ResNet50 architecture are as shown above. Batch Normalization is applied to normalize images within a batch. In stages from 2 to 5, there are multiple identity blocks in each. The fully connected layer finally would then connect all the networks to form the complete architecture. ResNet50 solves two problems, that are the degradation problem and vanishing gradient problem, accomplished by introducing skip connection in CNN architecture as shown below in the figure. The ResNet50 architecture as stated earlier comprises 50 CNN layers, of which 48 are convolution layers, and followed by max pooling layer and average pool layer with one each.

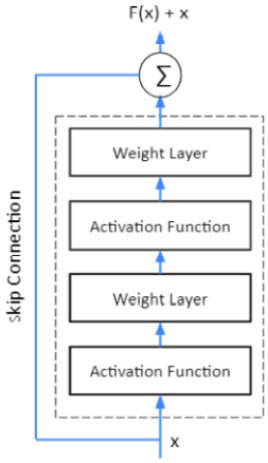



**Figure 4.10 :** Basic Building Block of ResNet50

The fundamental building blocks of the ResNet50 architecture would be the convolutional and the identity blocks with each having 3 convolutional layers, besides it can happen that the size of the input to the convolutional block is not equal to the size of the output of the block but it remains the same in the identity block.

ResNet50 has over 23 million trainable parameters. The main disadvantage is that the detection of errors becomes difficult for a deeper network. Additionally, the learning might be very inefficient if the network is too shallow.

## 4.1.6.2 VGG (Visual Geometry Group) - 16

The idea of the VGG framework was proposed by Karen Simonyan and Andrew Zisserman.

VGG-16 architecture comprises 13 convolutional layers, 2 fully connected layers and 1 SoftMax classifier as shown below. 3 x 3 convolutional layers have been put together one on top of each other for simplicity.

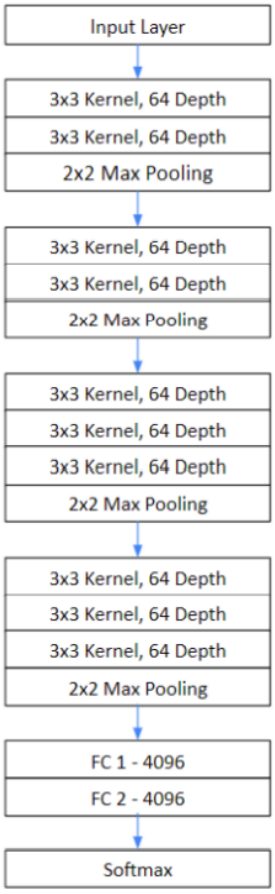



**Figure 4.11 :** VGG-16 Architecture

The figure above is a detailed illustration of the layers of VGG 16 architecture. The information in each layer represents the kernel size along with the filter size. Fully connected layer has its numbers indicating the filter size of that particular layer. VGG-16 as the name suggests comprises 16 layers that are convolutional layers and fully connected layers. It has only 3 x 3 convolutions, like AlexNet, but has a lot of filters. It can be trained on 4 GPUs for 2-3 weeks. The 16 layers have weights: 13 Convolutional layers, 5 Max Pooling layers, and 3 Dense layers which add up to 21 layers. 16 refers to weight layers that are known as learnable parameter layers.

The most distinguishing feature of VGG16 is that rather than a huge number of hyper parameters, the emphasis is on having convolution layers of 3x3 filters with stride 1 and employing the same padding and maxpool layer of the 2x2 filter with stride 2. The Max Pool and Convolutional layers are organized

regularly throughout the design. Conv-1-64 filters, Conv-2-28 filters, Conv-3-256 filters, Conv-4 and Conv-5 have 512 filters. A convolutional layer stack is followed by three FCs with 4096 channels each. The third uses 1000-way ILSVRC classification, therefore it has 1000 channels (one for each class). The soft-max layer would be the last layer.

### 4.1.6.3 InceptionV3

The model consists of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is employed extensively throughout the model and is applied to activation inputs. Softmax is used to compute loss.

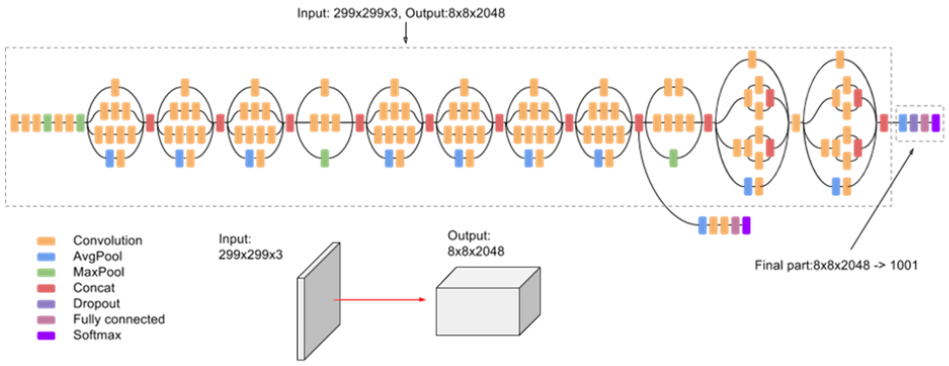



**Figure 4.12 :** Basic Architecture of InceptionV3

The architecture of an Inception v3 network is built in layers, as discussed below. It incorporates Factorized Convolutions, which aid in lowering computational efficiency as the number of parameters in a network decreases. It also monitors the network efficiency. Smaller convolutions replace larger convolutions, enabling faster training. Asymmetric convolutions are employed. During training, an auxiliary classifier (a tiny CNN) is added

between layers, and the loss it incurs is added to the main network loss. An auxiliary classifier serves as a regularizer in Inception v3. Finally, grid size reduction is done by pooling operations. All these concepts are incorporated into the final architecture. Inception v3 can achieve the lowest error rates when integrated with an auxiliary classifier, factorization of convolutions, RMSProp, and Label Smoothing.

## 4.1.7 Model creation and Training

This subsection deals with implementation of the custom sequential model along with using Resnet-50 , Vggnet and Inception V3 deep learning models for the quality detection of fruits and comparison of the results obtained with the sequential model .

The models are built on Google Collab using Keras 2.10.0 environment with a tensorflow 2.10.0 backend .

Each of the models , that is , the sequential model , Resnet-50 model , Vggnet model and inceptionV3 model are trained for a total of 50 epochs with a step size of 29 .

**4.1.7.1 Callbacks:**

Two callbacks are implemented, namely, ReduceLROnPlateau and a ModelCheckPoint callback .

