**QUALITY ASSESSMENT OF TOMATOES USING COMPUTER VISION**

# ABSTRACT

Agriculture sector plays a key role in the economic development of India. However, fruit grading by human is time consuming, labor intensive and prone to error. An automated grading system not only speeds up the time of processing, but also minimizes error.

The tomato fruit is very delicate and hence careful handling of this fruit is required during grading. This document proposes an effective tomato fruit grading system based on computer vision techniques which can be automated in future development. A software is developed using image processing techniques to analyze the tomato fruit for defects and ripeness. Experiments were carried out on several images of tomatoes for their quality.

In this project, we have developed near real-time Quality Assessment System from various tomatoes’ images using computer vision. The Quality Assessment System mainly focuses on the features of tomatoes like color, shape, surface defects and utilize them to assess the fruit.

An User Interface is created for educational purposes, where an image of tomato can be captured or uploaded and then assessed (as good or bad). It was observed that the proposed method was successful with 86.7% accuracy in predicting the quality of the tomato. In future developments, the grading system can be automated.

**LIST OF FIGURES AND TABLES**

**FIGURES**

Fig 1.1: Forms of tomatoes 2

Fig 1.2: Few Images used in this project 3

Fig 2.1: Process to assess the tomato through user interface 6

Fig 2.2: Process flow during quality assessment 7

Fig 5.1: Modules in the project 14

Fig 5.2: Use case diagram of Quality Assessment System 15

Fig 5.3: Class diagram for Quality Assessment System 16

Fig 5.4: Sequence diagram for Quality Assessment System 17

Fig 5.5: Activity diagram for Quality Assessment System 21

Fig 5.6: System Architecture for the Quality Assessment System 22

Fig 6.1: Images of tomatoes before pre-processing 28

Fig 6.2: Images of tomatoes after cropping 29

Fig 6.3: Images of tomatoes after complete pre-processing 29

Fig 6.4: Pre-processed images after labelling them 30

Fig 6.5: Convolution 31

Fig 6.6: Maxpooling 32

Fig 6.7: Activation 33

Fig 6.8: Flatten 34

Fig 6.9: Dropout 34

Fig 6.10: 3D CNN 35

Fig 6.11: Defining the neural network 36

Fig 6.12: Data generator for training and validation 36

Fig 6.13: Generating the model 37

Fig 6.14: Saving the Model 37

Fig 6.15: Client Server Architecture for static pages 38

Fig 6.16: Client Server Architecture for dynamic pages 39

Fig 6.17: Home page 39

Fig 6.18: File browser 40

Fig 6.19: Display page 40

Fig 6.20: Assess page 41

Fig 7.1: Testing with proper inputs 42

Fig 7.2: Testing with improper inputs 43

Fig 8.1: Training Accuracy 44

Fig 8.2: Overall model accuracy after testing 44

Fig 8.3: Confusion Matrix 45

Fig 8.4: Information from confusion matrix 45

Fig 9.1: Creating Environment 46

Fig 9.2: Activating Environment 46

Fig 9.3: Installing keras 46

Fig 9.4: Installing Tensorflow 47

Fig 9.5: Installing PIL 47

Fig 9.6: Installing Scikit-learn 47

Fig 9.7: Xampp control panel 48

Fig 9.8: Changing max\_execution\_time 49

Fig 9.9: Changing upload\_max\_filesize 49

Fig 9.10: Setting path for python files 50

Fig 9.11: Setting path for python execution in PHP files 50

**TABLES**

Table 5.1: Activity Diagram Symbols 18

**TABLE OF CONTENTS**

**CHAPTER 1**

1. INTRODUCTION 1

1.1 Related work 2

1.1.1 Data Gathering 2

1.1.2 Classification of tomatoes 4

**CHAPTER 2**

2. EXISTING SYSTEM AND PROPOSED SYSTEM 5

2.1 Existing System 5

2.1.1 Assessment at the farm 5

2.1.2 Assessment at the market 5

2.2 Drawbacks of Existing System 5

2.3 Proposed System 6

**CHAPTER 3**

3. Feasibility Study 8

3.1 Technical Feasibility 8

3.2 Economic Feasibility 8

3.3 Operational Feasibility 9

3.4 Social Feasibility 9

**CHAPTER 4**

4. SYSTEM ANALYSIS 10

4.1 Requirement Analysis 10

4.2 Requirements Specification 10

4.2.1 Data 10

4.2.2 Hardware 10

4.2.2.1 Training 11

4.2.2.2 Deployment 11

4.2.3 Software 11

4.2.3.1 Training 12

4.2.3.2 Deployment 12

4.3 Functional Requirements 12

4.3.1 User Interface 12

4.3.2 Server 12

**CHAPTER 5**

5. SOFTWARE DESIGN 13

5.1 Project Modules 13

5.1.1 Fruit Handling module 13

5.1.2 Quality Assessment Module 13

5.2 UML Diagrams 14

5.2.1 Use case Diagram 15

5.2.2 Class Diagram 16

5.2.3 Sequence Diagram 17

5.2.4 Activity Diagram 18

5.2.5 System Architecture 21

**CHAPTER 6**

6. IMPLEMENTATION 23

6.1 Environment 23

6.1.1 NumPy 23

6.1.2 Matplotlib 25

6.1.3 Keras 25

6.1.4 TensorFlow 26

6.1.5 PIL (Python Imaging Library) 27

6.1.6 Scikit-learn 28

6.2 Pre-processing 28

6.3 Model Development 30

6.3.1 Neural Networks 31

6.3.1.1 Convolution 31

6.3.1.2 Maxpooling 32

6.3.1.3 Activation 33

6.3.1.4 Flatten 33

6.3.1.5 Dense 34

6.3.1.6 Dropout 34

6.3.1.7 Convolution neural network with all connected layers 35

6.3.2 Training Model 35

6.3.2.1 Defining the neural network 35

6.3.2.2 Supplying data to neural network 36

6.3.3 Prediction 37

6.4 User Interface 37

**CHAPTER 7**

7. TESTING 42

**CHAPTER 8**

8. RESULTS 44

8.1 Training Accuracy 44

8.2 Testing Accuracy 44

8.3 Confusion Matrix 44

**CHAPTER 9**

9. INSTALLATION OF SOFTWARE AND DEPENDENCIES 46

9.1 Environment for Python 46

9.2 Installing and setting up Server 47

9.3 Execution 50

**CHAPTER 10**

10. CONCLUSION 51

**CHAPTER 11**

11. FUTURE SCOPE 51

**CHAPTER 12**

12. REFERENCES 52

**1. INTRODUCTION**

In most developing nations such as India, agriculture constitutes the major part of the country’s economy. It is due to this fact that a lot of money is spent each year by governments across the globe, for utilizing new technologies, discovering new methodologies of farming and also new techniques for prevention of crop reduction due to pests, natural calamities and drought. Generally, in India the quality inspection of fruits is performed by human experts. This manual sorting by visual inspection is labour intensive, time consuming and suffers from the problem of inconsistency and inaccuracy in judgment by different human. With the advent of fast and high precision machine vision technologies, automation of the grading process is expected to reduce labour cost, improve the efficiency and accuracy of the sorting process.

Machine vision and image processing techniques have been found increasingly useful in the fruit industry, especially for applications in quality inspection and defect sorting applications. The fruit produced in the farm is sorted according to quality, and then transported to different standard markets at different distances. Tomato is the world’s largest crop and known as a protective food both because of its special nutritive value and also because of its wide spread production. Tomatoes are grown extensively in India, producing about 9.362 million tons with an area of about 535,000 ha. Foreign trade policies have given high importance in boosting our agricultural exports. The surface of the tomato fruit is very soft and hence careful handling of the fruit is needed during quality evaluation.

Quality of tomato fruit, in particular, depends on size, colour, shape and the presence and type of surface defects. The ripeness and defect are the most important factors that determine the quality of the tomato fruit. A tomato grading system helps both consumers and farmers by providing good quality fruits in the market. Hence, this document proposes to develop a system for quality inspection of the tomato using image processing techniques. In this document, few achievements obtained through computer vision in the process of assessment of tomatoes are discussed. We will discuss about the data which is used in this project, and how we obtained it and the results of this project. It is then followed by the framework used in this project and the reasons for the development of User interface. Finally, we will conclude with a few plans on how to improve the scope and results of the project and how it will help the people and market in providing great quality Tomatoes.

**1.1 Related work**

In this section, we discuss about the gathered data and it’s usage in this project and how the tomatoes are classified.

**1.1.1 Data Gathering**

The computer vision used in this project mainly depends on the images of the tomatoes. So, we decided to capture few images of tomatoes and utilise them to make a model which helps in assessing the tomatoes. The images are captured manually with a white background using a camera of 13MP. Each image contains only one tomato fruit. A total of 310 images of tomatoes are captured which are used for training and validating the model. A total of 53 images are used for testing the model which was built using the training data.

Tomatoes come in various shapes as shown in the fig-1.1,

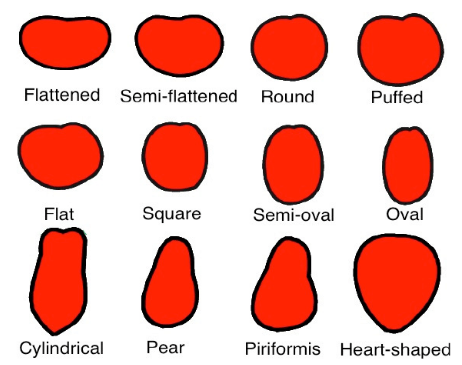


Fig. 1.1: Forms of Tomatoes

We couldn’t gather all forms of Tomatoes. In this project, we just used Round form of Tomatoes and a few Puffed, and semi-oval shaped tomatoes as well.

tomato ripeness can

be classified into six stages:

Stage 1 - Green mature: The entire surface of the fruit is

green, with the tone of green varying according to variety.

Stage 2 - Breaking: A color other than green appears on

not more than 10% of the fruit surface.

Stage 3 - Turning: Between 10 to 30% of the fruit surface

is colored pale yellow, pink, red or a combination of

these.

Stage 4 - Pink: Between 30 to 60% of the surface shows

a pink or red color.

Stage 5 - Light red: Between 60 to 90% of the surface is

red.

