PEPPER LEAF DISEASE DETECTION USING IMAGE PROCESSING

A Major Project (U20CSPR02) Report submitted

in partial fulfilment for the award of the Degree of

Bachelor of Technology
in
Computer Science and Engineering
by

K.NAVADEEP	**(U20CS457)**
K.SIVA	**(U20CS489)**
K.SAINATH	**(U20CS492)**
K.OBULAREDDY	**(U20CS499)**

Under the guidance of

Mrs. JENITA CHRISTY, M.Tech.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

BHARATH INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

CHENNAI 600073, TAMILNADU, INDIA

April, 2024

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
BHARATH INSTITUTE OF SCIENCE AND TECHNOLOGY
No.173, Agharam Road, Selaiyur, Chennai - 600 073.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BONAFIDECERTIFICATE

This is to Certify that this Innovation Project Report Titled **"PEPPER LEAF DISEASE DETECTION USING IMAGE PROCESSING"** is the Bonafide Work of **K.Navadeep (U20CS457), K.Siva (U20CS489), K.Sainath (U20CS492)** and **K.Obula Reddy (U20CS499)** of Final Year B.Tech. (CSE) who carried out the Major project work under my supervision Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on basis of which a degree or award conferred on an earlier occasion by any other candidate.

PROJECT GUIDE
Mrs. Jenita Christy
Assistant Professor Department of CSE
BIHER

HEAD OF THE DEPARTMENT
Dr.S. Maruthuperumal
Professor
Department of CSE
BIHER

Submitted for the project Viva-Voce held on ……

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We declare that this Major project report titled **"PEPPER LEAF DISEASE DETECTION USING IMAGE PROCESSING"** submitted in partial fulfillment of the degree of **B. Tech in (Computer Science and Engineering)** is a record of original work carried out by us under the supervision of **Mrs. Jenita Christy**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

K.Navadeep
(U20CS457)

K.Siva
(U20CS489)

K.Sainath
(U20CS492)

K.Obula Reddy
(U20CS499)

Chennai

ACKNOWLEDGMENTS

We express our sincere thanks to our beloved Honorable Chairman **Dr S.Jagathrakshakan, M.P.,** for continuous and constant encouragement in all academic activities.

We express our deepest gratitude to our beloved President **Dr. J. Sundeep Aanand** President **,** and Managing Director **Dr.E.Swetha Sundeep Aanand** Managing Director for providing us the necessary facilities to complete our project.

We take great pleasure in expressing sincere thanks to **Dr. K. Vijaya Baskar Raju** Pro Chancellor , **Dr. M. Sundararajan** Vice Chancellor (i/c), **Dr. S. Bhuminathan** Registrar and **Dr. R. Hariprakash** Additional Registrar**, Dr. M. Sundararaj** Dean Academics for moldings our thoughts to complete our project.

We thank our **Dr. S. Neduncheliyan** Dean, School of Computing for his encouragement and the valuable guidance.

We record indebtedness to our Head, **Dr. S. Maruthuperumal,** Department of Computer Science and Engineering for his immense care and encouragement towards us throughout the course of this project.

We also take this opportunity to express a deep sense of gratitude to our Supervisor **Mrs.Jenita Christy** and our Project Co-ordinator **Dr.Swarna Jyothi** for their cordial support, valuable information and guidance, They helped us in completing this project throughvarious stages.

We thank our department faculty, supporting staff and friends for their help andguidance to complete this project.

K.Nvadeep (U20CS457)
K.Siva (U20CS489)
K.Sainath (U20CS492)
K.Obula Reddy (U20CS499)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

1. RCNN: Regional-based convolutional neural networks
2. CNNs: Convolutional Neural Networks
3. CAD: Computer-aided Design
4. GPU: Graphic Processing Unit
5. CPU: Central Processing Unit
6. HCL: Hardware Compatibility List
7. RAM: Random Access Memory
8. PNG: Portable Network Graphic
9. JPEG: Joint Photographic Experts Group
10. GIF: Graphics Interchange Format
11. CT: Computer Tomography
12. MRI: Magnetic Resonance Imaging
13. FCM: Fuzzy C-means
14. DFDs: Data Flow Diagram
15. PCA: Principal Component Analysis
16. IOT: Internet of Things
17. KNN: K-nearest Neighbours
18. SSADM: Structured Systems Analysis and Design Method
19. UML: Unified Modelling Language.
20. OMG: Object Management Group
21. SGD: Stochastic Gradient Descent
22. QA: Quality Assurance

ABSTRACT

The cultivation of pepper plant life that are scientifically referred to as Genus Capsicum, is rather difficult. Those flowers are prone to numerous sicknesses that directly effects their leaves, resulting in lower crop yields and considerable economic losses. Therefore, the well timed detection of these sicknesses is vital for effective intervention and control strategies. In this paper, we advocate a groundbreaking method to come across pepper leaf diseases the usage of RCNN. RCNN is a complicated deep gaining knowledge of framework that famous promising capabilities in item detection tasks because it successfully merges location thought strategies with convolutional neural networks. With this study, we efficaciously adapt and optimize RCNN for the right identity of number one pepper leaf disorder, particularly bacterial spots.

This paper works on detecting target items within a photo! It seamlessly combines selective search algorithms that suggest ability item regions with convolutional neural networks, finally permitting green and correct identification. By utilising this sturdy model, we intention to beautify the detection of not unusual pepper leaf illnesses, offering valuable insights for disease management and control.

CHAPTER 1

1. INTRODUCTION

But agriculture and farming are treated as a “waste of time” in the modern era. The major reason behind this is plant disease and natural calamities. Natural calamities can be considered as out of our box content whereas plant diseases can be determined and needed precautions can be took for the better tomorrow. Here we are considering the case of Pepper plants. Most of the pepper plant disease can be detected by checking the plant leaves. Usually seen diseases are Damping off disease, Leaf spot, Mosaic virus etc. Manually and detecting will take time and is a difficult task. A machine learning algorithm is used to the leaf and detect whether the leaf is defected or not and if defected, it determines which is the disease. Further remedies will also be provided as part of the output. In classical patterns, tedious tasks such as image processing, feature extraction, and feature classification are now simplified by deep learning. The effects of picture recognition improve. Deep learning has significantly lowered the time and effort required to detect objects. It also aids in real-time decision making and saves agricultural losses. Our paper focuses on disease detection in the leaf and encourages farmers and producers to be more disease-aware. CNN was used to create our model. It's a conventional deep learning model. This approach can extract features from the source image automatically in the case of disease detection. CNN simplifies the recognition problem and saves time because it is an end-to-end structure. The model can be implemented on a Farming sector and can easily monitor pepper plants there. The system collects and check the

matches with the training data to detect if defected. The final output is disease with remedies if defected.

Background:

Plant diseases pose a significant threat to agricultural productivity and food security, with studies estimating that crop losses due to diseases range from 20% to 40% globally. Pepper plants, in particular, are susceptible to various diseases, including fungal, bacterial, and viral infections. Common diseases such as , Damping off disease, Leaf spot, and Mosaic virus can cause significant damage to Pepper plants if not detected and managed promptly. Traditionally, disease detection in plants has relied on manual inspection by agricultural experts, who visually examine plant leaves for symptoms of disease. However, this approach is time-consuming, labor-intensive, and subject to human error, leading to delays in diagnosis and ineffective disease management. With the advent of machine learning and computer vision techniques, there has been a paradigm shift in disease detection methodologies, with automated systems offering the promise of faster, more accurate, and scalable solutions. Deep learning, in particular, has emerged as a powerful tool for image recognition tasks, enabling the development of robust disease detection models capable of analyzing large volumes of image data with unprecedented accuracy. In recent years, CNNs have gained prominence in the field of computer vision, demonstrating state-of-the-art performance in various applications, including object detection, image classification, and segmentation. By leveraging the hierarchical feature learning capabilities of CNNs, researchers have achieved remarkable success in automating disease detection in plant leaves, paving the way for precision agriculture and sustainable farming practices.

Objective:

The primary objective of our study is to develop an automated system for disease detection in Pepper plants using deep learning techniques, specifically CNNs. By harnessing the power of CNNs, we aim to create a robust and scalable solution capable of accurately identifying common diseases such as Damping off disease, Leaf spot, and Mosaic virus in Pepper plant leaves. Our system will streamline the process of disease diagnosis, enabling farmers and producers to detect and manage plant diseases more effectively, thereby minimizing crop losses and improving agricultural productivity. Additionally, our system will provide actionable insights by recommending appropriate remedies for detected diseases, thereby empowering farmers with the knowledge needed to mitigate the impact of diseases on their crops. Overall, our objective is to leverage advanced technology to revolutionize disease management practices in agriculture and foster a culture of disease-awareness among farmers and agricultural stakeholders.

1.1 Software Requirements:

Python:

- Python is an object-oriented, high-level, interpreted, dynamic, multipurpose programming language.
- Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for application development.
- Python's syntax and dynamic typing, along with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas.
- Python supports multiple programming patterns, including object-oriented programming, imperative and functional programming, or procedural styles.
- Python is not intended to work on special areas such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD, etc.
- We don't need to use data types to declare variables because it is dynamically typed. So, we can write `a = 10` to declare an integer value in a variable.
- Python makes development and debugging fast because there is no compilation step included in Python development, and the edit-test-debug cycle is very fast. Python Features:

  Here's a clearer and revised version of the provided information:

  1) Easy to Use:

  Python is known for its simplicity, making it highly accessible to programmers of all levels. Its high-level nature contributes to its user-friendliness.

2) Expressive Language:

Python is renowned for its expressiveness, meaning that its code is easily understandable and readable.

3) Interpreted Language:

Python operates as an interpreted language, executing code line by line. This facilitates easy debugging, making it an ideal choice for beginners.

4) Cross-Platform Language:

Python is platform-independent, running seamlessly on various operating systems such as Windows, Linux, Unix and mac OS, thereby enhancing its portability.