The ReduceLROnPlateau basically reduces the learning rate when a metric has stopped improving . The metric being monitored is validation accuracy with a patience of 7 epochs and a reduce factor of 0.2 . This means that the learning rate will be reduced by a factor of 0.2 if the metric being monitored does not change over a period of 7 epochs .

The ModelCheckPoint callback is primarily used to save the Keras model or model weights on a regular basis. It is used in conjunction with training to save a model or weights (in a checkpoint file) at regular intervals so that the model or weights can be loaded later to continue training from the saved state . This callback too monitors a particular metric like validation accuracy and if there is no change in the monitored metrics over a patience level of 7 epochs , then the model training stops and a .h5 file is saved at that particular epoch .

**4.1.7.2 Activations :**

**1) RELU :**

After each layer of convolution and max pooling RELU activations are used .

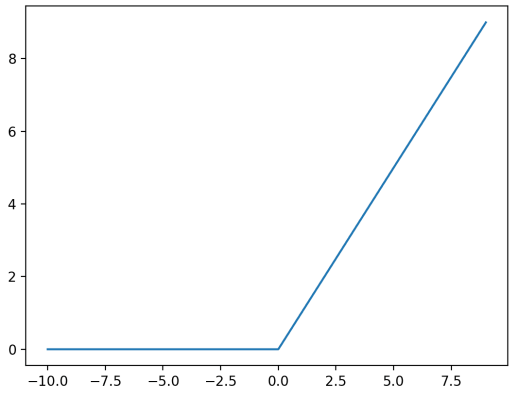
RELU stands for rectified linear activation function , is a piecewise linear function that outputs the input directly if it is positive; otherwise, it outputs zero.

It has become the default activation function for many types of neural networks because it is easier to train and frequently results in better performance .
Mathematically RELU can be represented as :

$$f(x) = \max(0, x)$$

$$f(x) = \begin{cases} 0, if\ x < 0 \\ x, if\ x \geq 0 \end{cases}$$

(a)




(b)

**Figure 4.13 : (a)** Mathematical Expression of Relu , (b) Graph of Relu Activation

**2) Softmax :**

Since the classification is of the multi class type , softmax activation must be used. Any prediction given by the model is in the form of numbers or logits .
Softmax activation converts numbers into probabilities .

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)}$$

where :

y = input vector
y(i) = i'th element of the input vector
exp(y(i)) = standard exponential function applied on y(i)
n = Number of classes

$$\sum_{j=1}^{n} \exp(y_j)$$

A normalization term. It ensures that the values of output vector $S(y)_i$ sums to 1 for $i$-th class and each of them and each of them is in the range 0 and 1 which makes up a valid probability distribution.

**Figure 4.14 :** Mathematical expression for softmax

#### 4.1.7.3 Loss Functions :

Loss functions are basically metrics that will help to evaluate the model based on the dataset . A way to assess how well your algorithm models your dataset is with the Loss function. It is an equation that depends on the machine learning algorithm's parameters.

There are many different types of loss functions based on the type of algorithm being used . For the purpose of classification , there exist 3 types of loss functions , namely

- binary cross entropy
- categorical cross entropy
- sparse categorical cross entropy

Binary cross entropy is a loss function that is used for a 2 class classification (binary classification).

Both categorical cross entropy and sparse categorical cross entropy are used for Multi class classification .

Choosing the right loss function is a very difficult task . All the models in this project make use of categorical cross entropy for calculating the loss .

**4.1.7.4 Optimizers :**

An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improving the accuracy.

Choosing the right weights for the model is a difficult task because a deep learning model typically has millions of parameters . It emphasizes the importance of selecting an appropriate optimization algorithm .

There are many different types of optimizers , namely

- Gradient Descent
- Stochastic Gradient Descent
- Stochastic Gradient descent with momentum
- Mini-Batch Gradient Descent
- Adagrad
- RMSProp
- AdaDelta
- Adam

Adam optimizer is used as the optimization algorithm for the models being trained as it provides several advantages over the others .

## 4.1.8 Codes for All models Implementation

- Sequential model

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    # tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

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

model.summary()
```

- ResNet-50

```
from tensorflow.keras.applications import Xception, VGG16, InceptionV3 , ResNet50
from tensorflow.keras import layers
from tensorflow.keras import Model
import pandas as pd

######################## init pre_trained_model ################################

# pre_trained_model = InceptionV3(input_shape=(150,150,3),
#                                 include_top=False)

pre_trained_model = ResNet50(input_shape=(150,150,3),
                           include_top=False)

# pre_trained_model = Xception(input_shape=(150,150,3),
#                              include_top=False)

# pre_trained_model.layers.pop()

######################## init pre_trained_model ################################

############################## freeze layer ####################################

for layer in pre_trained_model.layers:
    layer.trainable = False

# for layer in pre_trained_model.layers[:-4]:
#     layer.trainable = False

############################## freeze layer ####################################

############################## Flatten Layer ###################################

# last_output = pre_trained_model.layers[-1].output
# x = layers.Flatten()(last_output)


############################## Flatten Layer ###################################

######################### Fully Connected Layer ################################
x = layers.Flatten()(pre_trained_model.output)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dense(10, activation='softmax')(x)

######################### Fully Connected Layer ################################

model = Model(pre_trained_model.input, x)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

pd.set_option('max_colwidth', None)
layers = [(layer, layer.name, layer.trainable) for layer in pre_trained_model.layers]
pd.DataFrame(layers, columns=['Layer Type', 'Layer Name', 'Layer Trainable'])
```

- VggNet -16

```
from tensorflow.keras.applications import Xception, VGG16, InceptionV3
from tensorflow.keras import layers
from tensorflow.keras import Model
import pandas as pd

######################## init pre_trained_model ################################

# pre_trained_model = InceptionV3(input_shape=(150,150,3),
#                                 include_top=False)

pre_trained_model = VGG16(input_shape=(150,150,3),
                          include_top=False)

# pre_trained_model = Xception(input_shape=(150,150,3),
#                              include_top=False)

# pre_trained_model.layers.pop()

######################## init pre_trained_model ################################

############################## freeze layer ####################################

for layer in pre_trained_model.layers:
    layer.trainable = False

# for layer in pre_trained_model.layers[:-4]:
#     layer.trainable = False

############################## freeze layer ####################################

############################## Flatten Layer ###################################

# last_output = pre_trained_model.layers[-1].output
# x = layers.Flatten()(last_output)

x = layers.Flatten()(pre_trained_model.output)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dense(10, activation='softmax')(x)

######################### Fully Connected Layer ################################

model = Model(pre_trained_model.input, x)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

pd.set_option('max_colwidth', None)
layers = [(layer, layer.name, layer.trainable) for layer in pre_trained_model.layers]
pd.DataFrame(layers, columns=['Layer Type', 'Layer Name', 'Layer Trainable'])
```

- Inception V3