Stage 6 - Red: More than 90% of the surface is red.

tomato ripeness can

be classified into six stages:

Stage 1 - Green mature: The entire surface of the fruit is

green, with the tone of green varying according to variety.

Stage 2 - Breaking: A color other than green appears on

not more than 10% of the fruit surface.

Stage 3 - Turning: Between 10 to 30% of the fruit surface

is colored pale yellow, pink, red or a combination of

these.

Stage 4 - Pink: Between 30 to 60% of the surface shows

a pink or red color.

Stage 5 - Light red: Between 60 to 90% of the surface is

red.

Stage 6 - Red: More than 90% of the surface is red.

tomato ripeness can

be classified into six stages:

Stage 1 - Green mature: The entire surface of the fruit is

green, with the tone of green varying according to variety.

Stage 2 - Breaking: A color other than green appears on

not more than 10% of the fruit surface.

Stage 3 - Turning: Between 10 to 30% of the fruit surface

is colored pale yellow, pink, red or a combination of

these.

Stage 4 - Pink: Between 30 to 60% of the surface shows

a pink or red color.

Stage 5 - Light red: Between 60 to 90% of the surface is

red.

Stage 6 - Red: More than 90% of the surface is red.

tomato ripeness can

be classified into six stages:

Stage 1 - Green mature: The entire surface of the fruit is

green, with the tone of green varying according to variety.

Stage 2 - Breaking: A color other than green appears on

not more than 10% of the fruit surface.

Stage 3 - Turning: Between 10 to 30% of the fruit surface

is colored pale yellow, pink, red or a combination of

these.

Stage 4 - Pink: Between 30 to 60% of the surface shows

a pink or red color.

Stage 5 - Light red: Between 60 to 90% of the surface is

red.

Stage 6 - Red: More than 90% of the surface is red.

In General, the ripeness of tomatoes can be classified into six stages -

**Stage 1 (Green mature):** The entire surface of the fruit is green, with the tone of green varying according to variety.

**Stage 2** **(Breaking):** A colour other than green appears on not more than 10% of the fruit surface.

**Stage 3** **(Turning):** Between 10 to 30% of the surface is coloured yellow, pink, red or a combination of these.

**Stage 4** **(Pink):** Between 30 to 60% of the surface shows a pink or red colour.

**Stage 5** **(Light red):** Between 60 to 90% of the surface is red.

**Stage 6** **(Red):** More than 90% of the surface is red.

The types of tomatoes that are used while capturing images are:

1. Ripe tomatoes.
2. Unripe tomatoes.
3. Rotten tomatoes.
4. Tomatoes with surface defects.



Fig. 1.2: Few images of tomatoes used in this project.

All the images that are captured are utilized in building/training a model through convolutional neural network that helps in classifying the tomatoes. The details about building the model are discussed later in this document along with the technologies that are used in this project development.

**1.1.2 Classification of tomatoes**

Several physical and chemical parameters affect the quality of tomatoes. Studies have shown that there is a positive correlation between the ripening of tomatoes and their physical properties such as colour and firmness. In this project, the ripe tomatoes are classified as good tomatoes and all other categories of tomatoes i. e, unripe, rotten tomatoes and tomatoes with surface defects are classified as bad tomatoes. The reason for such a classification is that the ripe tomatoes are the only tomatoes that can be consumed immediately. The unripe tomatoes can be used in few ways but still they aren’t consumable immediately. As a reason they are classified as bad tomatoes in this project.

**2. EXISTING SYSTEM AND PROPOSED SYSTEM**

**2.1 Existing System**

The quality of tomatoes is being assessed manually by people at various stages. As the tomatoes are very delicate and can be easily spoiled, assessment at various stages is needed. The stages where the assessment is needed mainly are as follows:

**2.1.1 Assessment at the farm**

The farmers continuously assess the quality of tomatoes that are growing in the fields. Many chemicals and pesticides are used while growing the tomatoes at the farm. In this process, some tomatoes may get spoiled due to excessive use of chemicals. Assessing such tomatoes is very much necessary. This process of assessment is extremely difficult and time consuming.

**2.1.2 Assessment at the market**

After growing the tomatoes, they are sent to the market where the common people can buy them. Before selling the tomatoes to the consumers, the retailers have to transport the tomatoes from the farm to the market. During the transportation, few tomatoes might get damaged. The tomatoes must be assessed once again for the quality before selling them to the consumers. As this process is time consuming, it is overlooked most of the times. Even if the tomatoes are assessed by the people, it is prone to errors as it is done manually. Due to this tomatoes with poor quality are being sold to the consumers.

**2.2 Drawbacks of Existing System:**

The drawbacks of existing system are as follows:

* **Error prone:** As the assessment is done manually and involves many people, it is usually prone to error. Different people may classify tomatoes differently.
* **Time consuming:** The assessment of tomatoes manually, consumes excess amount of time, as each tomato must be assessed individually.
* **Difficult task:** Assessing each tomato by individual person is a difficult task and consumes lot of man power and time.

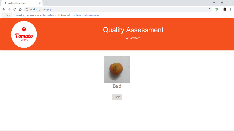
To overcome these drawbacks, the process of the quality assessment of tomatoes must be done using some computer/machine-based model. With this model, the process of quality assessment may be done with in short amount of time and with ease.

**2.3 Proposed System:**

A computer vision-based Quality Assessment System to assess the quality of tomatoes is proposed to overcome the drawbacks of the existing system. The idea of the Quality Assessment System is to automate the process of the assessment of the tomatoes and thereby reducing the efforts made by assessing them manually. But for educational purposes, in this project, a user interface has been created where the user can capture or upload images of tomatoes and assess them. The flow of process while using the user interface is shown in fig. 2.1.

The Quality Assessment System uses a trained model to assess the quality of the tomatoes. The model is trained by using previously labelled (Good or Bad) images of tomatoes with the help of machine learning algorithms like convolutional neural networks. After a model is trained it is validated for training accuracy and is followed by testing of the model with different set of images both for validating and testing of the model. Then the model is integrated to the user interface, in this project, to allow the user to assess the quality of tomatoes.

The proposed method implements an effective Quality Assessment System using various features of the tomato. The process flow during the assessment of the quality of the tomato is as shown in fig. 2.2.



Tomato image Capture/Upload image Assess (Good / Bad)

Fig. 2.1: Process to assess the tomato through user interface.

Capture Image

Pre-Processing

Store Image

Feature Extraction

Classification

Fig. 2.2: Process flow during quality assessment

**3. FEASIBILITY STUDY**

The feasibility of the project is studied in this section and business proposal is put forth with a general plan for the project and some cost estimates. For feasibility analysis, some major requirements for the system is essential.

**3.1 Technical feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. The technical needs of a system vary considerably but might include:

* The facility to produce outputs in each time.
* Response time under certain conditions.
* Ability to process a certain volume of images at a specified speed.

The technical feasibility also includes the software and hardware of the system. The level of technical requirements comprises of the software tools, the development platform and the machine environment, can be easily acquired.

The technical requirements for the Quality Assessment System are met with bare minimum specifications. A user interface is also created which helps the user to access the system with ease. Every time an image is uploaded, the quality of fruit in the image can be assessed.

**3.2 Economic Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available.

The economy that is used to develop a model is very high depending on the technical specifications of the system. The operational cost of the project is low and eventually it can be minimized to just few expenses like power supply to the system.

The Quality Assessment System needs a very high economy as it involves research on the tomatoes and gather data about various types and shapes of tomatoes. The training of a machine learning model also involves high economy with heavy resources. Once the system is deployed, the break-even point can be reached soon and profits from the system can also be obtained.

**3.3 Operational Feasibility**

It determines how acceptable the software is within the organization. As per this project, the Quality Assessment System can be utilized by anyone due to its user-friendly interface. In the future developments, if the assessment process is automated and can also be operational near the market yards and farms.

**3.4 Social feasibility**

It determines the level of impact and acceptance of the system in the society. As the Quality Assessment System comes into usage, the quality of the tomatoes that enter into the market and that are sold to the consumers will improve. As a result, the consumers will be satisfied by the quality assessment of the tomatoes.

**4. SYSTEM ANALYSIS**

**4.1 Requirement analysis**

A requirement is a feature that must be included in the system. Before the actual design and implementation start, getting to know the system to be implemented is of prime importance. The main requirements are:

* **Input:** As the Quality Assessment System needs images as input, there should be a camera to capture images or a way to upload the images into the system.
* **A user interface:** As the project is for educational purposes, a user interface is needed so that the users can capture/upload images of tomatoes.
* **Quality Assessment System:** A reliable Quality Assessment System is required which helps in assessing and returning the quality of the tomatoes as good or bad.
* **Output:** After assessment of the tomato by the Quality Assessment System, the results should be displayed as good or bad.

**4.2 Requirements Specification**

* + 1. **Data**

The data is the essential part of the project. The Quality Assessment System utilises the images of the tomatoes and assesses them as good or bad quality tomatoes. Hence, the type of the data which we require is image data. As the image data for tomatoes is not available in the internet, the images that are required for the project are captured manually. Each image contains only one tomato fruit with a white background. To capture an image, a camera of 13MP is used. As capturing images manually is time consuming work, only a few images were captured and are used in developing the model required for Quality Assessment.

* + 1. **Hardware**

A decent amount of hardware is required for the project at various stages and environments. The various stages involved are:

* + - 1. **Training**

The Quality Assessment System utilises a previously trained model to assess the quality of the tomatoes. To train the model on huge amounts of image datasets, a GPU with at least a computational capability of 5.2 is required with at least 4GB RAM and intel i5 processor. These requirements are the bare minimum and we have used a AMD RADEON GPU with 4 GB RAM. As the computational capability of the GPU and the RAM increase, the speed of the training of model increases significantly.