5) Free and Open Source:

Python is freely available for download from www.python.org, and its source code is open to the public. This characteristic makes it an open-source language.

6) Object-Oriented Language:

Python fully supports object-oriented programming, introducing concepts like classes and objects.

7) Extensible:

Python's extensibility allows integration with other languages such as C/C++, enabling code compilation and further expansion within Python projects.

8) Large Standard Library:

Python boasts a vast and diverse standard library, offering a wide range of pre-built modules and functions.

9) GUI Programming:

Python facilitates the development of graphical user interfaces (GUIs), making it suitable for creating visually appealing applications.

10) Integrated:

Python seamlessly integrates with other programming languages like C, C++, and Java, enabling interoperability and extending its functionality.

Python History:

- Python's foundation was laid in the late 1980s.
- Implementation began in December 1989 by Guido van Rossum at CWI in the Netherlands.
- ABC programming language served as a precursor to Python, capable of exception handling and interfacing with the Amoeba operating system.
- Python drew influence from programming languages such as ABC and Modula-3.

Python Applications:

Python finds extensive use across various domains:

1) Console Based Applications:

Python is employed in developing console-based applications like I Python.

2) Audio or Video Based Applications:

Python is handy in multimedia applications, for instance, Tim player and C play.

3) 3D CAD Applications:

Fandango offers comprehensive CAD features using Python.

4) Web Applications:

Python is utilized for developing web-based applications, including Python wiki engines, Po-coo, and Python blog software.

5) Enterprise Applications:

Python can create applications for enterprise use, like Open ERP, Try ton, and Picalo.

6) Image Applications:

Python supports the development of image applications like V Python, Gogh, and Image seek.

Python Example:

Python code is straightforward and easy to execute. Here's a simple Python code snippet that prints "Welcome to Python":

```python
A = "Welcome to Python"
print(A)
```

>>> Welcome to Python

In Python 3.4 and later versions, parentheses are required around the string to print it:

```python
A = ("Welcome to Python Example")
print(A)
```

>>> Welcome to Python Example

1.2 Introduction to Computer Vision:

Computer vision involves using software to analyze visual content, which is crucial for various applications:

- Object Classification: Models classify objects into predefined categories.
- Object Identification: Models recognize specific instances of objects.
- Other tasks include video motion analysis, image segmentation, scene reconstruction, and image restoration.

How Computer Vision Works:

Computer vision operates by interpreting images as a series of pixels, each with its own set of color values. Despite significant advancements, understanding how the human brain processes visual information remains a challenge, leading to uncertainties in mimicking cognitive processes in algorithms.

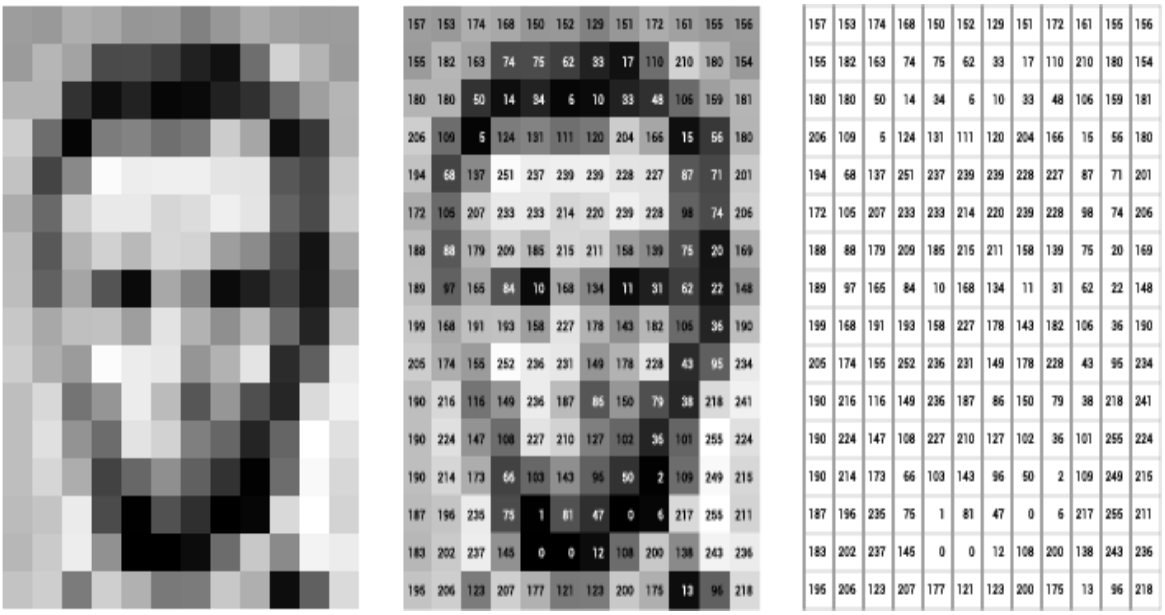

Fig.No.1.2.1:-Computer Vision Works

Source: Open frameworks

An image can be envisioned as a vast grid composed of numerous squares, known as pixels. Each pixel within an image is represented by a numerical value, typically ranging from 0 to 255. In simpler terms, imagine each pixel as a tiny dot on a screen. The series of numbers assigned to each pixel provides the software with information about the image's composition.

In our example, depicting either Abraham Lincoln or a De-mentor, the image is segmented into 12 columns and 16 rows, resulting in a total of 192 input values for the entire image.

As we introduce colour, the complexity increases. Computers interpret colour using a combination of three values: red, green, and blue (RGB), each ranging

from 0 to 255. Therefore, every pixel in a colour image not only retains its position within the grid but also holds three colour values.

For instance, if we were to colorize the image of President Lincoln (or Harry Potter's worst fear), the total count of values would multiply: 12 columns x 16 rows x 3colour values, resulting in a total of 576 numbers to represent the image.

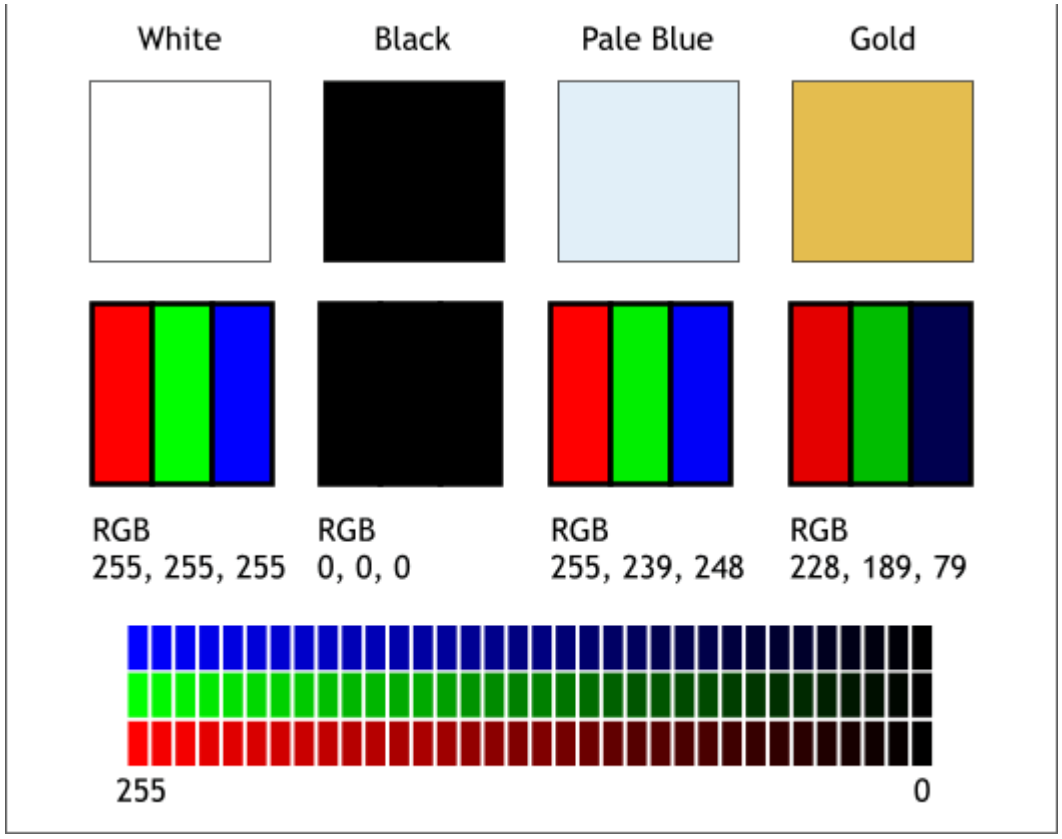

Fig.No.1.2.2:-Creating colours with RGB pixels

Source: Xerox one

To comprehend the computational complexity of images, consider the following breakdown:

- ✓ Each colour value is stored in 8 bits.
- ✓ With 3 colours per pixel (red, green, and blue), each pixel requires 24 bits.
- ✓ For a standard-sized 1024 x 768 image, this equates to almost 19 million bits, roughly 2.36 megabytes of memory.

This substantial memory requirement per image poses challenges for algorithms, especially those involving deep learning. Achieving meaningful accuracy, particularly in deep learning, necessitates tens of thousands of images for training. Even with transfer learning, where insights from pre-trained models are utilized, thousands of images are still required for effective training.

Given the substantial computational power and storage demands for training deep learning models in computer vision, it's evident why advancements in these fields have propelled machine learning forward.

Business Use Cases for Computer Vision:

Computer vision, integrated into major products and services, offers numerous business applications:

- ✓ Google utilizes computer vision in maps to identify street names, businesses, and office buildings.
- ✓ Facebook employs computer vision for facial recognition and photo tagging.
- ✓ Automotive companies like Ford invest in autonomous vehicles, relying on computer vision for analysing video feeds and navigation.

Moreover, computer vision plays a significant role in the medical field, aiding in diagnoses through image processing. Collaborative efforts between tech giants like Google and medical research teams have led to advancements in medical workflows, such as diabetic retinopathy detection.