```
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras import layers
from tensorflow.keras import Model
import pandas as pd

pre_trained_model = InceptionV3(input_shape=(150,150,3),
                                include_top=False)

for layer in pre_trained_model.layers:
    layer.trainable = False

x = layers.Flatten()(pre_trained_model.output)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dense(10, activation='softmax')(x)

model = Model(pre_trained_model.input, x)
model.compile(optimizer='adam', #RMSprop(lr=0.0001), adam
              loss='categorical_crossentropy',
              metrics=['accuracy'])

pd.set_option('max_colwidth', None)
layers = [(layer, layer.name, layer.trainable) for layer in pre_trained_model.layers]
pd.DataFrame(layers, columns=['Layer Type', 'Layer Name', 'Layer Trainable'])
```

## 4.2 Hardware Implementation

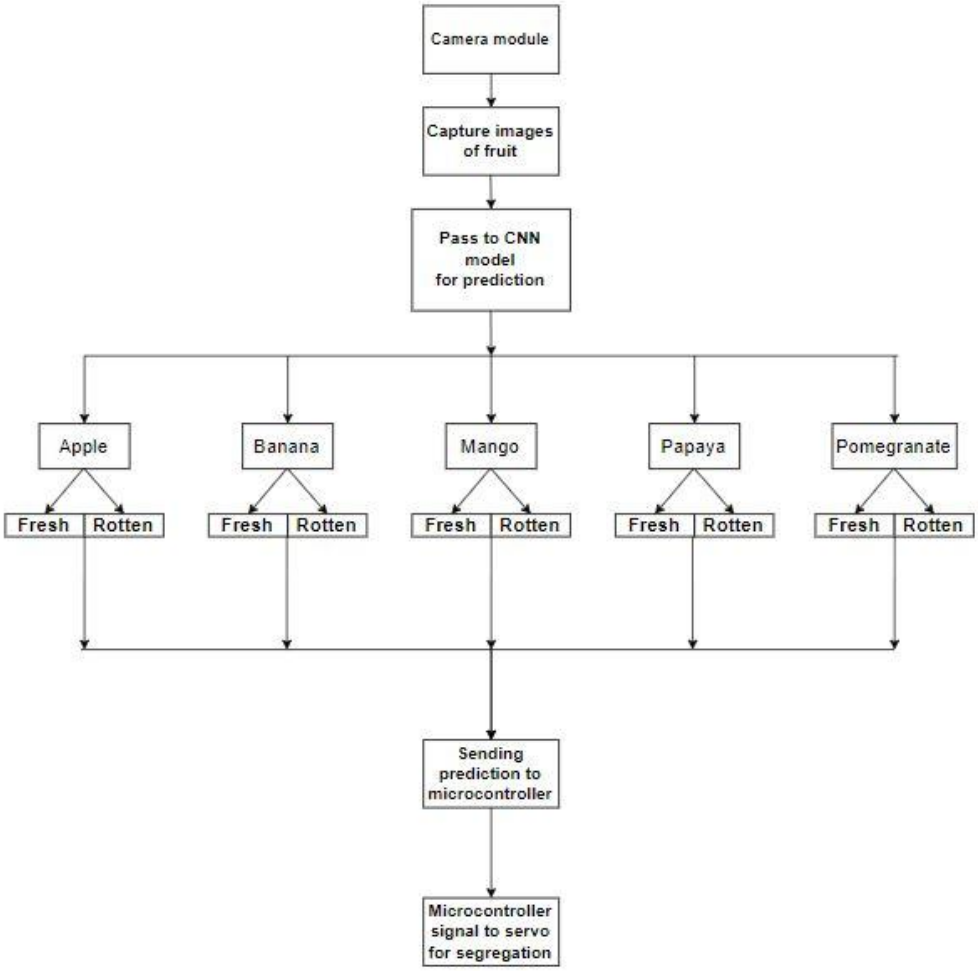



**Figure 4.15 :** Basic implementation flow chart

**Figure 4.15** shows the basic block diagram for hardware implementation of the prototype . The prototype design consists of a conveyor belt run by a set motors . The fruits to be classified as fresh or rotten have to be placed over the conveyor belt . The camera module placed will capture the image of the fruit and feed these images to the Sequential model for preprocessing and data augmentation .

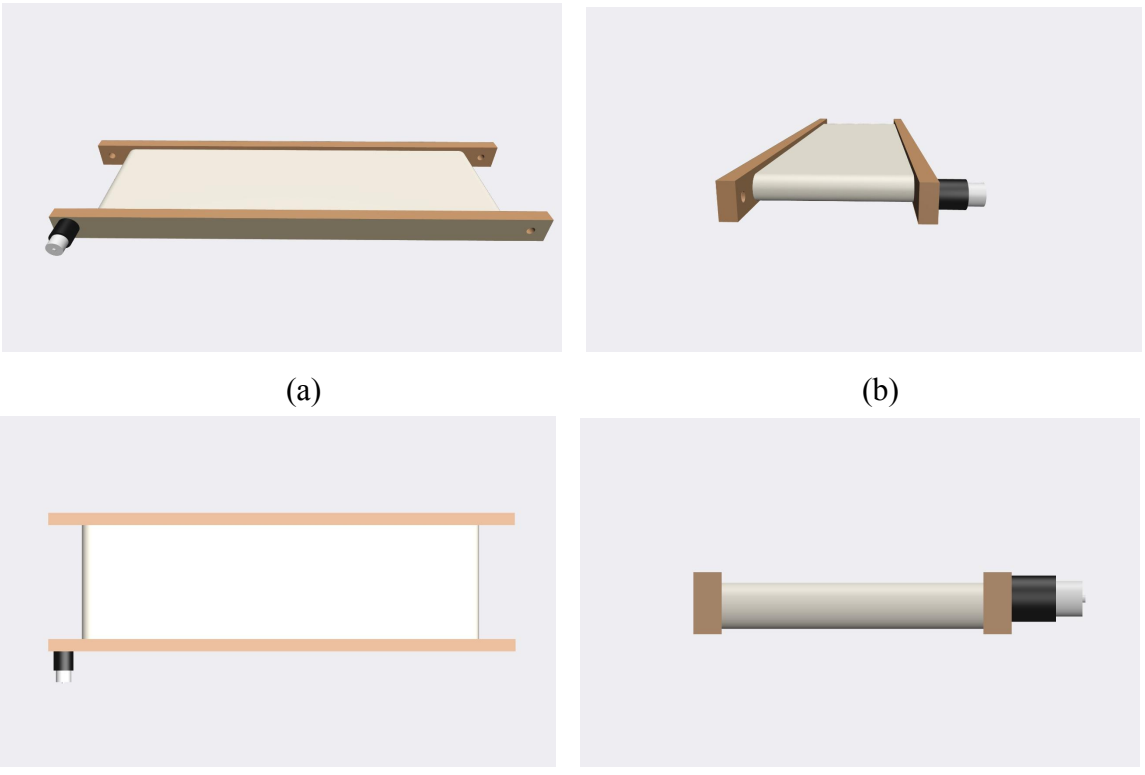
OpenCv is used to capture the images and a mobile phone camera is being used as the camera module .

The Sequential model will predict to which class of fruits the images belong to and further classify them as fresh or rotten . This prediction obtained is then sent to a microcontroller (Arduino uno) through serial communication using the python library known as pyserial .