* + - 1. **Deployment**

As this project is for educational purposes, a user interface is created so that every person can use the Quality Assessment System to assess the tomatoes by capturing their images or by uploading some of the images from their local system into the server. The requirements for the user to be able to use the Quality Assessment System are:

**At user end**

* Minimum of 4GB RAM
* Internet connectivity
* Camera
* Minimum storage space of 2GB

**At server end**

* GPU with computational capability 5.2 or above
* Minimum of 4GB RAM
* Minimum of 300GB storage space
* Intel core processor i5 or above
  + 1. **Software**

The software and tools that is required for the project that helps in quality assessment of tomatoes are:

* + - 1. **Training**

The software and tools that are required to train the model that helps in quality assessment of tomatoes are:

* Anaconda navigator
* Xampp server
* PHP
* Python 3.5 and above with dependencies
  + - 1. **Deployment**

**At user end**

* Windows 8/8.1/10 or Linux or Mac
* HTML
* PHP

**At server end**

* Xampp server
* Python 3.5 or above
* Dependencies of Python like keras, tensorflow.
* Windows 10 operating system

**4.3 Functional Requirements**

The functional requirements documents about the operations and activities that a system performs. The functional requirements of the Quality Assessment System are as follows:

**4.3.1 User Interface**

* Capture and upload images to the server
* Display uploaded images
* Display results

**4.3.2 Server**

* Fetch the uploaded files
* Pre-process the images
* Save the pre-processed images
* Run the model to predict the pre-processed images
* Send the results to the user interface
* Save the files into a filesystem

**5. SOFTWARE DESIGN**

* 1. **Project modules**

The modules that are present in Quality Assessment System and their roles are:

**5.1.1** **Fruit handling module**

The Fruit handling system contains the following:

* Computer
* Camera

The camera captures the images and uploads to the computer which acts as a server. The computer then stores the images. These images are assessed using the quality assessment module.

* + 1. **Quality Assessment module**

The Quality assessment module consists of the following operations:

* Pre-process
* Feature Extraction and selection
* Classification
* Return results

The Quality assessment module assesses the images. It pre-processes the images, then extracts the features from the images like colour, shape etc using neural networks. Using the extracted data, it then classifies the tomato as good or bad. Then the results are sent back to the Fruit handling system where they are shown to the user.

The Modules of the Quality Assessment System are shown in fig. 5.1

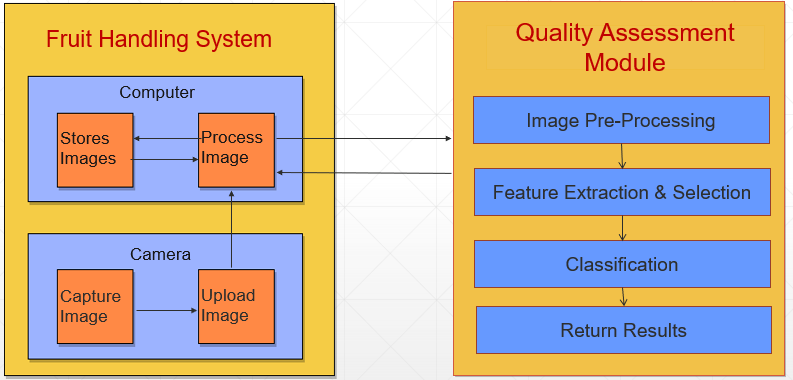


Fig. 5.1: Modules in the project

* 1. **UML Diagrams**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

* User Model View
* This view represents the system from the user’s perspective.
* The analysis representation describes a usage scenario from the end-user’s perspective.
* Structural Model View
* In this model the data and functionality are arrived from inside the system.
* This model view models the static structures.
* Behavioural Model View
* It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**5.2.1 Use Case Diagram**

Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from the external point of view.

The actor is outside the boundary of the system, whereas the use cases are inside the boundary of the system. The use case diagram for quality assessment is shown in Fig. 5.2

The Quality Assessment System has mainly two actors. They are:

* Fruit Handling System
* Quality assessment module

The Quality Assessment System consists of mainly 8 use cases which are the major functions/behaviours of the system. As shown in Fig. 5.2, there are 4 use cases associated with fruit handling system and 4 use cases associated with quality assessment module.

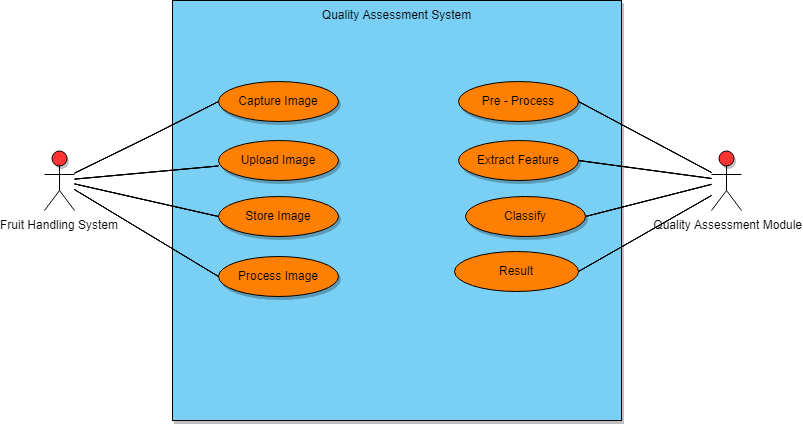
****

Fig. 5.2: Use case Diagram of Quality Assessment System

The Fruit handling system first captures the images using a camera present in it. Then it uploads the images to a computer and stores them in the computer. Then the fruit handling system utilises the quality assessment module to process the images of the tomatoes.

The quality assessment module pre-processes the images that it gets from the fruit handling system. Then the features of the tomato like colour, shape etc., are extracted from the image using neural networks and then they are used for the classification of the tomato as good or bad. In this project, the fruit handling system and Quality Assessment System are present in only one computer. So, the computer acts as both client and server.

**5.2.2 Class Diagram**

Class Diagrams are used to describe the structure of the system. Classes are abstraction that specify the common structure and behaviour of a set of objects.

Class diagrams describe the system in terms of objects, classes, attributes, operations and their associations.

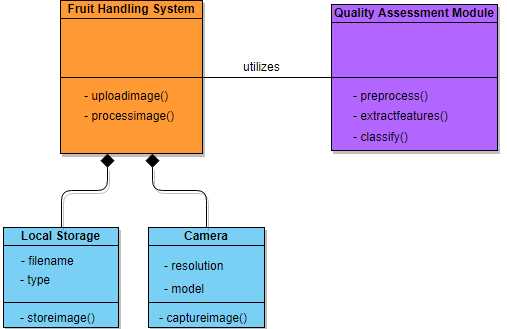
****

Fig 5.3: Class Diagram for Quality Assessment System

The Quality Assessment System has mainly 2 classes. They are Fruit handling system and Quality assessment module.

The Fruit handling system has two classes as its composition of Camera and Local storage in computer. The fruit handling system has two methods namely uploadimage(), captureimage(). The Local Storage is used to save the images using storeimages() and it has filename and type as its attributes. The model also consists of a camera which is used to capture the images using captureimage() with resolution and model acting as its attributes.

The Quality Assessment System consists of three methods namely preprocess(), extractfeatures() and classify().

**5.2.3 Sequence Diagram**

Sequence Diagrams are used to illustrate the flow of control across the entire system. A Sequence Diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence Diagrams are typically associated with the use case realizations in the Logical View of the system under development. Sequence Diagrams are sometimes also called as Event Diagrams or Event Scenarios.

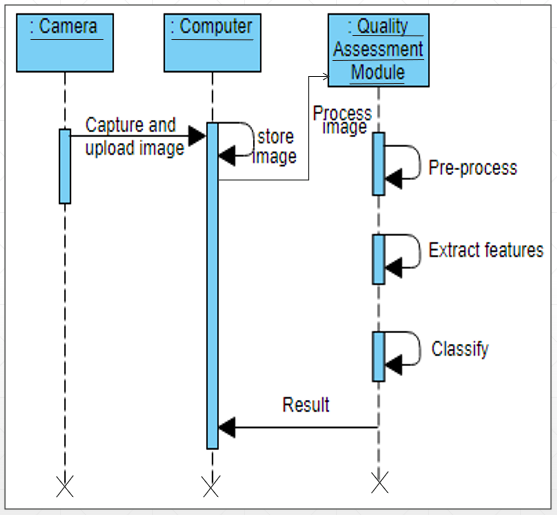


Fig. 5.4: Sequence Diagram of Quality Assessment System

A Sequence Diagram shows, as parallel vertical lines (Lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows specification of simple run-time scenario’s in a graphical manner.

The Quality Assessment System works in a sequential manner as shown in the figure 5.4. As shown in figure 5.4, the camera first captures the images and then sends it to the computer wherein the captured images are stored in the system. Then, the processing of the images is done via the Quality Assessment Module which helps us extract the features from the image and then classify them into categories (like whether the tomato is good or bad).

**5.2.4 Activity Diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

These activity diagram shapes and symbols are some of the most common types you'll find in UML diagrams -

| Symbol | Name | Description |
| --- | --- | --- |
| start Symbol | Start symbol | Represents the beginning of a process or workflow in an activity diagram. It can be used by itself or with a note symbol that explains the starting point. |
| activity Symbol | Activity symbol | Indicates the activities that make up a modeled process. These symbols, which include short descriptions within the shape, are the main building blocks of an activity diagram. |
| connector Symbol | Connector symbol | Shows the directional flow, or control flow, of the activity. An incoming arrow starts a step of an activity; once the step is completed, the flow continues with the outgoing arrow. |
| joint Symbol | Joint symbol/ Synchronization bar | Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time. Represented with a thick vertical or horizontal line. |
| fork Symbol | Fork symbol | Splits a single activity flow into two concurrent activities. Symbolized with multiple arrowed lines from a join. |
| decision Symbol | Decision symbol | Represents a decision and always has at least two paths branching out with condition text to allow users to view options. This symbol represents the branching or merging of various flows with the symbol acting as a frame or container. |
| note Symbol | Note symbol | Allows the diagram creators or collaborators to communicate additional messages that don't fit within the diagram itself. Leave notes for added clarity and specification. |
| send signal Symbol | Send signal symbol | Indicates that a signal is being sent to a receiving activity. |
| receive signal Symbol | Receive signal symbol | Demonstrates the acceptance of an event. After the event is received, the flow that comes from this action is completed. |
| shallow history pseudostate symbol | Shallow history pseudo state symbol | Represents a transition that invokes the last active state. |
| option loop symbol | Option loop symbol | Allows the creator to model a repetitive sequence within the option loop symbol. |
| flow final symbol | Flow final symbol | Represents the end of a specific process flow. The flow final symbol should be placed at the end of a process in a single activity flow. |
| condition text | Condition text | Placed next to a decision marker to let you know under what condition an activity flow should split off in that direction. |
| end symbol | End symbol | Marks the end state of an activity and represents the completion of all flows of a process. |

Table 5.1: Activity Diagram Symbols

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other diagrams are used to show the message flow from one object to another, but activity diagram is used to show message flow from one activity to another.