Computer Vision on Algorithmia:

Algorithmia offers accessible deployment of computer vision applications as scalable micro services. Some available algorithms include:

- ❖ Sal Net: Identifies crucial image components automatically.
- ❖ Nudity Detection: Detects nudity in images.

- ❖ Emotion Recognition: Parses emotions displayed in images.
- ❖ Deep Style: Applies advanced filters to images.
- ❖ Face Recognition: Identifies faces.
- ❖ Image Memorability: Evaluates image memorability.

These algorithms find application in various scenarios, from security camera analysis to content moderation on web platforms.

1.3 Computer Vision Resources:

Several packages and frameworks facilitate computer vision tasks:

- ➢ OpenCV: Designed for computational efficiency, widely adopted for real-time applications.
- ➢ SimpleCV: Open-source framework integrating powerful computer vision libraries like OpenCV.
- ➢ Mahotas: Python library for image processing and computer vision, with over 100 functions.
- ➢ NumPy: Fundamental library for numerical computing in Python, essential for array manipulation and mathematical operations. It is often used alongside other libraries like SciPy and Matplotlib.

Understanding these resources is essential for developing sophisticated computer vision applications, from basic image processing to complex analysis tasks.

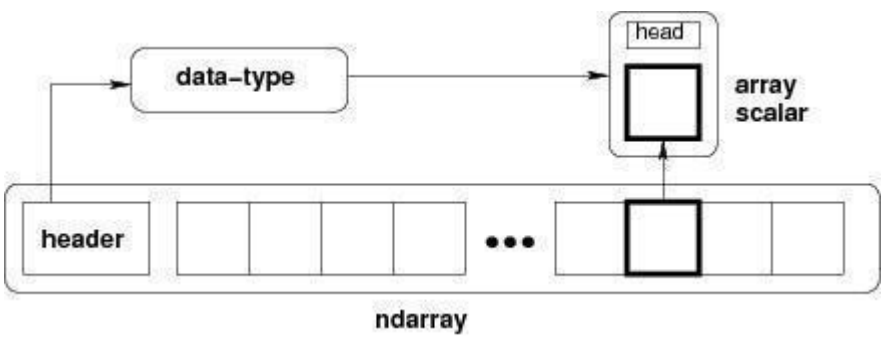

Fig.No.1.3:-Ndarray

An instance of Ndarray class can be constructed by different array creation routines described later in the tutorial. The basic Ndarray is created using an array function in Numpy as follows –

Numpy.Array

An NDArray can be generated from any object exposing an array interface or from methods returning an array.

Imutils:

Imutils offers a range of utility functions for basic image processing tasks, simplifying operations such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images with OpenCV and Python.

Translation:

Translation involves shifting an image along the X or Y direction. In OpenCV, translation requires supplying the (X, Y)-shift as (Tx, Ty) to construct the translation matrix M. Instead of manually constructing M and calling cv2.warpAffine, you can use the translate function in Imutils.

Rotation:

Rotating an image in OpenCV typically involves calling cv2.getRotationMatrix2D and cv2.warpAffine. The rotate function in Imutils streamlines this process, allowing rotation about a specified point.

Resizing:

Resizing an image in OpenCV is done using cv2.resize. Imutils' resize function maintains the aspect ratio and offers keyword arguments for specifying width and height, simplifying the resizing process.

Skeletonization:

Skeletonization constructs the "topological skeleton" of an object in an image. While OpenCV lacks a direct function for this, Imutils provides the skeletonize function to generate the skeleton conveniently.

Displaying with Matplotlib:

In OpenCV's Python bindings, images are represented as NumPy arrays in BGR order. Matplotlib's imshow function expects images in RGB order. To address this, cv2.cvtColor can be used to convert the image, or the opencv2matplotlib function in Imutils can be employed for convenience.

Tensor Flow:

The most famous deep learning library in the world is Google's Tensor Flow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations. To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword the search bar, Google provides a recommendation about what could be the next word. Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

- Researchers
- Data scientists
- Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency. Google does not just have any data; they have the world's most massive computer, so Tensor Flow was built to scale. Tensor Flow is a library developed by the Google Offense and non-offense Team to accelerate machine learning and deep neural network research. It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java. In this tutorial, you will learn

Tensor Flow Architecture:

Tensor flow architecture works in three parts:

1. Pre-processing the data
2. Build the model
3. Train and estimate the model

It is called Tensor flow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input.

The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output. This is why it is called Tensor Flow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Where can Tensor flow run?

Tensor Flow can hardware, and software requirements can be classified into Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop. Run Phase or Inference Phase: Once training is done Tensor Flow can be run on many different platforms. You can run it on Desktop running Windows, mac OS or Linux Cloud as a web service Mobile devices like iOS and Android You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. Tensor Flow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, Tensor Flow can be accessed and controlled by other languages mainly, Python. Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what Tensor Flow is doing.

List of Prominent Algorithms supported by Tensor Flow

- Linear regression:
- Classification :
- Deep learning classification:
- Booster tree regression:
- Boosted tree classification:

1.4 Hardware Requirements:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a HCL, especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Architecture– All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power– The power of the CPU is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored.

Memory– All software, when run, resides in the RAM of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Secondary storage– Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

Display adapter– Software requiring a better than average computer graphics Display like graphics editors and high-end games, often define high-end display adapters in the system requirements.

Peripherals – Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

1) Operating System: Windows Only

2) Processor: i5 and above

3) Ram: 4GB and above

4) Hard Disk: 50 GB

1.5 Digital Image Processing:

The identification of objects in an image and this process would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures. The clever bit is to interpret collections of these shapes as single objects, e.g. cars on a road, boxes on a conveyor belt or cancerous cells on a microscope slide. One reason this is an AI problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system performs these tasks mostly unconsciously but a computer requires skilful programming and lots of processing power to approach human performance. Manipulation of data in the form of an image through several possible techniques. An image is usually interpreted as a two-dimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. An image can be processed optically or digitally with a computer.

Fundamentals of Digital Image:

Image:

An image is a two-dimensional picture, which has a similar appearance to some subject usually a physical object or a person. Image is a two-dimensional, such as a photograph, screen display, and as well as a three-dimensional, such as a statue. They may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. In this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph.

Fig.No.1.5.1:-Colour image to Grey scale Conversion Process

An image is a rectangular grid of pixels. It has a definite height and a definite width counted in pixels. Each pixel is square and has a fixed size on a given display. However different computer monitors may use different sized pixels. The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and colour.

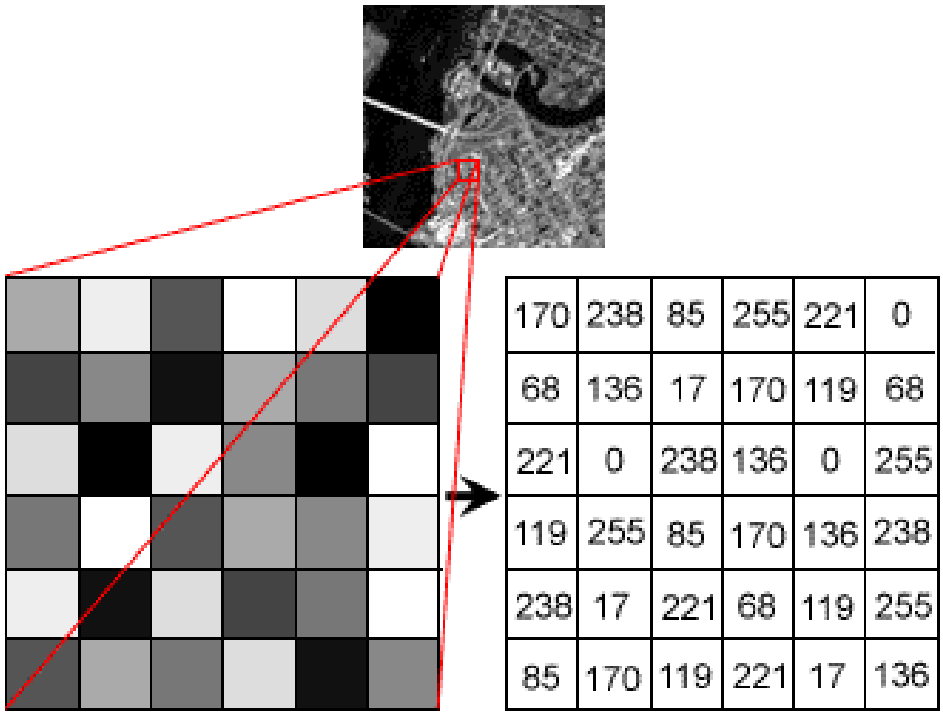

Fig.No.1.5.2:-Grey Scale Image Pixel Value Analysis

Each pixel has a colour. The colour is a 32-bit integer. The first eight bits determine the redness of the pixel, the next eight bits the greenness, the next eight bits the blueness, and the remaining eight bits the transparency of the pixel.