The servo is attached with a plastic bar which sweeps across the conveyor belt and acts as a barrier for rotten fruits .

Once serial communication is established , based on the prediction , the conveyor belt runs and the position of the servo changes . If the fruit is fresh , then the servo stays at its original position which is 90 degrees . If the fruit is rotten , then the servo changes its position to 180 degrees along with which the plastic bar acts as a barrier to the rotten fruit knocking it out of the conveyor belt .

## 4.2.1 Hardware architecture of Conveyor belt



(a) (b)

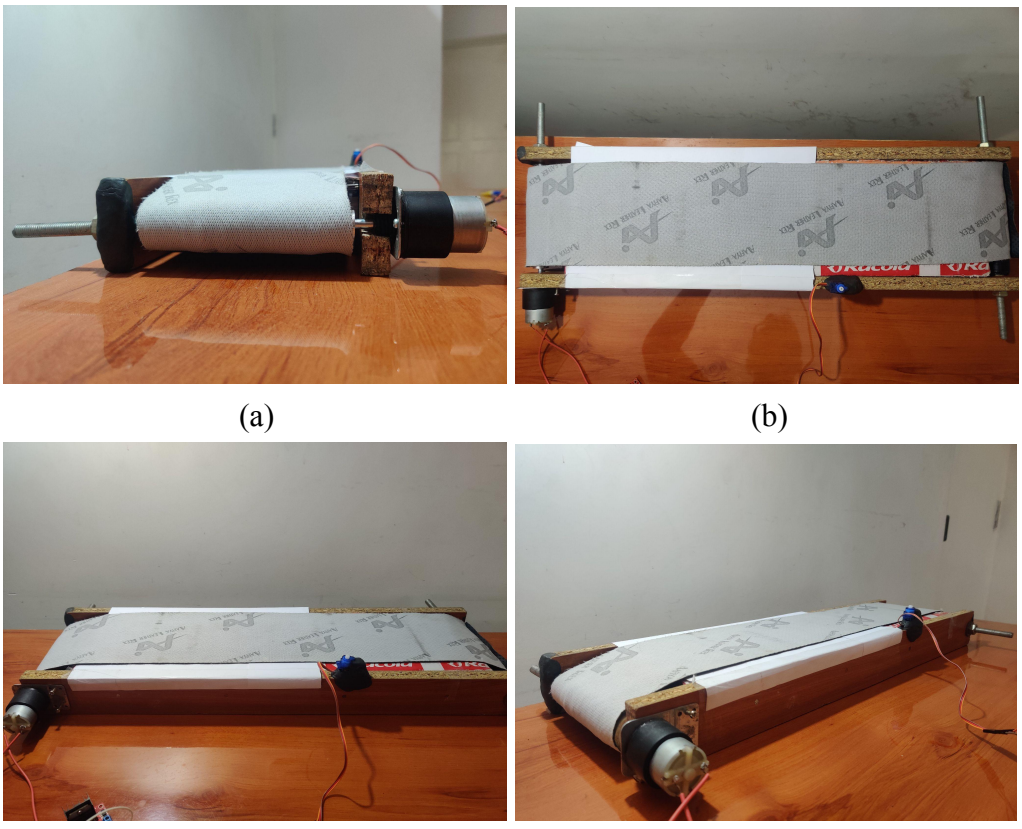
(c) (d)

**Figure 4.16 :** Different views of the proposed conveyor belt **(a)** front view , **(b)** side view , **(c)** topview , **(d)** side view

The conveyor belt consists of two long wooden pieces of length 65 cm , width 2 cm and height 5cms (65x2x5) . Both these wooden pieces are placed parallelly as shown in **figure 4.16** and are attached together by another wooden piece of dimensions 50 cm in length , 15 cm in width and 2 cm height (50x15x2) .

Towards both the ends , 2 rollers made from pvc are attached and are wound with rubber to increase traction .

The gear motor is attached near one end to one of the rollers using ball bearings and a simple axle . The motor used is a 12V 60 rpm gear motor and has a pulling capacity up to 1kg .



(a) (b)

(c) (d)

**Figure 4.17 :** Implementation of conveyor belt , **(a)** front view , **(b)** side view , **(c)** topview , **(d)** side view

## 4.2.2 Jupyter Notebook implementation

All the models discussed in the previous section are implemented on google colab and thus controlling arduino serially through google colab is not possible .

Hence the custom sequential model is loaded locally using jupyter notebook .

Through the jupyter notebook , serial communication is established .