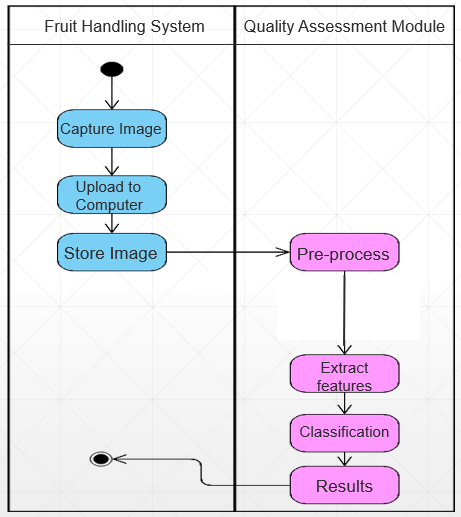


Fig. 5.5: Activity Diagram of Quality Assessment System

The fig. 5.5, depicts the activities involved in the Quality Assessment System. In the Quality Assessment System, the Fruit Handling System first captures the images. The captured images are uploaded to the computer and are stored in it. Then the fruit handling system utilises the Quality Assessment Module. The role of the Quality Assessment Module is to pre-processes the images, then extract the features from it using neural networks, then classify the images as good or bad, and finally return the results to the Fruit Handling System.

**5.2.5 System Architecture**

A system architecture or system’s architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

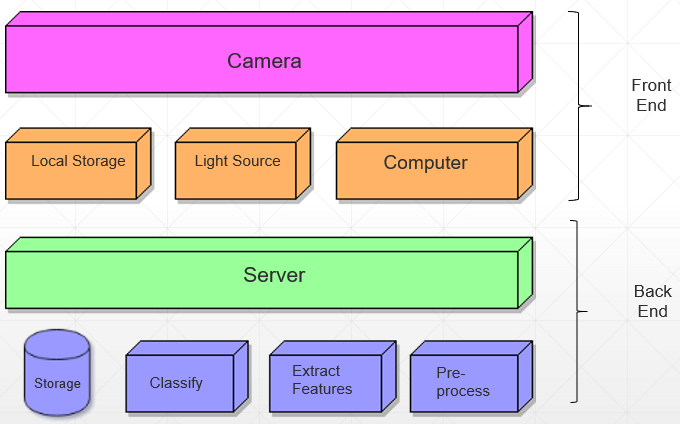


Fig. 5.6 System Architecture of the Quality Assessment System

The Quality Assessment System uses a three-tier system architecture. It consists of a user interface residing in the computer. A middleware which resides in the server. The computer acts for both user interface and as a server in this project. The filesystem in the computer is used to store the images and it acts as a database.

In the front end, the user interface which resides in the computer. The user interface can use the camera to capture the images or upload the images from the local file storage system. It utilises few modules to interact with the server.

The back end consists of the server and a filesystem. The server contains all the business logic and processes the images in it. The filesystem which acts as the database, is used to store the images in the computer.

**6. IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered as the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

The implementation of the Quality Assessment System involves the following:

* Environment to run the project
* Pre-processing
* Model development
* User Interface

**6.1 Environment**

The Quality Assessment System utilises python as its underlying programming language along with many other python dependencies. To properly manage all the dependencies of the python language, Anaconda navigator software is installed. The creation of the environment and installing all the software along with required dependencies is described in chapter 9 of this document. The dependencies of python required for the project are described below.

**6.1.1 NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

1. Arrays in NumPy: NumPy’s main object is the homogeneous multidimensional array.

* It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
* In NumPy dimensions are called *axes*. The number of axes is *rank*.
* NumPy’s array class is called ndarray. It is also known by the alias array.

2. Array creation: There are various ways to create arrays in NumPy.

* For example, you can create an array from a regular Python list or tuple using the array function. The type of the resulting array is deduced from the type of the elements in the sequences.
* Often, the elements of an array are originally unknown, but its size is known. Hence, NumPy offers several functions to create arrays with initial placeholder content. These minimize the necessity of growing arrays, an expensive operation.  
  For example: np.zeros, np.ones, np.full, np.empty, etc.
* To create sequences of numbers, NumPy provides a function analogous to range that returns arrays instead of lists.
* arange: returns evenly spaced values within a given interval. step size is specified.
* linspace: returns evenly spaced values within a given interval. num no. of elements are returned.
* Reshaping array: We can use reshape method to reshape an array. Consider an array with shape (a1, a2, a3, …, aN). We can reshape and convert it into another array with shape (b1, b2, b3, …, bM). The only required condition is:  
  a1 x a2 x a3 … x aN = b1 x b2 x b3 … x bM. (i.e original size of array remains unchanged.)
* Flatten array: We can use flatten method to get a copy of array collapsed into one dimension. It accepts *order* argument. Default value is ‘C’ (for row-major order). Use ‘F’ for column major order.

3. Array Indexing: Knowing the basics of array indexing is important for analysing and manipulating the array object. NumPy offers many ways to do array indexing.

* Slicing: Just like lists in python, NumPy arrays can be sliced. As arrays can be multidimensional, you need to specify a slice for each dimension of the array.
* Integer array indexing: In this method, lists are passed for indexing for each dimension. One to one mapping of corresponding elements is done to construct a new arbitrary array.
* Boolean array indexing: This method is used when we want to pick elements from array which satisfy some condition.

4. Basic operations: Plethora of built-in arithmetic functions are provided in NumPy.

* Operations on single array: We can use overloaded arithmetic operators to do element-wise operation on array to create a new array. In case of +=, -=, \*= operators, the existing array is modified.

**6.1.2 Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and Ipython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery**.**

**6.1.3 Keras**

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, [CNTK](https://github.com/Microsoft/cntk), or [Theano](https://github.com/Theano/Theano). It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

* Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
* Supports both convolutional networks and recurrent networks, as well as combinations of the two.
* Runs seamlessly on CPU and GPU.

## **Guiding principles**

* **User friendliness.** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
* **Modularity.** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.
* **Easy extensibility.** New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.
* **Work with Python**. No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

**6.1.4 TensorFlow**

TensorFlow is an open source software library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture enables you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. TensorFlow also includes [TensorBoard](https://www.tensorflow.org/guide/summaries_and_tensorboard), a data visualization toolkit.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

TensorFlow provides stable Python API and C APIs as well as without API backwards compatibility guarantee like C++, Go, Java, JavaScript and Swift.

The main components in a TensorFlow system are the client, which uses the Session interface to communicate with the master, and one or more worker processes, with each worker process responsible for arbitrating access to one or more computational devices (such as CPU cores or GPU cards) and for executing graph nodes on those devices as instructed by the master.

TensorFlow offers some powerful features such as: it allows computation mapping to multiple machines, unlike most other similar frameworks; it has built in support for automatic gradient computation; it can partially execute subgraphs of the entire graph and it can add constraints to devices, like placing nodes on devices of a certain type, ensure that two or more objects are placed in the same space etc.

**6.1.5 PIL (Python Imaging Library)**

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, supports Python 1.5.2–2.7, with Python 3support to be released "later".

Development appears to be discontinued with the last commit to the PIL repository coming in 2011. Consequently, a successor project called Pillow has forked the PIL repository and added Python 3.x support.

**Capabilities**

Pillow offers several standard procedures for image manipulation. These include:

* per-pixel manipulations,
* masking and transparency handling,
* image filtering, such as blurring, contouring, smoothing, or edge finding,
* image enhancing, such as sharpening, adjusting brightness, contrast or colour
* adding text to images and much more.

**File Formats**

Some of the file formats supported are [PPM](https://en.wikipedia.org/wiki/Netpbm_format), [PNG](https://en.wikipedia.org/wiki/Portable_Network_Graphics), [JPEG](https://en.wikipedia.org/wiki/JPEG), [GIF](https://en.wikipedia.org/wiki/GIF), [TIFF](https://en.wikipedia.org/wiki/TIFF), and [BMP](https://en.wikipedia.org/wiki/BMP_file_format). It is also possible to create new file decoders to expand the library of file formats accessible.

**6.1.6 Scikit-learn**

Scikit-learn (formerly scikits.learn) is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language. It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy). The scikit-learn is mostly used to perform calculations of confusion matrix.

**6.2 Pre-processing**

The captured images from the camera to the Quality Assessment System looks like the images in the figure shown below.



Img-1 Img-2 User Interface Output (Good/Bad)

Fig. 6.1: Images of tomatoes before pre-processing

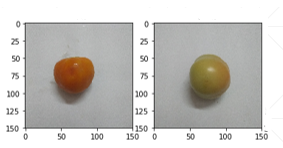
As the Quality Assessment System assess the tomatoes, the excess background image can be cropped for faster processing of the image. The captured images might look like as shown in the figure 6.2



Img-1 Img-2

Fig. 6.2: Images of tomatoes after cropping

These cropped images are fitted into a graph of size 150 X 150 i.e., the images are divided into blocks that can be used in training the model. But for understanding purpose, the images are shown as they are in fig 6.2 along with an axis which helps in understanding us, that the image is divided into 150 X 150 blocks, as shown in fig. 6.3



Img-1 Img-2

Fig. 6.3: Images of tomatoes after complete pre-processing

The pre-processed images as shown in the figure are labelled as 0.0 or 1.0 for good and bad respectively. The labelled images are as shown in the figure 6.4.

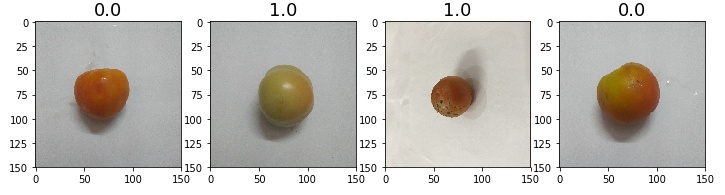


Fig. 6.4: pre-processed images after labelling them.