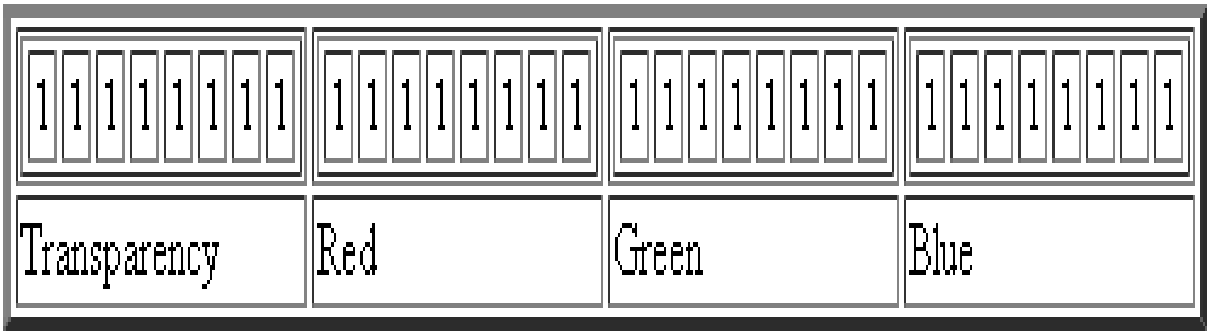

Fig.No.1.5.3:-BIT Transferred for Red, Green and Blue plane

Image File Sizes:

Image file size is expressed as the number of bytes that increases with the number of pixels composing an image, and the colour depth of the pixels. The greater the number of rows and columns, the greater the image resolution, and the larger the file. Also, each pixel of an image increases in size when its colour depth increases, an 8-bit pixel (1 byte) stores 256 colours, a 24-bit pixel (3 bytes) stores 16 million colours, the latter known as true colour. Image compression uses algorithms to decrease the size of a file. High resolution cameras produce large image files, ranging from hundreds of kilobytes to megabytes, per the camera's resolution and the image-storage format capacity. High resolution digital cameras record 12 megapixel (1MP = 1,000,000 pixels / 1 million) images, or more, in true colour. For example, an image recorded by a 12 MP camera; since each pixel uses 3 bytes to record true colour, the uncompressed image would occupy 36,000,000 bytes of memory, a great amount of digital storage for one image, given that cameras must record and store many images to be practical. Faced with large file sizes, both within the camera and a storage disc, image file formats were developed to store such large images.

Image File Formats:

Image file formats are standardized means of organizing and storing images. This entry is about digital image formats used to store photographic and other images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. Including proprietary types, there are hundreds of image file types. The PNG, JPEG, and GIF formats are most often used to display images on the Internet.

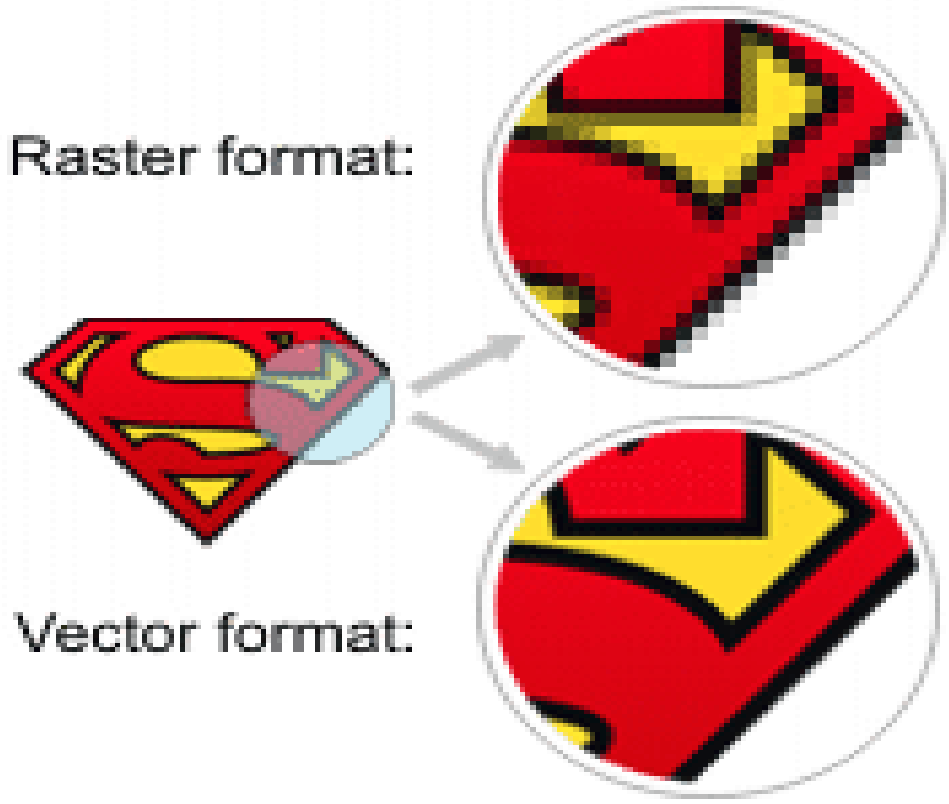

Fig.No.1.5.4:-Horizontal and Vertical Process

In addition to straight image formats, Metafile formats are portable formats which can include both raster and vector information. The metafile format is an intermediate format. Most Windows applications open metafiles and then save them in their own native format.

Image Processing:

Digital image processing, the manipulation of images by computer, is relatively recent development in terms of man's ancient fascination with visual stimuli. In its short history, it has been applied to practically every type of images with varying degree of success. The inherent subjective appeal of pictorial displays attracts perhaps a disproportionate amount of attention from the scientists and also from the layman. Digital image processing like other glamour fields, suffers from myths, miss connections, miss understandings and miss information. It is vast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology.

It is truly multidisciplinary endeavour ploughed with imprecise jargon. Several factor combine to indicate a lively future for digital image processing. A major factor is the declining cost of computer equipment. Several new technological trends promise to further promote digital image processing. These include parallel processing mode practical by low cost microprocessors, and the use of charge coupled devices for digitizing, storage during processing and display and large low cost of image storage arrays.

FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:

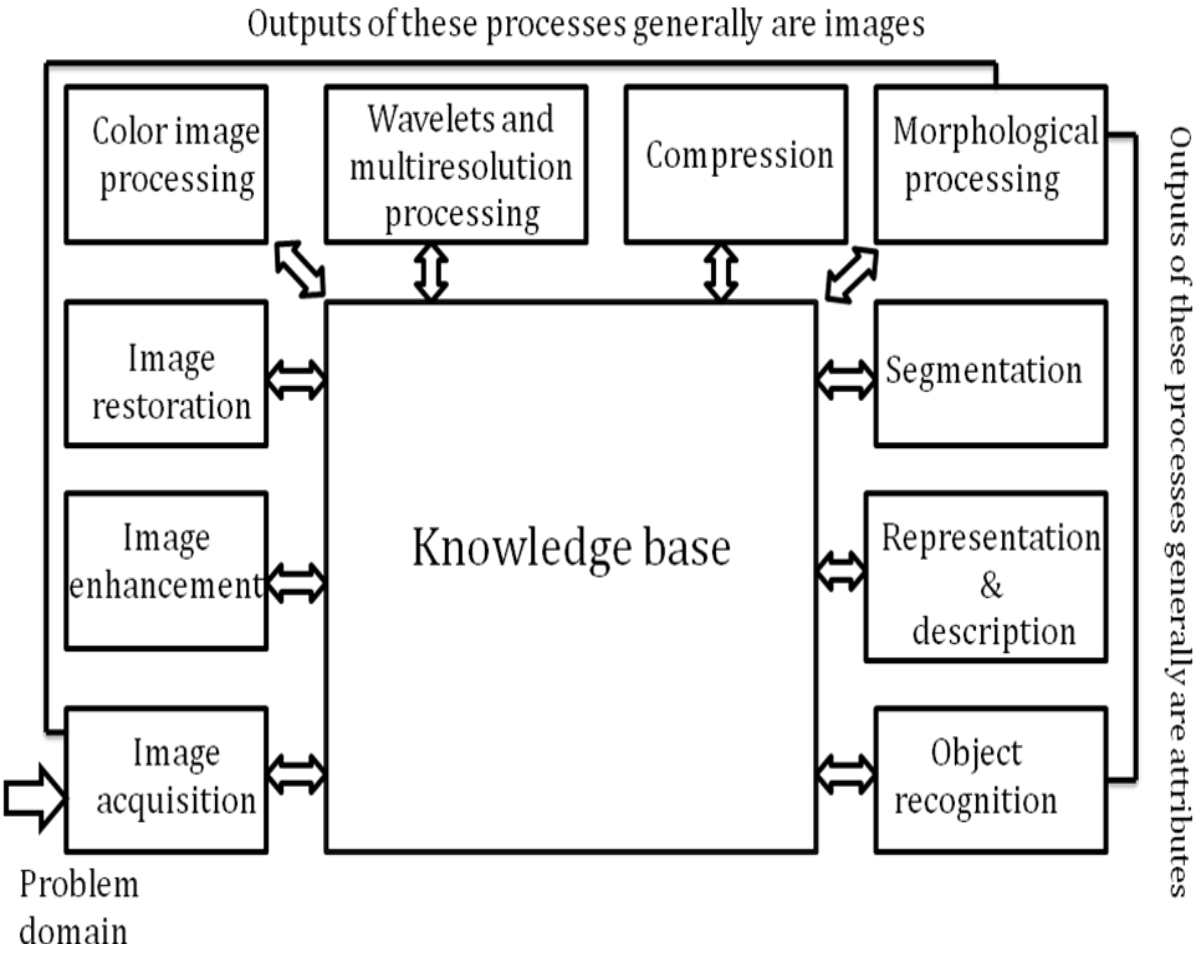

Fig.No.1.5.6:-Basics steps of image Processing

Image Acquisition:

Image Acquisition is to acquire a digital image. To do so requires an image sensor and the capability to digitize the signal produced by the sensor.

The sensor could be monochrome or colour TV camera that produces an entire image of the problem domain every 1/30 sec. the image sensor could also be line scan camera that produces a single image line at a time. In this case, the objects motion past the line.

Fig.No.1.5.7:-Digital camera

Scanner produces a two-dimensional image. If the output of the camera or other imaging sensor is not in digital form, an-log to digital converter digitizes it. The nature of the sensor and the image it produces are determined by the application.

Fig.No.1.5.8:-Mobile based Camera

Image Enhancement:

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interesting an image. A familiar example of enhancement is when we increase the contrast of an image because "it looks better." It is important to keep in mind that enhancement is a very subjective area of image processing.

Fig: Image enhancement process for Grey Scale Image and Colour Image using Histogram Bits

Image restoration:

Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.