```
import serial                              # import serial library
arduino = serial.Serial('COM5', 9600)      # create serial object named arduino
# print arduino echo to console
```

**Figure 4.18 :** Serial Communication Establishment with arduino using PySerial

The custom sequential model is saved as a .h5 file on the local device storage and is loaded into Jupyter notebook . Once the model is loaded , it is compiled and ready for prediction .

The image for prediction is captured using OpenCv library with the help of a mobile phone camera being used as a camera module .

## 4.2.3 Arduino and Servo mechanism

This subsection deals with the structure of the servo mechanism and program to control the servo through arduino using serial communication and pyserial python libraries .

PySerial library is imported into the Jupyter notebook and serial communication is established as shown in **figure 4.18** . Once this is done the fruit image is captured using OpenCv library on a mobile phone camera and is passed on to the sequential model for prediction .

This prediction is stored as a label in an array named predicted_result .

If the stored result is fresh , then we write command 0 to the arduino which executes the fresh part of the arduino code which is to retain the servo at its original position and only run the conveyor belt .

If the stored result is rotten , then we write command 1 to the arduino which executes the rotten part of the arduino code which is to sweep the servo along

with the plastic barrier across the conveyor belt and also run the conveyor belt .

```
if predict_result[0] in ["Fresh Banana","Fresh Apple","Fresh Mango","Fresh papaya","Fresh pomogrenate"]:
    print("Fresh")
    command ="0"

else:
    print("Rotten")
    command = "1"      # query servo position
  #  arduino.write(command)

arduino.write(str.encode(command))
reachedPos = str(arduino.readline())
```

**Figure 4.19 :** Writing Command 0 or 1 to arduino

Thus if the fruit is rotten then the plastic barrier across the conveyor belt will not let the fruit to pass and will knock the fruit out of the belt .

If the fruit is fresh then the servo remains at its original position and only the conveyor belt moves .

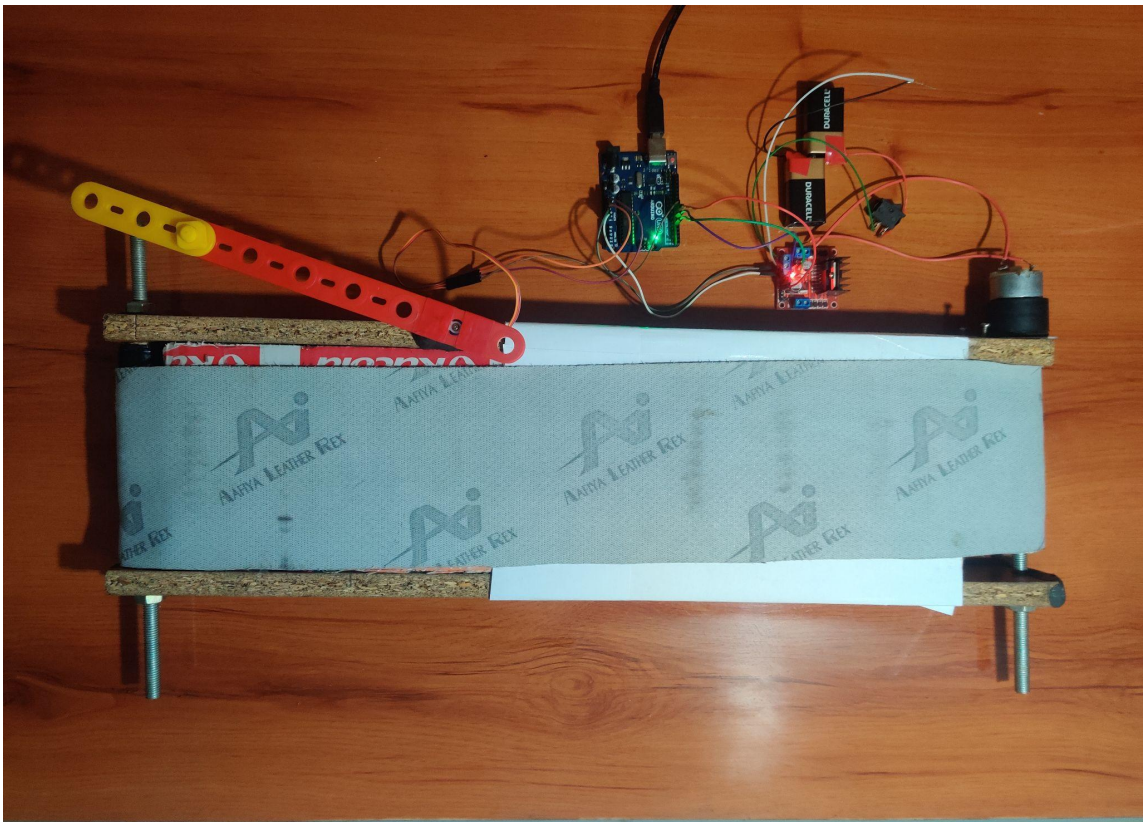


**Figure 4.20 :** Servo at its Original position (Fresh fruit condition)

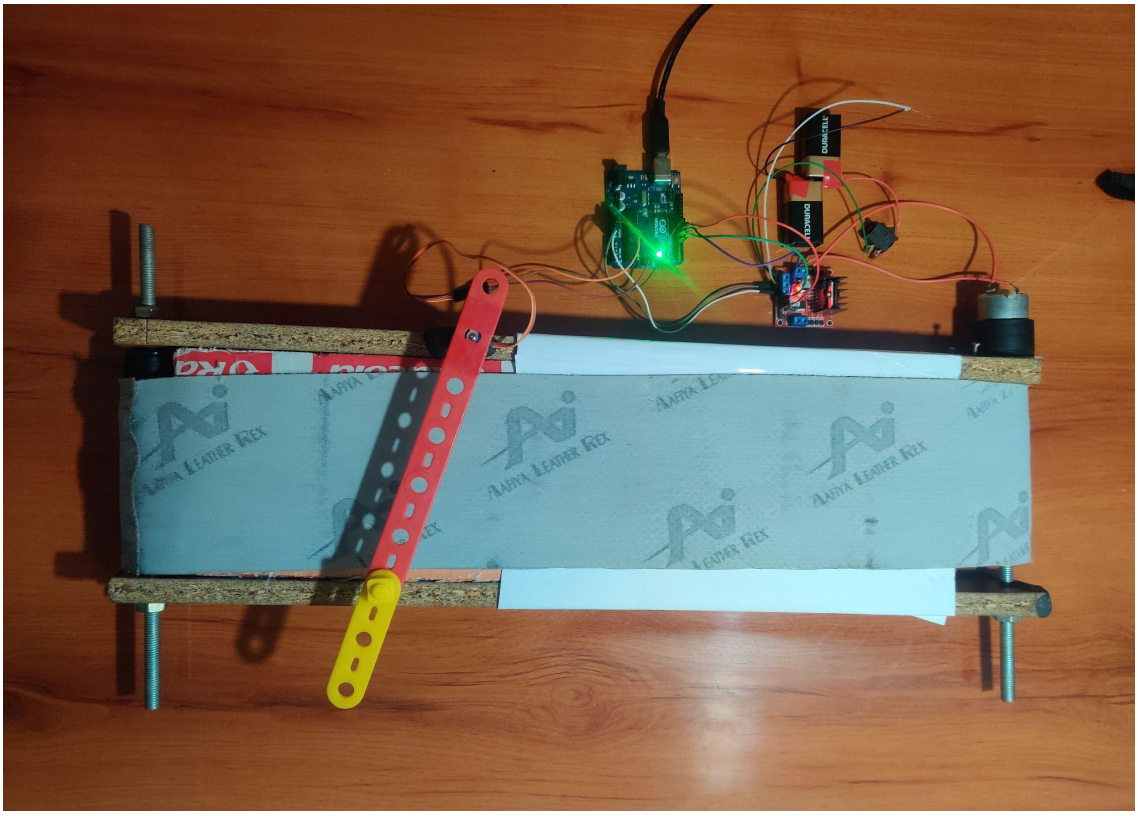



**Figure 4.21 :** Servo at its closed position (Rotten fruit condition)