The final output for the pre-processing images are the labelled images of the tomatoes of dimensions 150 X 150. These dimensions fit exactly as the input dimensions for the neural network. But for better results, the dimensions of the images can be increased, which in-turn require higher computational resources and time required to train the model. Hence, for this project the dimensions of 150 X 150 are used for training the model and for predictions.

**6.3 Model Development**

Model development involves teaching a network of artificial of neurons, called neural network, with the image data that is labelled as the input. Then the neural network learns all the distinct features present in the images and will be able to classify them based on those features. For the development of model for the Quality Assessment System, the input to the neural network is image data that contains tomatoes with labels. After learning, the neural network will be able to predict the quality of the tomato as good or bad.

To achieve this, a convolutional neural network algorithm is used which contains various layers and different functions performing some mathematical operations on the input data and its labels provided.

Before going to the model which we developed we need to understand what a neural network is, and some functions like convolution, maxpooling, activation, flatten, dense and dropout which form the neural network.

**6.3.1 Neural networks**

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

**6.3.1.1 Convolution**

In [mathematics](https://en.wikipedia.org/wiki/Mathematics) (and, in particular, [functional analysis](https://en.wikipedia.org/wiki/Functional_analysis)) convolution is a [mathematical operation](https://en.wikipedia.org/wiki/Operation_(mathematics)) on two [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) (f and g) to produce a third function that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it. Convolution is similar to [cross-correlation](https://en.wikipedia.org/wiki/Cross-correlation).

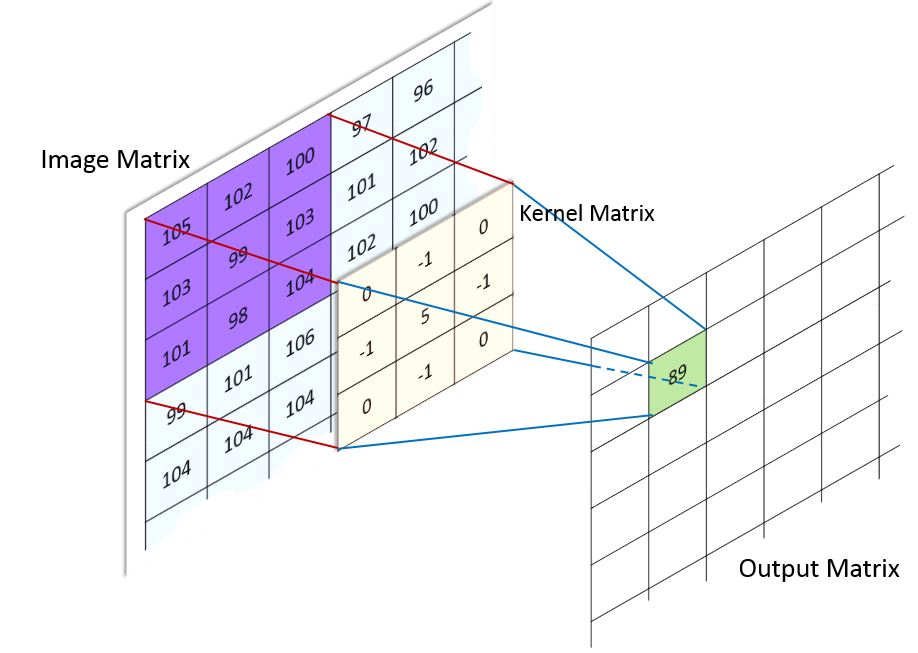


Fig:6.5 convolution

Convolutional Neural Networks are very similar to ordinary Neural Networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other.

Convolutional Neural Network architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network.

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation

**6.3.1.2 Maxpooling**

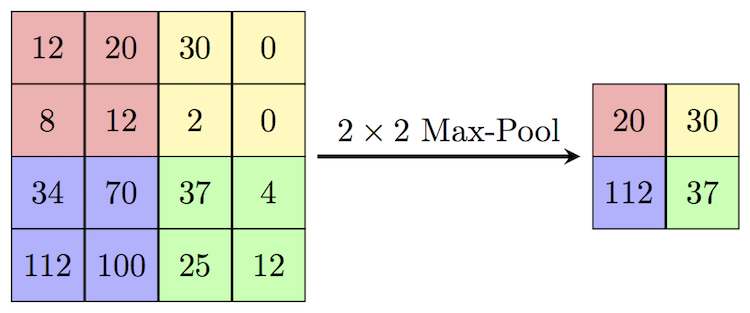


Fig:6.6 Maxpooling

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

Max pooling is done by applying a max filter to (usually) non-overlapping sub regions of the initial representation.

**6.3.1.3 Activation**

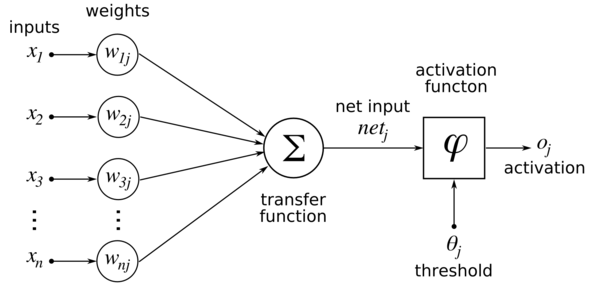


Fig. 6.7 Activation

Activation functions are really important for an Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable. They introduce non-linear properties to our Network*.* Their main purpose is to convert an input signal of a node in an A-NN to an output signal. That output signal now is used as an input in the next layer in the stack.

**6.3.1.4 Flatten**

It flattens the input by reducing it to lower dimensions. In this step, all the pooled feature maps are taken and put into a single vector. The Flatten function flattens all the feature maps into a single column.

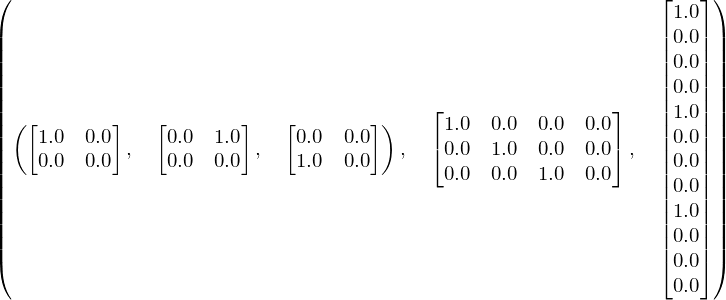


Fig:6.8 flatten

**6.3.1.5 Dense**

A dense layer represents a matrix vector multiplication. The values in the matrix are the trainable parameters which get updated during backpropagation. So, you get a m dimensional vector as output. A dense layer thus is used to change the dimensions of your vector. Mathematically speaking, it applies a rotation, scaling, translation transform to your vector.

**6.3.1.6 Dropout**

A dropout layer is used for regularization where you randomly set some of the dimensions of your input vector to be zero with probability given. A dropout layer does not have any trainable parameters i.e. nothing gets updated during backward pass of backpropagation. Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

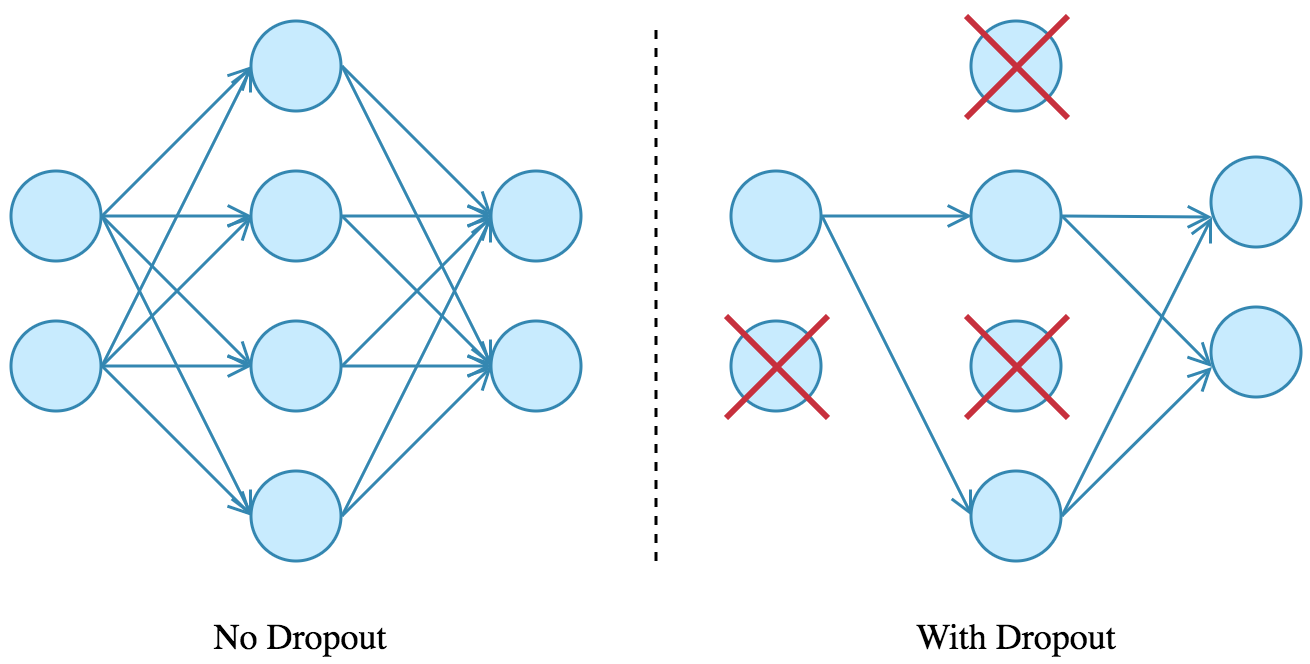


Fig:6.9 dropout

**6.3.1.7 Convolutional neural network with all connected layers**

Computers read images as pixels and it is expressed as matrix (NxNx3) — (height by width by depth). Images makes use of three channels (RGB), so that is why we have a depth of 3. The Convolutional Layer makes use of a set of learnable filters. A filter is used to detect the presence of specific features or patterns present in the original image (input). It is usually expressed as a matrix (MxMx3), with a smaller dimension but the same depth as the input file. This filter is convolved (slide) across the width and height of the input file, and a dot product is computed to give an activation map. Different filters which detect different features are convolved on the input file and a set of activation maps is outputted which is passed to the next layer in the CNN. See fig 6.10.