Fig.No.1.5.9:-Noise Image & Image Enhancement

Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a "good" enhancement result. For example, contrast stretching is considered an enhancement technique because it is based primarily on the pleasing aspects it might present to the viewer, whereas removal of image blur by applying a de-blurring function is considered a restoration technique.

Colour image processing:

The use of colour in image processing is motivated by two principal factors. First, colour is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of colour shades and intensities, compared to about only two dozen shades of grey. This second factor is particularly important in manual image analysis.

Fig.No.1.5.10:-grey Scale Image & Colour Image

Segmentation:

Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually.

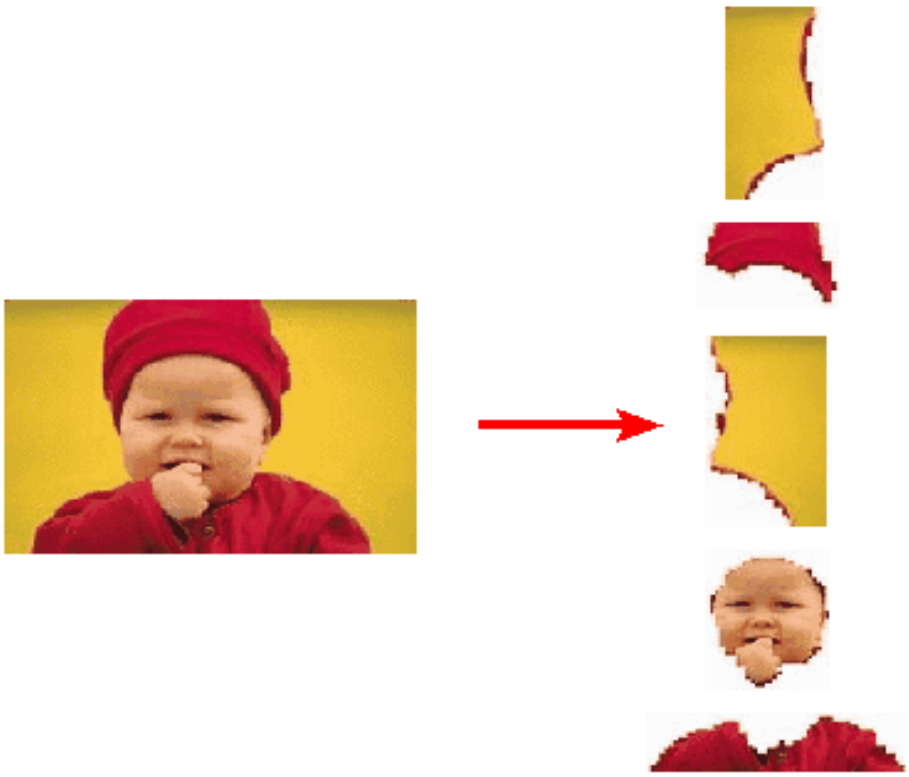

Fig.No.1.5.11:-Image Segment Process

On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

Digital image is defined as a two dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude off at any pair of coordinates (x, y) is called intensity or grey level of the image at that point. The field of digital image processing refers to processing digital images by means of a digital computer. The digital image is composed of a finite number of elements, each of which has a particular location and value. The elements are referred to as picture elements, image elements, and pixels. Pixel is the term most widely used.

Image Compression

Digital Image compression addresses the problem of reducing the amount of data required to represent a digital image. The underlying basis of the reduction process is removal of redundant data. From the mathematical viewpoint, this amounts to transforming a 2D pixel array into a statically uncorrelated data set. The data redundancy is not an abstract concept but a mathematically quantifiable entity. If n1 and n2 denote the number of information-carrying units in two data sets that represent the same information, the relative data redundancy R_D [2] of the first data set (the one characterized by n1) can be defined as,

$$R_D = 1 - \frac{1}{C_R}$$

Where C_R called as compression ratio [2]. It is defined as

$$C_R = \frac{n1}{n2}$$

In image compression, three basic data redundancies can be identified and exploited: Coding redundancy, inter pixel redundancy. Image compression is achieved when one or more of these redundancies are reduced or eliminated. The image compression is mainly used for image transmission and storage. Image transmission applications are in broadcast television; remote sensing via satellite, air-craft, radar, or sonar; teleconferencing; computer communications; and facsimile transmission. Image storage is required most commonly for educational and business documents, medical images that arise in CT, MRI and digital radiology, motion pictures, satellite images, weather maps, geological surveys, and so on.

Image Compression Model

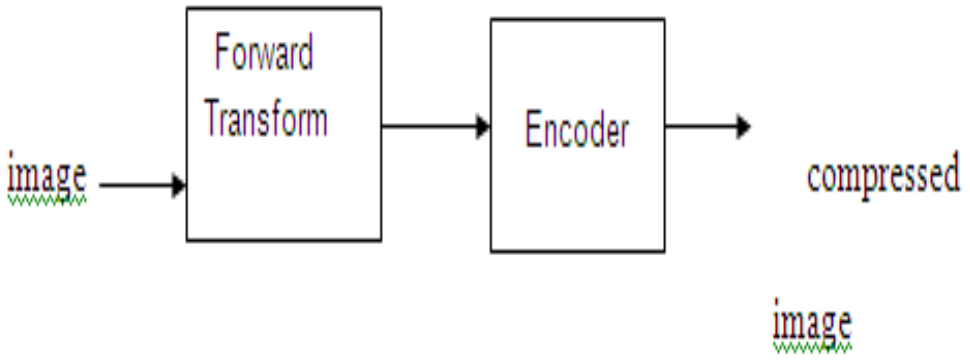

Fig.No.1.5.12:-Block Diagram of Image compression

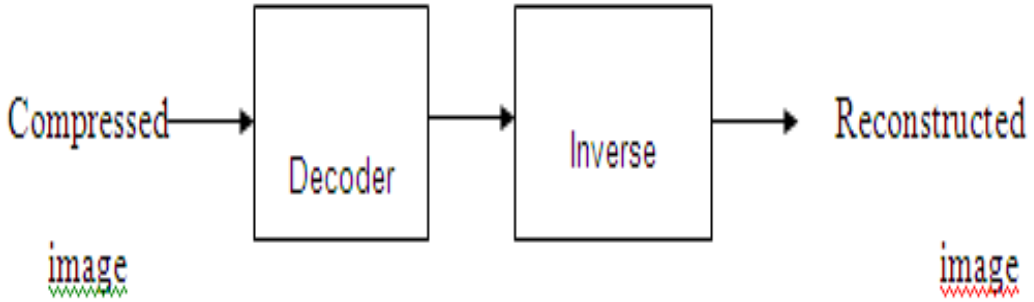

Fig.No.1.5.13:-Decompression Process for Image

Image Compression Types:

There are two types' image compression techniques.

1. Lossy Image compression

2. Lossless Image compression

Compression ratio:

$$compression\ ratio = \frac{B_0}{B_1}$$

B_0 – number of bits before compression
B_1 – number of bits after compression

Lossy Image compression:

Lossy compression provides higher levels of data reduction but result in a less than perfect reproduction of the original image. It provides high compression ratio. Loss image compression is useful in applications such as broadcast television, videoconferencing, and facsimile transmission, in which a certain amount of error is an acceptable trade-off for increased compression performance. Originally, PGF has been designed to quickly and progressively decode loss compressed aerial images. A lossy compression mode has been preferred, because in an application like a terrain explorer texture data (e.g., aerial Ortho-photos) is usually mid-mapped filtered and therefore loss mapped onto the terrain surface. In addition, decoding loss compressed images is usually faster than decoding lossless compressed images.

In the next test series we evaluate the loss compression efficiency of PGF. One of the best competitors in this area is for sure JPEG 2000. Since JPEG 2000 has two different filters, we used the one with the better trade-off between compression efficiency and runtime. On our machine the 5/3 filter set has a better trade-off than the other. However, JPEG 2000 has in both cases a remarkable good compression efficiency for very high compression ratios but also a very poor encoding and decoding speed. The other competitor is JPEG. JPEG is one of the most popular image file formats.