```ino
#include <Servo.h>
Servo myservo;
String inByte;
int pos;
int In1=7;
int In2=8;
int ENA=5;
int SPEED=210;

void setup() {

  myservo.attach(9);
  Serial.begin(9600);
  pinMode(In1,OUTPUT);
  pinMode(In2,OUTPUT);
  pinMode(ENA,OUTPUT);
  analogWrite(ENA,SPEED);
}

void loop()
{
  if(Serial.available())  // if data available in serial port
    { inByte = Serial.readStringUntil('\n'); // read data until newline
    //Serial.println(inByte);
    if(inByte == "1"){
    digitalWrite(In1,HIGH);
    digitalWrite(In2,LOW);
    pos = 180 ;  // change datatype from string to integer
    myservo.write(pos);
    }
  else{
    digitalWrite(In1,HIGH);
    digitalWrite(In2,LOW);
     pos =90;
     myservo.write(pos);
     // move servo
    //Serial.print("Servo in position: ");
    //Serial.println(inByte);
  }
    }
```

**Figure 4.22 :** Arduino code to receive serial write and control servo

# 5 Results

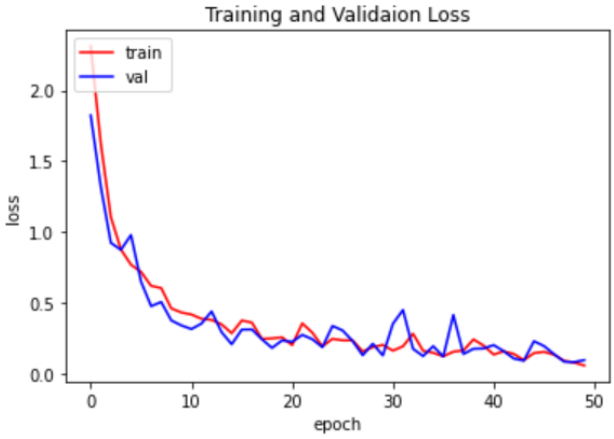
This section consists of the various results obtained from the evaluation of each of the models discussed in the earlier sections . For each of the models , the training and validation accuracy against the number of epochs has been plotted .

The second graph plots the training and validation loss against the number of epochs

Each model is trained for 50 epochs with a step size of 29 . Other metrics used to assess the sequential models potential are Precision , Recall , F1 Score and confusion matrix .

## 5.1 Sequential Model


(a)




(b)

**Figure 5.1 : (a)** Training and Validation Accuracy , **(b)** Training and Validation Loss

| Validation Accuracy | Validation loss | Test Accuracy | Test Loss |
|---|---|---|---|
| 0.9604 | 0.0936 | 0.9703 | 0.0680 |

**Figure 5.2 :** Performance parameters for the sequential model

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.55 | 0.98 | 0.71 | 43 |
| 1 | 1.00 | 1.00 | 1.00 | 160 |
| 2 | 1.00 | 1.00 | 1.00 | 100 |
| 3 | 1.00 | 1.00 | 1.00 | 95 |
| 4 | 0.99 | 0.87 | 0.92 | 120 |
| 5 | 1.00 | 0.84 | 0.91 | 116 |
| 6 | 1.00 | 1.00 | 1.00 | 170 |
| 7 | 1.00 | 1.00 | 1.00 | 120 |
| 8 | 1.00 | 1.00 | 1.00 | 140 |
| 9 | 0.99 | 1.00 | 1.00 | 150 |
| accuracy | | | 0.97 | 1214 |
| macro avg | 0.95 | 0.97 | 0.95 | 1214 |
| weighted avg | 0.98 | 0.97 | 0.97 | 1214 |

**Figure 5.3 :** Classification Report

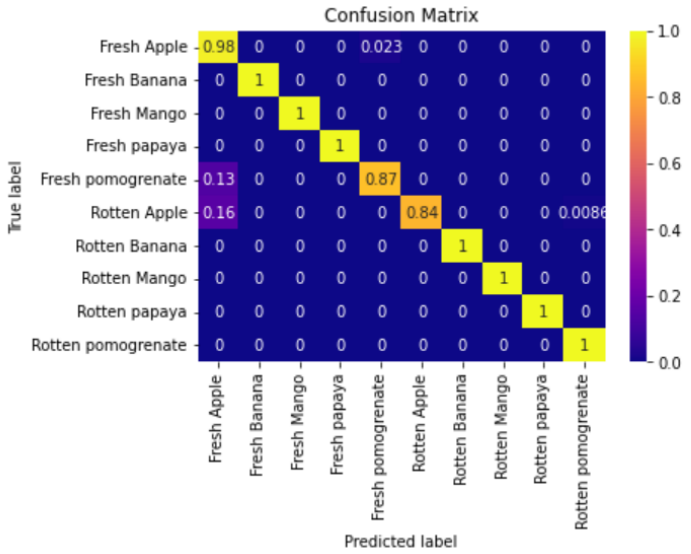
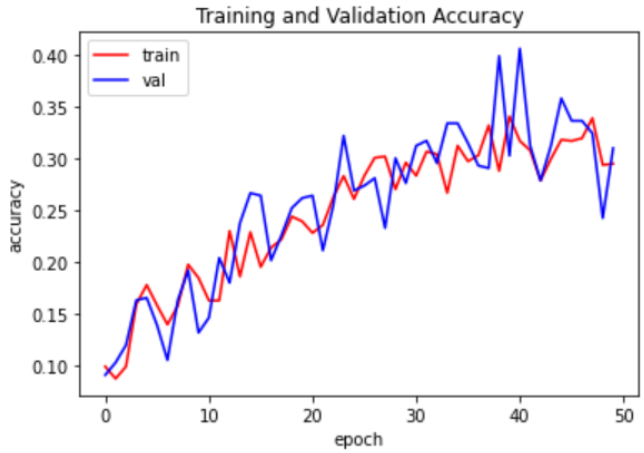



**Figure 5.4 :** Confusion Matrix


**Figure 5.5 :** Predictions

## 5.2 ResNet50




(a)


(b)

**Figure 5.6 : (a)** Training and Validation Accuracy , **(b)** Training and Validation Loss

| Validation Accuracy | Validation loss | Test Accuracy | Test Loss |
|---|---|---|---|
| 0.3101 | 1.8080 | 0.4761 | 1.4907 |

**Figure 5.7 :** Performance parameters for ResNet50

## 5.3 VGG-16


(a)


(b)

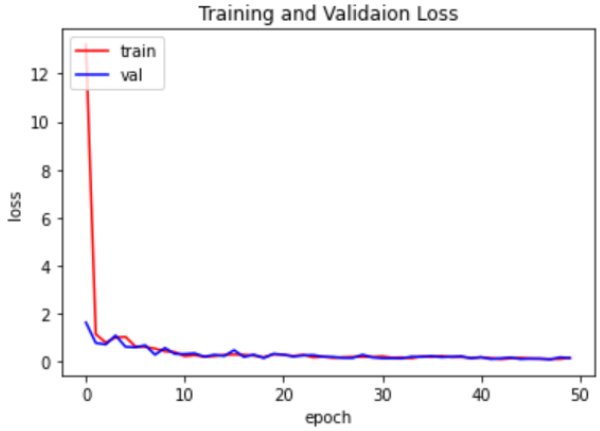
**Figure 5.8 : (a)** Training and Validation Accuracy , **(b)** Training and Validation Loss

| Validation Accuracy | Validation loss | Test Accuracy | Test Loss |
|---|---|---|---|
| 0.9567 | 0.1799 | 0.9182 | 0.1992 |

**Figure 5.9 :** Performance parameters for VGG-16

## 5.4 InceptionV3


(a)




(b)

**Figure 5.10 : (a)** Training and Validation Accuracy , **(b)** Training and Validation Loss

| Validation Accuracy | Validation loss | Test Accuracy | Test Loss |
|---|---|---|---|
| 0.9567 | 0.1378 | 0.9449 | 0.1376 |