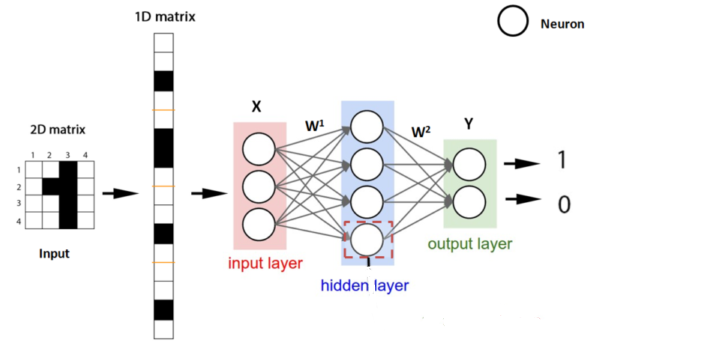


Fig. 6.10: 2D CNN

The neural network that is used for training for the Quality Assessment System contains about 260 thousand parameters and they continuously undergo changes as they learn from the images of dimensions 150 X 150.

**6.3.2 Training Model**

The training of the model involves defining the neural network and supplying the data to the neural network to train itself.

**6.3.2.1 Defining the Neural network**

Defining the neural network involves adding all the functions to a network as shown in the fig. 6.11.

In the fig. 6.11, the neural network is defined with one input layer, three hidden layers and one output layer. The output activation function used is sigmoid, which generates the output in the form of 0’s and 1’s, as Quality Assessment System assesses the tomatoes into good or bad.



Fig. 6.11: Defining the neural network

**6.3.2.2 Supplying Data to Neural Network**

The data containing pre-processed images of tomatoes are supplied as the input to the neural network. The input data is stored in a directory and a data generator is created which helps in flowing the data from the directory to the neural network as it learns to classify. Fig. 6.12 depicts the data generators for training and validating the network.

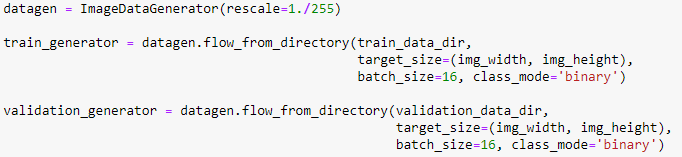


Fig. 6.12: Data generator for training and validation

Then the neural network is trained, and a model is generated using the image data. The generation of the model is shown in fig. 6.13.

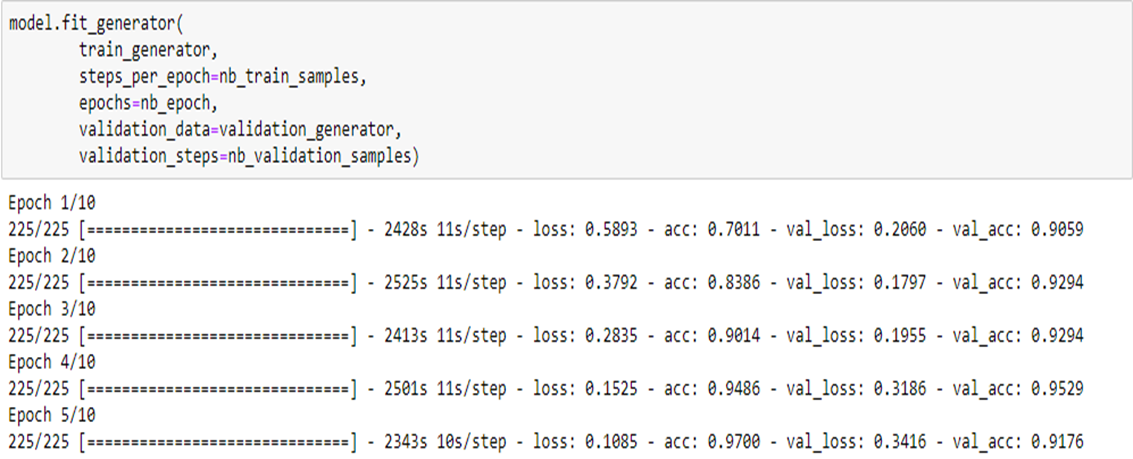


Fig 6.13: Generating the model

**6.3.3 Prediction**

Once the model is generated, it is saved into a file with extensions, json and h5 as shown in the fig. 6.14. This model is loaded whenever prediction of an image is required. These files are stored in the server and is used whenever necessary.

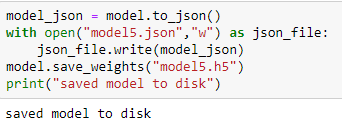


Fig. 6.14: Saving the model

**6.4 User Interface**

Firstly, let’s look briefly about Web Designing,

Web design is the process of creating websites. It encompasses several different aspects, including webpage layout, content production, and graphic design. While the terms web design and web development are often used interchangeably, web design is technically a subset of the broader category of web development. Websites are created using a mark-up language called HTML. Web designers build web-pages using HTML tags that define the content and metadata of each page. The layout and appearance of the elements within a webpage are typically defined using CSS, or cascading style sheets. Therefore, most websites include a combination of HTML and CSS that defines how each page will appear in a browser. Website is a location on web and is hosted on a web server. It is a set of related web pages. It is accessed using Internet address known as Uniform Resource Locator.

## **Static Websites**

Static websites are also known as flat or stationary websites. They are loaded on the client’s browser as exactly they are stored on the web server. Such websites contain only static information. User can only read the information but can’t do any modification or interact with the information.

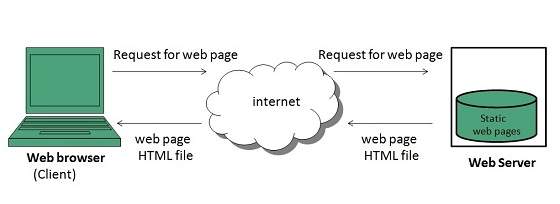


Fig. 6.15: Client Server Architecture for static pages

**Dynamic Websites**

Dynamic websites show different information at different point of time. It is possible to change a portion of a web page without loading the entire web page.

### Server-side dynamic web page

It is created by using server-side scripting. There are server-side scripting parameters that determine how to assemble a new web page which also include setting up of more client-side processing.

### Client-side dynamic web page

It is processed using client-side scripting such as JavaScript. And then passed in to **Document Object Model (DOM).**



Fig. 6.16: Client Server Architecture for dynamic pages

From project’s perspective, a user interface is created so that any user can utilise the Quality Assessment System and assess the quality of the tomatoes. This user interface is clearly for educational purpose only. The user interface consists of the following web pages:

* **A home page:** In this page, a user can either capture the image of a tomato using a camera or can upload the image of a tomato from the local storage of a system. The home page is shown in the fig. 6.17. A file browser for uploading an image is shown in the fig. 6.18.
* **Display page:** In this page, a user can view a semi pre-processed (only cropped version) of the image that is either captured or uploaded as shown in fig 6.19.
* **Assessing Page:** In this page, the quality of the tomato is assessed and is displayed below the image as good or bad as shown in fig 6.20.