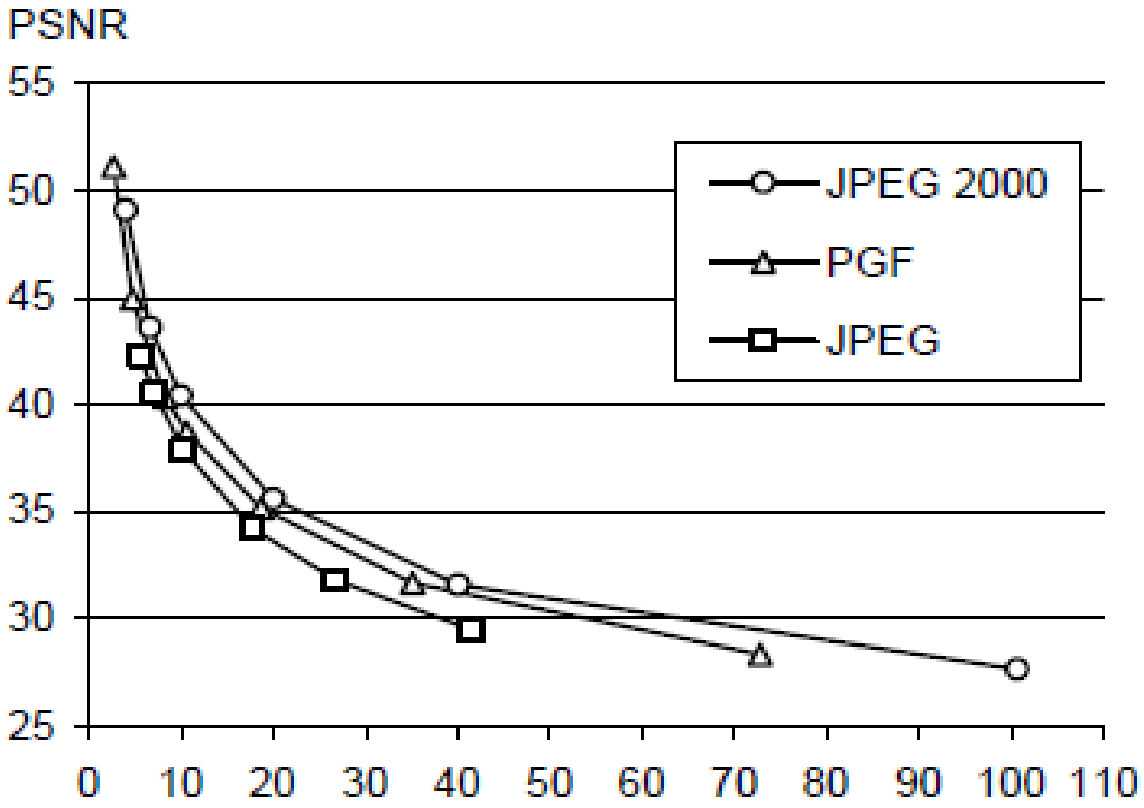

Fig.No.1.5.14:-PSNR of lossy compression ratio

It is very fast and has a reasonably good compression efficiency for a wide range of compression ratios. The drawbacks of JPEG are the missing lossless compression and the often missing progressive decoding. Fig. 4 depicts the average rate-distortion behaviour for the images in the Kodak test set when fixed (i.e., non-progressive) loss compression is used. The PSNR of PGF is on average 3% smaller than the PSNR of JPEG 2000, but 3% better than JPEG.

These results are also qualitative valid for our PGF test set and they are characteristic for aerial Ortho-photos and natural images. Because of the design of PGF we already know that PGF does not reach the compression efficiency of JPEG 2000. However, we are interested in the trade-off between compression efficiency and runtime.

To report this trade-off we show in Table 4 a comparison between JPEG 2000 and PGF and in Fig. 5 (on page 8) we show for the same test series as in Fig. 4 the corresponding average decoding times in relation to compression ratios. Table 4 contains for seven different compression ratios (mean values over the compression ratios of the eight images of the Kodak test set) the corresponding average encoding and decoding times in relation to the average PSNR values. In case of PGF the encoding time is always slightly longer than the corresponding decoding time. The reason for that is that the actual encoding phase (cf. Subsection 2.4.2) takes slightly longer than the corresponding decoding phase. For six of seven ratios the PSNR difference between JPEG 2000 and PGF is within 3% of the PSNR of JPEG 2000. Only in the first row is the difference larger (21%), but because a PSNR of 50 corresponds to an almost perfect image quality the large PSNR difference corresponds with an almost undiscoverable visual difference. The price they pay in JPEG 2000 for the 3% more PSNR is very high. The creation of a PGF is five to twenty times faster than the creation of a corresponding JPEG 2000 file, and the decoding of the created PGF is still five to ten times faster than the decoding of the JPEG 2000 file. This gain in speed is remarkable, especially in areas where time is more important than quality, maybe for instance in real-time computation.

	JPEG 2000 5/3			PGF		
Ratio	enc	dec	PSNR	enc	dec	PSNR
2.7	1.86	1.35	64.07	0.34	0.27	51.10
4.8	1.75	1.14	47.08	0.27	0.21	44.95
8.3	1.68	1.02	41.98	0.22	0.18	40.39
10.7	1.68	0.98	39.95	0.14	0.13	38.73
18.7	1.61	0.92	36.05	0.12	0.11	35.18
35.1	1.57	0.87	32.26	0.10	0.09	31.67
72.9	1.54	0.85	28.86	0.08	0.08	28.37

Table No.1.5.1:-Trade-off between quality and speed for the Kodak test set

In Table No.1.5.1 we see that the price we pay in PGF for the 3% more PSNR than JPEG is low: for small compression ratios (< 9) decoding in PGF takes two times longer than JPEG and for higher compression ratios (> 30) it takes only ten percent longer than JPEG. These test results are characteristic for both natural images and aerial Ortho-photos. Again, in the third test series we only use the 'Lena' image. We run our loss coder with six different quantization parameters and measure the PSNR in relation to the resulting compression ratios. The results (ratio: PSNR) are:

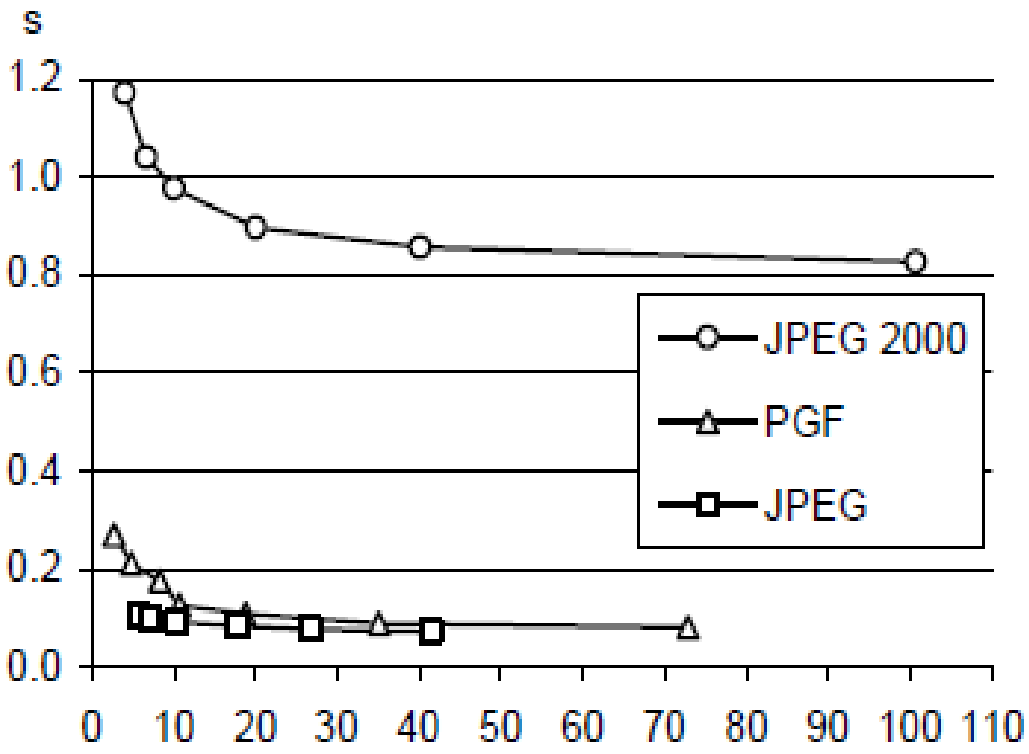

Fig.No.1.5.15:-Decoding time in relation to compression ratio

Lossless Image compression:

Lossless Image compression is the only acceptable amount of data reduction. It provides low compression ratio while compared to lossy.

In Lossless Image compression techniques are composed of two relatively independent operations: (1) devising an alternative representation of the image in which it inter pixel redundancies are reduced and (2) coding the representation to eliminate coding redundancies.

Lossless Image compression is useful in applications such as medical imaginary, business documents and satellite images. Table 2 summarizes the lossless compression efficiency and Table 3 the coding times of the PGF test set. For WinZip we only provide average runtime values, because of missing source code we have to use an interactive testing procedure with runtimes measured by hand. All other values are measured in batch mode.

	WinZip	JPEG-LS	JPEG 2000	PNG	PGF
aerial	1.352	2.073	2.383	1.944	2.314
compound	12.451	6.802	6.068	13.292	4.885
hibiscus	1.816	2.200	2.822	2.087	2.538
houses	1.241	1.518	2.155	1.500	1.965
logo	47.128	16.280	12.959	50.676	10.302
redbrush	2.433	4.041	4.494	3.564	3.931
woman	1.577	1.920	2.564	1.858	2.556
average	**9.71**	**4.98**	**4.78**	**10.70**	**4.07**

Table No.1.5.2:- Lossless Compression ratios of the PGF test set

In Table No.1.5.2 it can be seen that in almost all cases the best compression ratio is obtained by JPEG 2000, followed by PGF, JPEG-LS, and PNG. This result is different to the result in [SEA+00], where the best performance for a similar test set has been reported for JPEG-LS.

PGF performs between 0.5% (woman) and 21.3% (logo) worse than JPEG 2000. On average it is almost 15% worse. The two exceptions to the general trend are the ‘compound’ and the ‘logo’ images. Both images contain for the most part black text on a white background. For this type of images, JPEG-LS and in particular WinZip and PNG provide much larger compression ratios. However, in average PNG performs the best, which is also reported in [SEA+00].

These results show, that as far as lossless compression is concerned, PGF performs reasonably well on natural and aerial images. In specific types of images such as ‘compound’ and ‘logo’ PGF is outperformed by far in PNG.

	WinZip		JPEG-LS		JPEG 2000		PNG		PGF	
	enc	dec	enc	dec	enc	dec	enc	dec	enc	dec
a			1.11	0.80	5.31	4.87	3.70	0.19	0.99	0.77
c			1.61	0.38	3.46	3.06	2.95	0.18	0.95	0.80
hi			0.69	0.30	1.45	1.29	1.77	0.10	0.35	0.27
ho			0.65	0.30	1.62	1.47	0.85	0.11	0.41	0.32
l			0.09	0.02	0.26	0.21	0.16	0.01	0.07	0.06
r			0.65	0.44	4.29	4.01	3.61	0.16	0.66	0.59
w			0.39	0.30	1.76	1.63	1.08	0.08	0.35	0.27
av	1.14	0.37	0.74	0.36	2.59	2.36	2.02	0.12	0.54	0.44

Table No.1.5.3:-Runtime of lossless compression of the PGF test set

Table No.1.5.3 shows the encoding and decoding times (measured in seconds) for the same algorithms and images as in Table 2. JPEG 2000 and PGF are both symmetric algorithms, while WinZip, JPEG-LS and in particular PNG are asymmetric with a clearly shorter decoding than encoding time.