**Figure 5.11 :** Performance parameters for InceptionV3

## 5.5 Comparative Analysis

| On scale (0-1) | Sequential Model | Inception V3 | VGG-16 | ResNet-50 |
|---|---|---|---|---|
| Val_Accuracy | 0.9604 | 0.9567 | 0.9567 | 0.3101 |
| Test_accuracy | 0.9703 | 0.9449 | 0.9182 | 0.4761 |

**Figure 5.12 :** Comparative analysis of the sequential model against ResNet50, VGG-16 and InceptionV3

# 6 Industrial Visit (Namdhari Pvt Ltd)

In this industry manual inspection of fruits is done. Fruits are received and sorted based on their quality and instantly dispatched to the respective stores. The facility also has a cold storage area for exotic fruits and other fruits that are further dispatched when needed.

(a)

(b)

(c)

(d)

(e)

**Figure 6.1 :** Industrial Visit , **(a)** facility , **(b)** manual segregation of fruits , **(c)** inside the facility , **(d)** cold storage , **(e)** cold storage

# 7 Conclusion and Future Scope

## 7.1 Conclusion

A sequential model based on a convolutional neural network is designed. Three convolution layers, three max pooling layers, a flatten layer, and two dense layers comprise the model. It offers the output that was obtained after each convolutional layer and maximum pooling. A prototype design that consists of a conveyor belt system run by a DC motor, a servo motor and a camera module to capture images and frameworks to trigger the model is built. The model is evaluated in comparison to Inception V3, ResNet-50, and VGG-16.

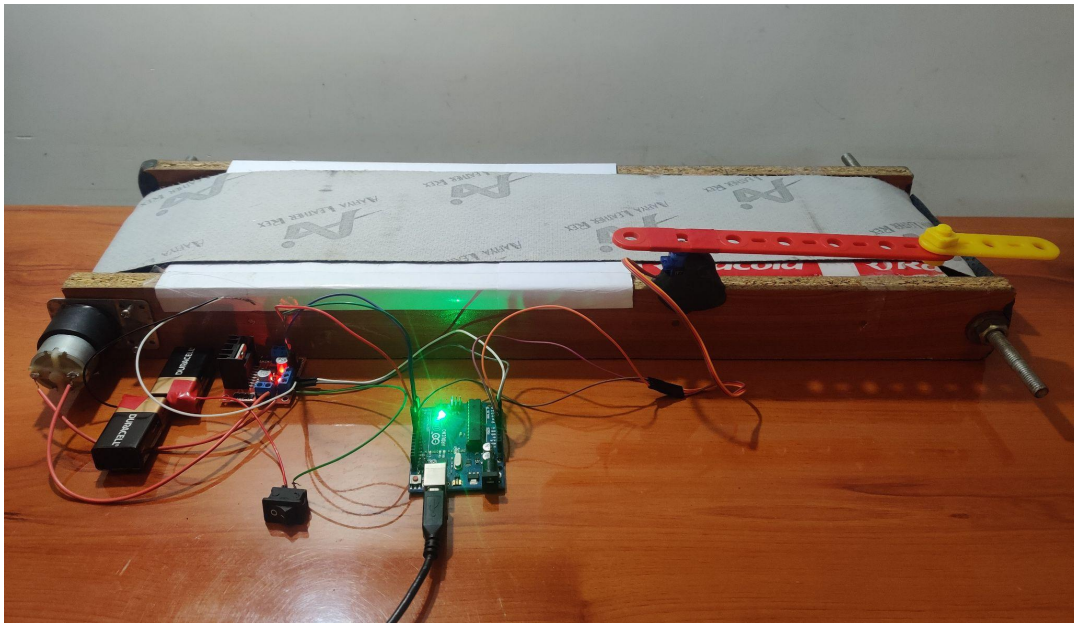


**Figure 7.1: (a)** Conveyor Belt System

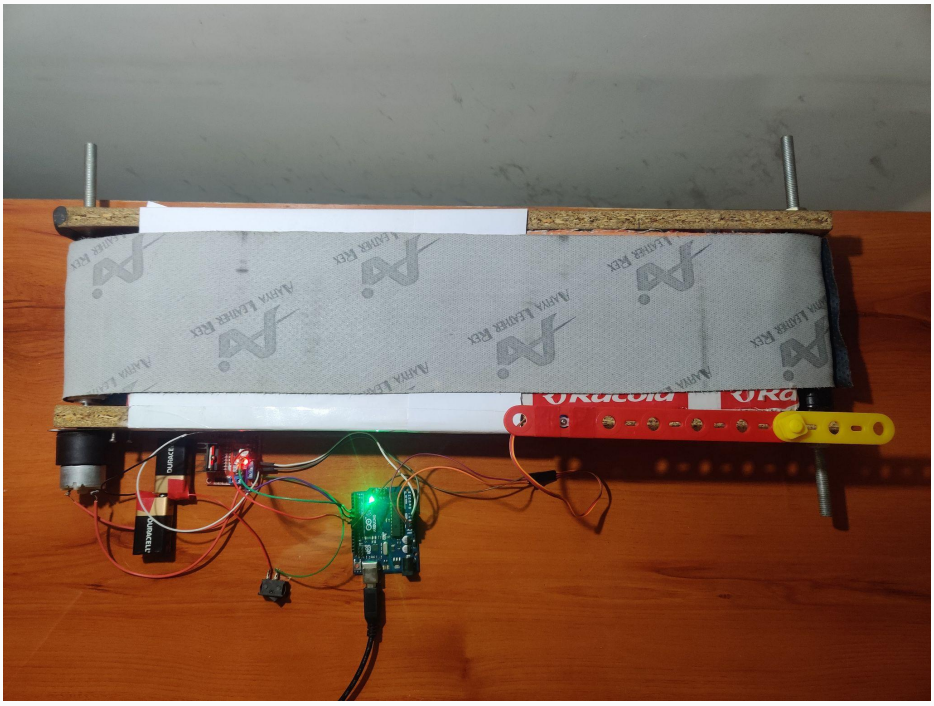


**Figure 7.1: (b)** Final Conveyor Belt System

In this way the segregation mechanism of fresh and rotten fruits in the industries becomes more accurate and less time consuming. It also reduces the man-power which was needed till this point of time to perform these functions.

## 7.2 Future Scope

As for the future scope, we aim to improve the working of the prototype by decreasing human intervention in its working. At present our prototype needs assistance from the operator/user to run it repeatedly for each subsequent fruit. But, our goal is to modify the prototype in such a way that it does not require any human assistance for rerunning the system and the camera module each time a fruit needs to be classified. Hence, it will be able to run independently and continuously. We also aim at including more cameras at multiple angles so that the images will cover the fruit completely from all angles. Thus,

improving the recognition accuracy of the rotten and fresh fruits. We also aim to improve the working of the model by using better segregation methods and also to increase the scope of the model, so that it can even segregate rotten and fresh vegetables. Presently, the model has been trained to detect and classify five fruits. The challenge is to train the model to classify a greater number of fruits into fresh and rotten. The sequential model built can be implemented for automation in fruit processing industries which are currently largely dependent on man power.

# 8 References

[1] Yuhang Fu, Minh Nguyen, Wei Qi Yan. “Grading Methods for Fruit Freshness Based on Deep Learning”, 2022, https://doi.org/10.1007/s42979-022-01152-7

[2] Ukwuoma, C.C., Zhiguang, Q., Bin Heyat, M.B., Ali, L., Almaspoor, Z. and Monday, H.N., 2022. Recent advancements in fruit detection and classification using deep learning techniques. Mathematical Problems in Engineering, 2022.