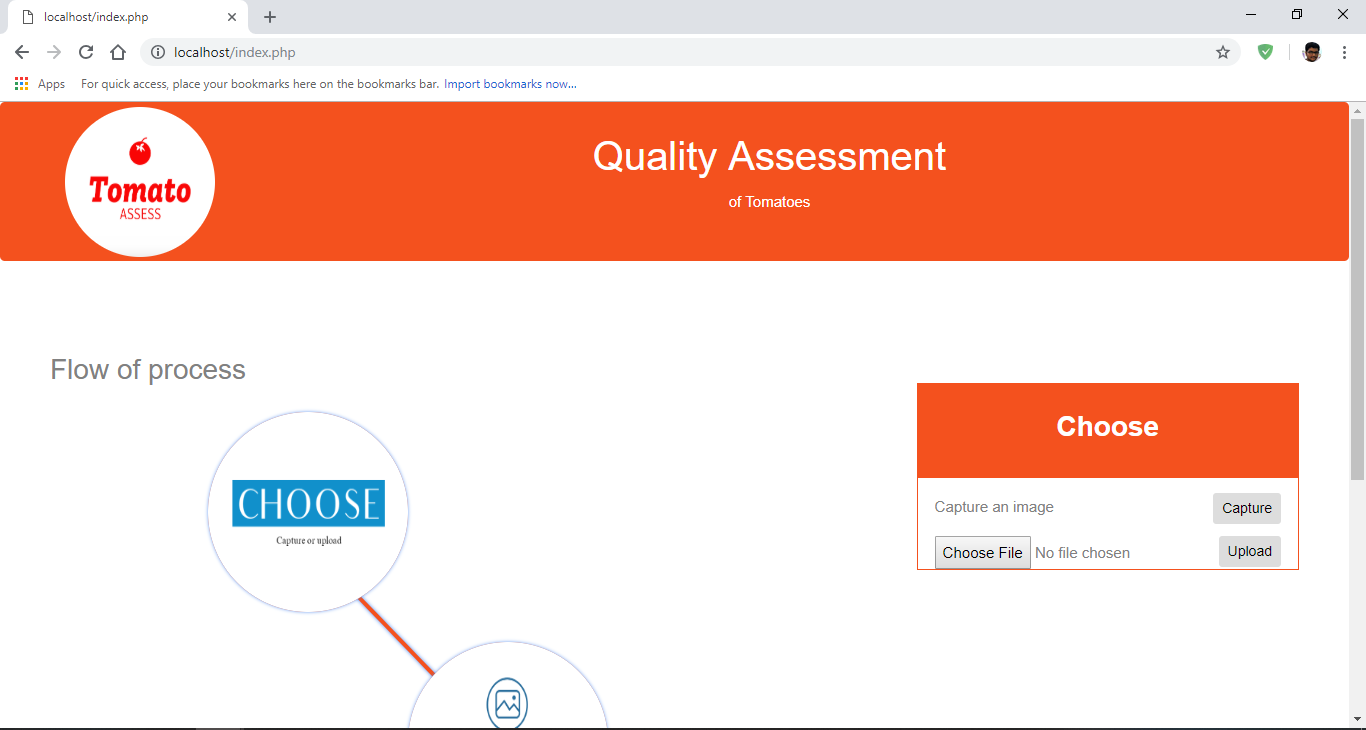


Fig. 6.17: Home Page

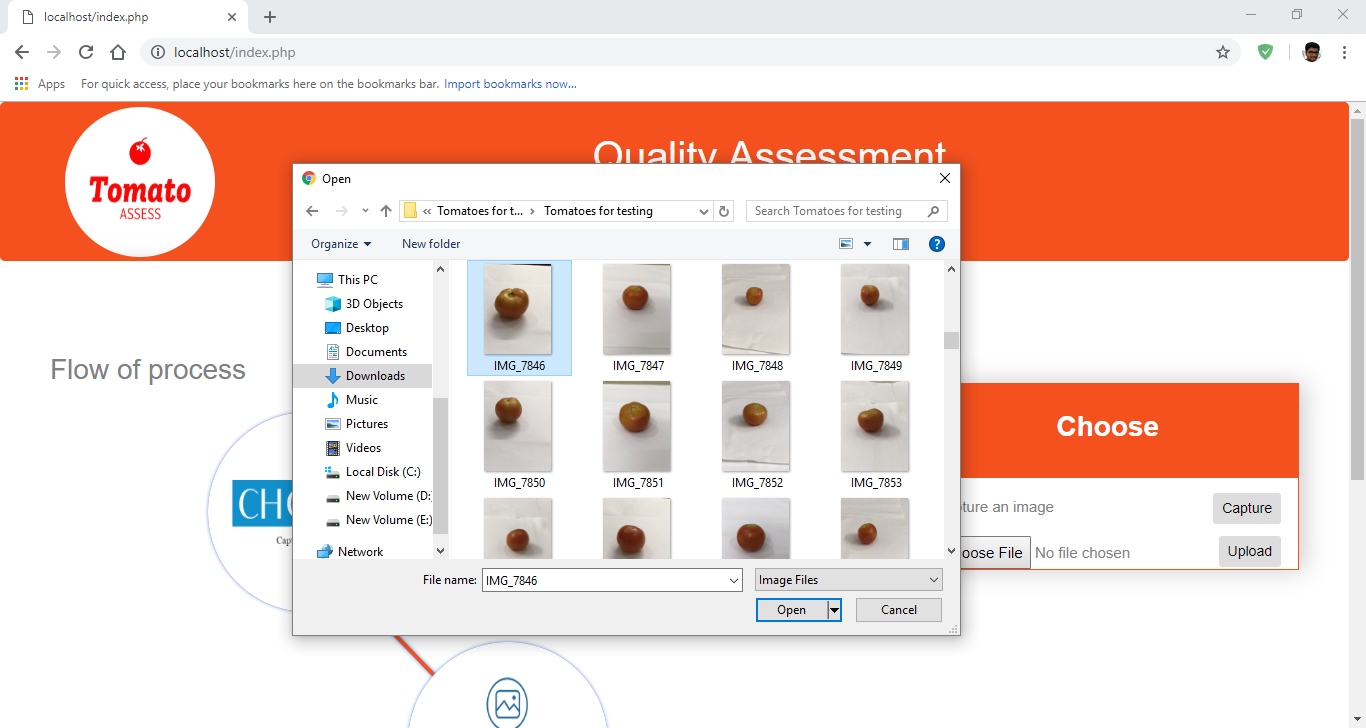


Fig. 6.18: File browser

After the completion of the assessment of the tomato using the image, the user can return to the home page and can upload other image for assessment if required.

Thus, a Quality Assessment System is implemented for assessing the tomatoes and can be used by any user as of now.



Fig. 6.19: Display page

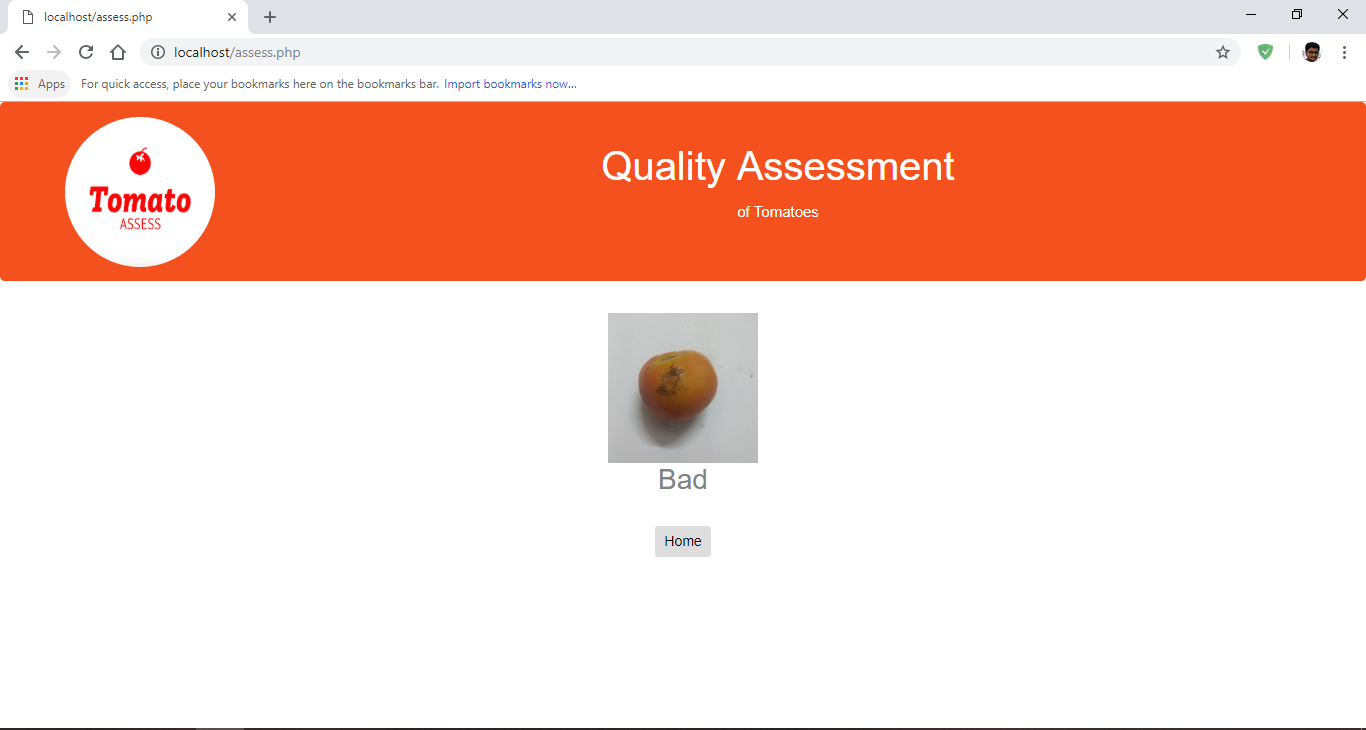


Fig. 6.20: Assess page

**7. TESTING**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

Testing tries to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

The Quality Assessment System uses a model which is trained on a dataset of 225 images and is validated upon a dataset of 85 images. The test cases are as follows:

* **Proper input:** The Quality Assessment System is given few inputs of images containing tomatoes. The results are as shown in the fig 7.1.

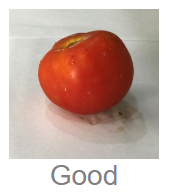
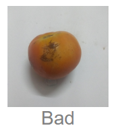
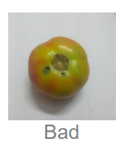


Fig 7.1: Testing with proper inputs

As shown in fig. 7.1 the tomatoes are classified as good or bad depending on the type of tomato. In the figure, the good tomatoes are classified as good and bad tomatoes are classified as bad.

**Improper input:** The Quality Assessment System is given few inputs of images containing fruits other than tomatoes. The results are as shown in the fig 7.2. As shown in fig. 7.2, the fruits other than tomato are classified as bad because they should not be classified into good when they are not even tomatoes. The Quality Assessment System is a classifier of only tomatoes, as the model used for this system is trained only for tomatoes. If any other fruit is given as an input to the Quality Assessment System, the results will be given as bad, as shown in fig. 7.2.

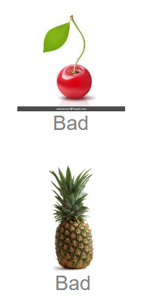
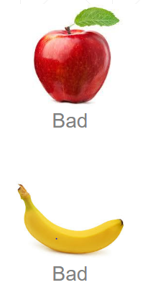


Fig. 7.2: Testing with improper inputs

**8. RESULTS**

**8.1 Training Accuracy**

The model is validated for training accuracy after completion of the training. The model is validated using 85 images and the validation accuracy is shown in fig 8.1

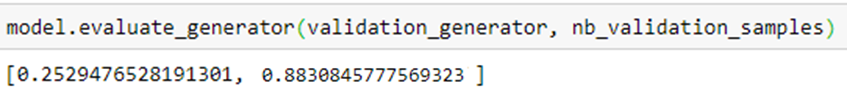


Fig. 8.1: Training accuracy

**8.2 Testing Accuracy**

Then the model is tested for overall accuracy at the completion of the development of the model. The accuracy of the model is shown in the fig. 8.2. The testing is done using 53 images of tomatoes.



Fig. 8.2: Overall Model accuracy after testing

**8.3 Confusion Matrix**

A confusion matrix is a summary of prediction results on a classification problem.  
The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions.  
It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made. The confusion matrix for the Quality Assessment System is given below

The information that can be obtained from the confusion matrix is shown in the fig. 8.4. It contains information like precision, recall, f1-score, support and also accuracy for that test data.

|  |  |  |
| --- | --- | --- |
| predicted  original | Positive | Negative |
| Positive | 39 | 2 |
| Negative | 4 | 8 |

Fig 8.3: Confusion Matrix

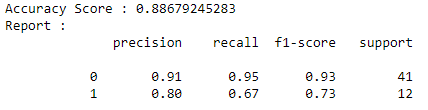


Fig. 8.4: Information from confusion matrix

**Precision:** Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

Precision = True positives / (True positives + False Positives)

**Recall:** The precise definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives.

Recall = True Positives / (True Positives + False Negatives)

**F1-score:** The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

F1 = 2 \* (Precision \* Recall) / (Precision + Recall)

**Support:** The support is the number of occurrences of each class.

**9. Installation of Software and Dependencies**

This section discusses about installing the required software and the dependencies required for the setup of the project, in other words, the environment required for the project.