JPEG 2000, the slowest in encoding and decoding, takes more than four times longer than PGF. This speed gain is due to the simpler coding phase of PGF. JPEG-LS is slightly slower than PGF during encoding, but slightly faster in decoding images.

WinZip and PNG decode even more faster than JPEG-LS, but their encoding times are also worse. PGF seems to be the best compromise between encoding and decoding times.

Our PGF test set clearly shows that PGF in lossless mode is best suited for natural images and aerial Ortho-photos. PGF is the only algorithm that encodes the three Mega Byte large aerial Ortho-photo in less than second without a real loss of compression efficiency. For this particular image the efficiency loss is less than three percent compared to the best. These results should be underlined with our second test set, the Kodak test set.

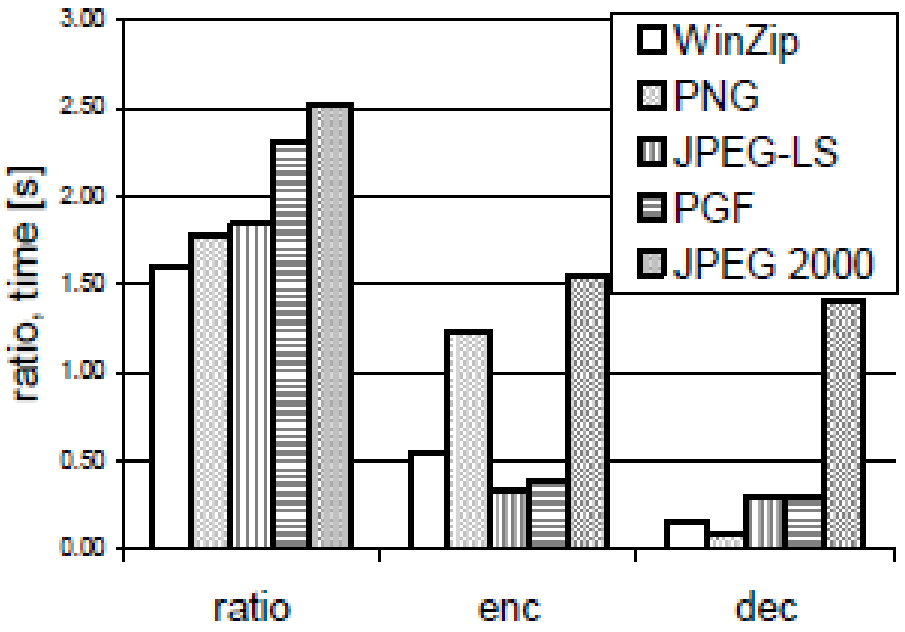

Fig.No.1.5.16:-Lossless compression results

Fig. 3 shows the averages of the compression ratios, encoding, and decoding times over all eight images. JPEG 2000 shows in this test set the best compression efficiency followed by PGF, JPEG-LS, PNG, and WinZip. In average PGF is eight percent worse than JPEG 2000. The fact that JPEG 2000 has a better lossless compression ratio than PGF does not surprise, because JPEG 2000 is more quality driven than PGF.

However, it is remarkable that PGF is clearly better than JPEG-LS (+21%) and PNG (+23%) for natural images. JPEG-LS shows in the Kodak test set also a symmetric encoding and decoding time behaviour. It is encoding and decoding times are almost equal to PGF. Only PNG and WinZip can faster decode than PGF, but they also take longer than PGF to encode.

If both compression efficiency and runtime is important, then PGF is clearly the best of the tested algorithms for lossless compression of natural images and aerial Ortho-photos. In the third test we perform our lossless coder on the ‘Lena’ image.

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512 × 512 or roughly 250,000 pixels, although much larger images are becoming common. Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighbouring pixels in the input image determine the value of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

Correspondingly, these combinations attempt to strike a winning trade off: be flexible and hence bring tolerance toward intra class variation, while also being discriminative enough to be robust to background clutter and interclass similarity. An important feature of our contour-based recognition approach is that it affords us substantial flexibility to incorporate additional image information. Specifically, we extend the contour-based recognition method and propose a new hybrid recognition method which exploits shape tokens and SIFT features as recognition cues. Shape-tokens and SIFT features are largely orthogonal, where the former corresponds to shape boundaries and the latter to sparse salient image patches. Here, each learned combination can comprise features that are either 1) purely shape-tokens, 2) purely SIFT features, or 3) a mixture of shape-tokens and SIFT features. The number and types of features to be combined together are learned automatically from training images, and represent the more discriminative ones based on the training set. Consequently, by imparting these two degrees of variability (in both the number and the types of features) to a combination, we empower it with even greater flexibility and discriminative potential. A shorter version of this paper appeared in [9].

Classification of Images:

There are 3 types of images used in Digital Image Processing. They are

- Binary Image
- Grey Scale Image
- Colour Image

Binary Image:

A binary image is a digital image that has only two possible values for each pixel. Typically the two colours used for a binary image are black and white though any two colours can be used. The colour used for the object(s) in the image is the foreground colour while the rest of the image is the background colour.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images

Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, and dithering. Some input/output devices, such as laser printers, fax machines, and bi-level computer displays, can only handle bi-level images

Grey Scale Image:

A grayscale Image is digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of grey (0-255), varying from black (0) at the weakest intensity to white (255) at the strongest.

Grayscale images are distinct from one-bit black-and-white images, which in the context of computer imaging are images with only the two colours, black, and white (also called bi-level or binary images). Grayscale images have many shades of grey in between. Grayscale images are also called monochromatic, denoting the absence of any chromatic variation.

Grayscale images are often the result of measuring the intensity of light at the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full colour image; see the section about converting to grayscale.

Colour Image:

A (digital) colour image is a digital image that includes colour information for each pixel. Each pixel has a particular value which determines it appearing colour. This value is qualified by three numbers giving the decomposition of the colour in the three primary colours Red, Green and Blue. Any colour visible to human eye can be represented this way. The decomposition of a colour in the three primary colours is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255).

In other words, an image is an enormous two-dimensional array of colour values, pixels, each of them coded on 3 bytes, representing the three primary colours. This allows the image to contain a total of 256x256x256 = 16.8 million different colours. This technique is also known as RGB encoding, and is specifically adapted to human vision

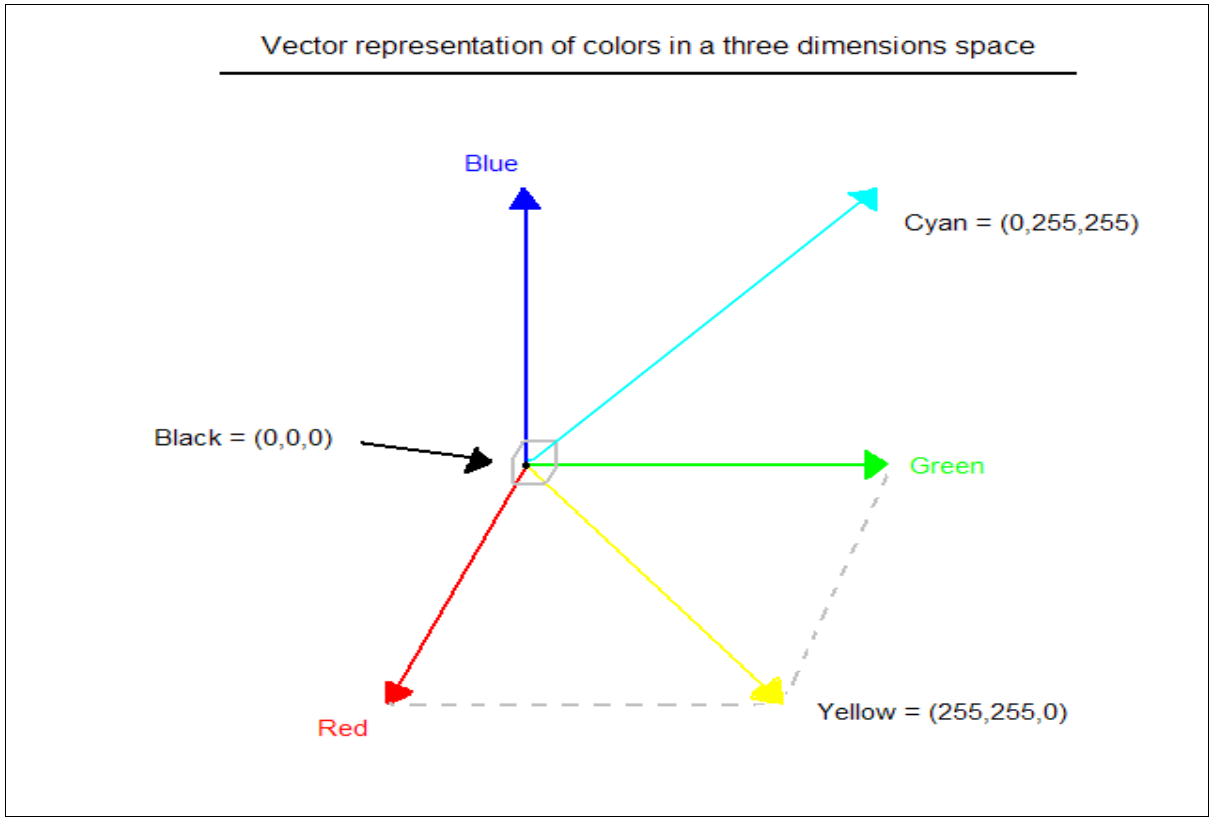

Fig.No.1.5.17:-Hue Saturation Process of RGB SCALE Image

From the above figure, colours are coded on three bytes representing their decomposition on the three primary colours. It sounds obvious to a mathematician to immediately interpret colours as vectors in a three dimension space where each axis stands for one of the primary colours. Therefore we will benefit of most of the geometric mathematical concepts to deal with our colours, such as norms, scalar product, projection, rotation or distance.