[3] Jaeyong Kang, Jeonghwan Gwak. (2021). “Ensemble of multi-task deep convolutional neural networks using transfer learning for fruit freshness classification”.

[4] C. C. Foong, G. K. Meng and L. L. Tze, "Convolutional Neural Network based Rotten Fruit Detection using ResNet50," 2021 IEEE 12th Control and System Graduate Research Colloquium (ICSGRC), 2021, pp. 75-80, doi: 10.1109/ICSGRC53186.2021.9515280.

[5] Md Sohel Miah, Tayeeba Tasnuva, Mirajul Islam, “An Advanced Method of Identification Fresh and Rotten Fruits using Different Convolutional Neural Networks”, 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 978-1-7281-8595-8/21/$31.00, 2021 IEEE, DOI: 10.1109/ICCCNT51525.2021.9580117.

[6] Sumitra Nuanmeesri, Lap Poomhiran, Kunalai Ploydanai (2021). Improving the Prediction of Rotten Fruit Using Convolutional Neural Network. International Journal of Engineering Trends and Technology Volume 69 Issue 7, 51-55. doi:10.14445/22315381/IJETT-V69I7P207

[7] Jana S., Parekh R., Sarkar B., “Detection of Rotten Fruits and Vegetables Using Deep Learning”, In: Uddin, M.S., Bansal, J.C. (eds) Computer Vision and Machine Learning in Agriculture. Algorithms for Intelligent Systems. Springer, Singapore, 24 March 2021.

[8] Anuja Bhargava, Atul Bansal, “Fruits and vegetables quality evaluation using computer vision: A review”, Journal of King Saud University - Computer and Information Sciences, Volume 33, Issue 3, 2021.

[9] Deepali M Bongulwar. “Identification of Fruits Using Deep Learning Approach”, IOP Conference Series: Materials Science and Engineering, 2021, doi:10.1088/1757-899X/1049/1/012004

[10] Suescún, César & Pinzón Arenas, Javier & Moreno, Robinson. (2020). Fruit Identification and Quality Detection by Means of DAG-CNN. International Journal on Advanced Science Engineering and Information Technology. 10. 2183-2188. 10.18517/ijaseit.10.5.8684.

[11] Perez-Daniel K., Fierro-Radilla A., Peñaloza-Cobos J.P. (2020) Rotten Fruit Detection Using a One Stage Object Detector. In: Martínez-Villaseñor L., Herrera-Alcántara O., Ponce H., Castro-Espinoza F.A. (eds) Advances in Computational Intelligence. MICAI 2020. Lecture

Notes in Computer Science, vol 12469. Springer, Cham, DOI: 10.1007/978-3-030-60887-3_29

[12] Naranjo Torres, José & Mora, Marco & Hernández García, Ruber & Barrientos, Ricardo & Fredes, Claudio & Valenzuela Keller, Andres. (2020). A Review of Convolutional Neural Network Applied to Fruit Image Processing. Applied Sciences. 10. 3443. 10.3390/app10103443.

[13] D. Karakaya, O. Ulucan and M. Turkan, "A Comparative Analysis on Fruit Freshness Classification," 2019 Innovations in Intelligent Systems and Applications Conference (ASYU), 2019, pp. 1-4, doi: 10.1109/ASYU48272.2019.8946385.

[14] K. Roy, S. S. Chaudhuri, S. Bhattacharjee, S. Manna and T. Chakraborty, "Segmentation Techniques for Rotten Fruit detection," 2019 International Conference on Opto-Electronics and Applied Optics (Optronix), 2019, pp. 1-4, doi: 10.1109/OPTRONIX.2019.8862367.

[15] K. R. Sriram, "Kaggle - Fruits fresh and rotten for classification," 2018. [Online]. Available: https://www.kaggle.com/sriramr/fruitsfresh-and-rotten-for-classification

[16] C.G. Pachón, J.O. Pinzón, and R.J. Moreno, “FRUIT-16K”, [Online], Available: https://github.com/DEEP-CGPS/FRUIT-16K.git

[17] Oltean, M. Fruits 360 dataset. Mendeley Data 2018.
Available: https://data.mendeley.com/datasets/rp73yg93n8/

Custom DataSet : FruitsDataset