**9.1 Environment for Python**

* Download the Anaconda navigator that is compatible with your computer from the website <https://www.anaconda.com/download/>
* Open command prompt and execute the following set of commands to setup an environment
  1. conda create --name QAS

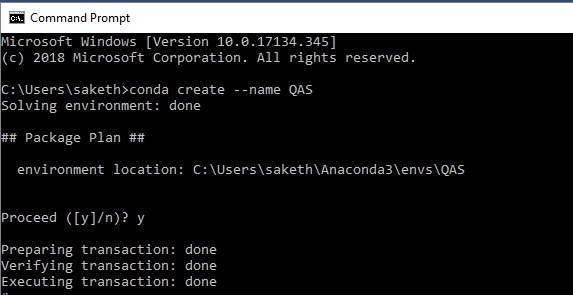


Fig. 9.1: Creating Environment

* 1. activate QAS



Fig 9.2: Activating Environment

* 1. conda install keras

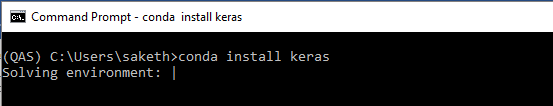


Fig 9.3: Installing Keras

* 1. conda install tensorflow

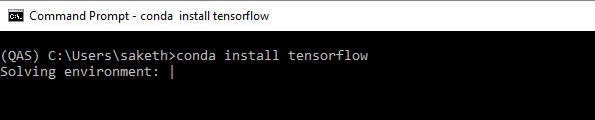


Fig 9.4: Installing TensorFlow

* 1. conda install pillow

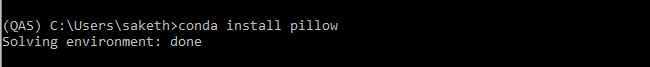


Fig. 9.5: Installing PIL

6. conda install scikit-learn

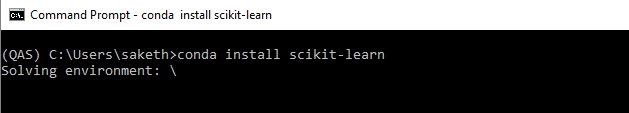


Fig. 9.6 Installing Scikit-learn

While installing the dependencies, whenever conda asks to proceed ([y] / n), type y. To deactivate the environment, enter the command ‘deactivate QAS’ at the command prompt.

**9.2 Installing and setting up Server**

* Download the Xampp server from <https://www.apachefriends.org/index.html> and make sure the software requirements are met beforehand like the architecture of the system and the other software required for Xampp as instructed on Xampp web page.
* After installing the Xampp server, open the Xampp control panel. The Xampp control panel looks like as shown in the fig. 9.7
* Enable the CGI feature in Xampp to run python and PHP scripts by following the instructions in this link <https://www.youtube.com/watch?v=uRHq6v1IDzc>
* Click on the config button in the Apache module in the Xampp control panel and open the php.ini file.

Make the following changes in the php.ini file:

upload\_max\_filesize = 100M

max\_execution\_time = 100

as shown in fig. 9.8 and 9.9, so that there will be no problem when uploading files or executing scripts from the user interface.



Fig. 9.7: Xampp control panel

* Copy all the files in the Project directory to the htdocs directory in the Xampp directory. All the files that are copied into the htdocs directory are the server side scripts that make up the Quality Assessment System.
* Open all the python scripts and change the first line to the path of the environment which was created earlier. For example, change the first line in all python files to #!C:\Users\saketh\Anaconda3\envs\QAS\python.exe as shown in fig. 9.10 and 9.11. Then change all the python environment paths present in the php files in the shell\_exec() function to the same environment path. This allows to run all the python scripts and php scripts properly.

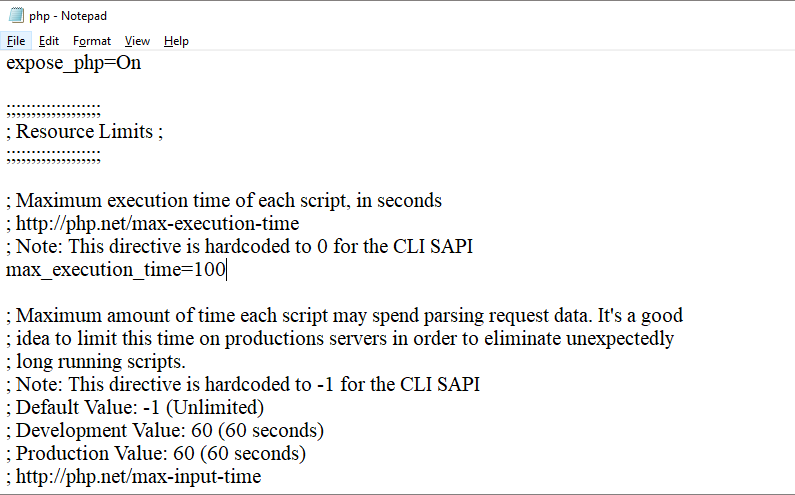


Fig. 9.8: Changing max\_execution\_time

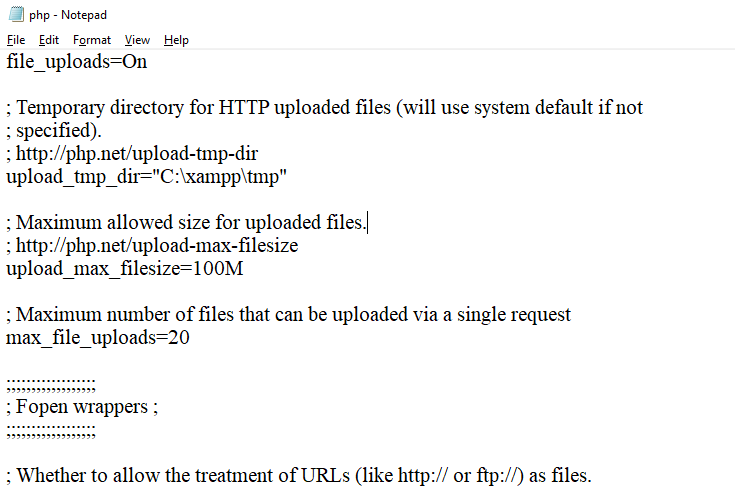


Fig 9.9: Changing upload\_max\_filesize

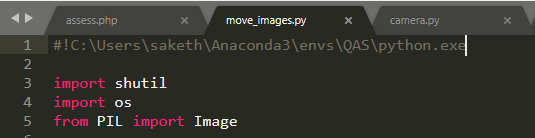


Fig. 9.10: Setting path for python documents

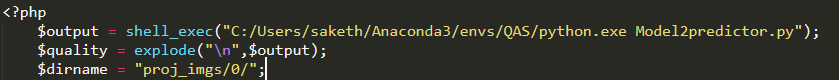


Fig. 9.11: Setting path for python execution in PHP files

* Place a directory MINI in the D: drive of the system. Place two more directories namely dest and src, in which the uploaded files will reside before and after execution of the program. The files in the dest folder will be moved and are pre-processed.
* All the server setup is finished.

**9.3 Executing**

Once the server setup is done, open the Xampp control panel and start the Apache service. Open a web browser and type the URL <https://localhost>. The home page of the user interface will be opened. In the home page, the user can either capture or upload an image of a tomato.

Then the uploaded image will be displayed in the next web page. When the user clicks the assess button then the image is assessed for the quality of the tomato present in the image. The results will be displayed as good or bad in the final page of the web site.

**10. CONCLUSION**

The quality of tomatoes has been assessed by the model with an accuracy of 86.7%. This model can be considered as a fairly good model because out of 41 good tomatoes, it has classified 39 tomatoes correct. Various tweaks and changes to any layers as well as the introduction of new layers can provide completely different results. The current model of the Quality Assessment System is not suitable for implementation in the industry as there are lot of improvements to be done. Few improvements are increasing the data set to provide more accuracy, automating the Assessment system etc.

**11. FUTURE SCOPE**

The accuracy of the project can be improved by designing a more complex neural network. This involves further experimenting with the neural network. One option is to replace all the convolutional neural network with a completely new network. Another possibility is to replace the rectified linear units to exponential linear units to provide better results and performance.

The data set can be increased so that, the accuracy of the Quality Assessment System can be improved. Other possibility is to add images of other kinds of fruits to the training dataset, so that the Quality Assessment System is not limited to only assessing the tomato fruits.

In the near future, the Quality Assessment System can be automated and can be implanted in bots so that they can assess the tomato fruit while they are growing in the controlled conditions of farming.

The model which is used to predict gives only two classes of tomatoes as output, they are good or bad. With more data and complex neural network, if the model is trained, the Quality Assessment System might identify unripe tomatoes as good tomatoes which can be utilised later.

**12. REFERENCES**

<http://cs231n.github.io/convolutional-networks/#add>

<https://www.sciencedirect.com/science/article/pii/S1877050916001861>

<https://keras.io/>

<https://www.tensorflow.org/>

<https://en.wikipedia.org/wiki/Convolutional_neural_network>

<https://conda.io/docs/user-guide/tasks/manage-environments.html>

<https://www.researchgate.net/publication/321475443_Fruit_recognition_from_images_using_deep_learning>

<http://scikit-learn.org/stable/tutorial/basic/tutorial.html>

<https://en.wikipedia.org/wiki/Scikit-learn>

<https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>

<https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f>

<https://www.superdatascience.com/convolutional-neural-networks-cnn-step-3-flattening/>

<https://keras.io/layers/pooling/>

<https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Introduction%20to%20neural%20networks>

<https://en.wikipedia.org/wiki/Python_Imaging_Library>

<https://github.com/tensorflow/tensorflow>

<https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/>

<https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>