1.6 Feasibility Study:

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses. It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

The development of a computer based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly. Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

A feasibility study is a preliminary study undertaken to determine and document a project's viability. The term feasibility study is also used to refer to the resulting document. These results of this study are used to make a decision whether to proceed with the project, or table it. If it indeed leads to a project being approved, it will - before the real work of the proposed project starts - be used to ascertain the likelihood of the project's success. It is an analysis of possible alternative solutions to a problem and a recommendation on the best alternative. It, for example, can decide whether an order processing be carried out by a new system more efficiently than the previous one.

A feasibility study could be used to test a proposal for new system, which could be used because:

- The current system may no longer carry its purpose,
- Technological advancement may have rendered the current system obsolete,
- The business is expanding, allowing it to cope with extra work load,

- Customers are complaining about the speed and quality of work the business provides,
- A feasibility study should examine three main areas:
- Market issues
- Technical and organizational requirements
- Financial overview

Within a feasibility study, seven areas must be reviewed, including those of are:

1. Needs Analysis,
2. Economics,
3. Technical,
4. Schedule,
5. Organizational,
6. Cultural, and
7. Legal.

Needs analysis:

A needs analysis should be the first undertaking of a feasibility study as it clearly defines the project outline and the clients' requirements. Once these questions have been answered the person/s undertaking the feasibility study will have outlined the project needs definition. The following questions need to be asked to define the project needs definition: What is the end deliverable? What purpose will it serve? What are the environmental effects? What are the rules and regulations? What standards will we be measured against? What are the quality requirements? What is the minimal quality requirements allowed? What sustainability can we expect? What carry over work can we expect? What are the penalty clauses? How much do we need to outsource? How much do we need to in source?

Technical Feasibility Study:

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on an outline design of system requirements in terms of Input, Output, Fields, Programs, and Procedures. This can be qualified in terms of volumes of data, trends, frequency of updating, etc. In order to give an introduction to the technical system.

Cultural Feasibility Study:

In this stage, the project's alternatives are evaluated for their impact on the local and general culture. For example, environmental factors need to be considered.

CHAPTER 2

2. LITERATURE SURVEY

In recent years, there has been a growing interest in leveraging machine learning and image processing techniques for the detection and diagnosis of plant diseases. The following literature survey provides an overview of several studies that focus on detecting diseases on plant leaves using various methodologies and technologies.

1. Detection of Apple Leaf Diseases using Machine Learning Techniques:

Midhun P Mathew et al. [1] proposed a method for detecting diseases on apple leaves using machine learning techniques. The study highlights the significance of addressing plant diseases as a major factor contributing to agricultural failures. By utilizing computer vision and deep learning advancements, the researchers aimed to provide highly accurate disease detection in real-time scenarios. They compared different models and found that YOLO (You Only Look Once) demonstrated superior performance in disease detection. The integration of deep learning into modern farming practices holds promise for enhancing disease management strategies.

2. Tracing Plant Defects through Leaf Image Analysis:

Abirami Devaraj et al. [2] introduced a concept for tracing plant defects by analyzing images of plant leaves. The study outlines a multi-step process for illness detection, including image loading, preprocessing, segmentation, feature extraction, and classification. By leveraging image processing techniques, the researchers demonstrated the feasibility of diagnosing plant diseases based on leaf images. This approach offers a non-invasive and efficient method for detecting and classifying diseases in agricultural settings.

3. Early Detection of Tomato Plant Leaf Diseases using Image Processing Techniques:

Surampalli Ashok et al. [3] proposed a method for the early detection of diseases affecting tomato plant leaves. The study focuses on leveraging image processing techniques, including image segmentation, clustering, and open-source algorithms, to identify leaf diseases. By employing these techniques, the researchers aimed to develop a reliable, safe, and accurate system for detecting diseases specific to tomato plants. Early detection of diseases can help farmers implement timely interventions to prevent crop losses and ensure food security.

4. Identification of Leaf Spot Disease using Image Processing and Neural Networks:

R Anand and Ch Usha Kumari et al. [4] presented a system for identifying leaf spot disease using image processing techniques. The study proposed a four-stage process for disease identification, including picture acquisition, image segmentation, feature extraction, and classification. The researchers utilized the K-means clustering approach for image segmentation and extracted features from the disease-affected clusters. They employed a neural network classifier to classify the leaf spots based on the extracted features. This approach demonstrates the potential of combining image processing and machine learning techniques for automated disease diagnosis in plants.

5. Novel Approaches for Disease Detection in Crop Plants:

In addition to the above studies, several other researchers have explored novel approaches for disease detection in crop plants. These approaches include the use of hyperspectral imaging, unmanned aerial vehicles (UAVs) for remote sensing, and smartphone-based applications for on-the-spot disease diagnosis. By leveraging advancements in technology and interdisciplinary research collaborations, scientists aim to develop efficient and scalable solutions for managing plant diseases and ensuring global food security.

6. Challenges and Future Directions:

Despite the progress made in plant disease detection, several challenges remain, including the need for large and diverse datasets, robust algorithms for feature extraction, and scalable deployment of detection systems in field conditions. Future research directions may include the integration of advanced sensors, IOT devices, and artificial intelligence algorithms for real-time monitoring and management of plant diseases. Collaboration between researchers, farmers, and policymakers is essential to address these challenges and develop sustainable solutions for agricultural disease management.

In conclusion, the literature survey highlights the significant progress made in the field of plant disease detection using machine learning and image processing techniques. By addressing key challenges and exploring innovative approaches, researchers aim to develop effective tools and technologies for safeguarding global crop production and ensuring food security for future generations.

CHAPTER 3

3. EXISTING SYSTEM

Leaf shape description is a critical aspect of leaf identification, serving as the foundation for distinguishing between different plant species. Over the years, researchers have explored various methods for characterizing leaf morphology, aiming to develop robust classification systems capable of accurately identifying leaves based on their visual features. However, despite significant advancements in image processing and machine learning techniques, challenges persist in achieving precise and efficient leaf classification. In this section, we delve into the existing system of leaf classification, focusing on the FCM clustering algorithm, its history, general description, algorithmic intricacies, and associated disadvantages.

3.1 History of Fuzzy C-means Clustering:

The Fuzzy C-means clustering algorithm, commonly referred to as FCM, traces its origins back to the early 1970s when J.C. Dunn first introduced the concept of fuzzy clustering. Dunn's pioneering work laid the foundation for subsequent advancements in the field, paving the way for J.C. Bezdek's seminal contributions to FCM in 1981. Bezdek's enhancements to the algorithm solidified its position as a prominent technique for pattern recognition and data clustering.

General Description of FCM:

The FCM algorithm shares similarities with the well-known k-means clustering method, albeit with some distinct characteristics. Like k-means, FCM operates by iteratively partitioning a dataset into a predefined number of clusters. However, unlike k-means, which assigns data points to clusters based on hard assignments where each data point is assigned membership grades representing its degree of belongingness to each cluster.

Algorithmic Workflow:

The workflow of the FCM algorithm can be summarized into several key steps:

- Initialization: Begin by specifying the desired number of clusters and initializing the cluster centroids and membership grades.
- Membership Grade Update: Compute the membership grades for each data point based on its distance from the cluster centroids.
- Centroid Update: Update the cluster centroids based on the weighted average of the data points, with weights determined by their membership grades.
- Convergence Check: Repeat steps 2 and 3 iteratively until convergence is achieved, i.e., until the membership grades and centroids stabilize or the algorithm reaches a predefined convergence criterion.

Centroid Calculation:

At the heart of the FCM algorithm lies the computation of cluster centroids, which serve as representative points for each cluster. The centroid of a cluster is calculated as the weighted mean of the data points belonging to that cluster. The weighting factor is determined by the membership grades assigned to each data point, with higher membership grades indicating stronger association with the cluster.

Principal Component Analysis:

In addition to FCM clustering, PCA is often employed as a pre-processing step in leaf classification tasks. PCA is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional space while preserving the essential variance in the data. By reducing the dimensionality of the feature space, PCA helps simplify the classification task and improve computational efficiency.

K-nearest Neighbours Classifier:

Another commonly used method in leaf classification is the KNN classifier. KNN is a simple yet effective machine learning algorithm used for both classification and regression tasks. In leaf classification, KNN works by assigning a class label to a given leaf based on the class labels of its nearest neighbours in the feature space. The choice of the number of neighbours (K) is a crucial parameter that affects the classifier's performance.

3.2 Disadvantages of FCM:

While FCM clustering offers several advantages, such as flexibility and robustness, it is not without its limitations. Some of the key disadvantages of the FCM algorithm include:

- High Computational Load: FCM can be computationally intensive, particularly when dealing with large datasets or high-dimensional feature spaces. The iterative nature of the algorithm requires multiple iterations to converge, leading to increased computational time and resource consumption.

- Poor Discriminatory Power: In certain scenarios, FCM may struggle to effectively differentiate between clusters with overlapping distributions or complex geometric shapes. This can result in suboptimal clustering performance and reduced discriminative ability.

- Sensitivity to Initialization: The performance of FCM clustering is sensitive to the initial selection of cluster centroids and membership grades. Poor initialization can lead to convergence to suboptimal solutions or premature convergence, impacting the quality of the clustering results.

- Dependency on Hyper parameters: FCM requires the specification of hyper parameters such as the fuzziness parameter (m) and the convergence threshold. The choice of these parameters can significantly influence the clustering outcomes and may require manual tuning or optimization.

- Limited Scalability: While FCM is suitable for small to moderate-sized datasets, it may face scalability issues when applied to large-scale datasets with millions of data points or high-dimensional feature spaces.

Future Directions and Research Opportunities:

Despite its drawbacks, FCM clustering remains a valuable tool in the field of pattern recognition and data mining. However, addressing the aforementioned limitations and challenges requires further research and innovation. Future directions for improving FCM clustering and enhancing its applicability in leaf classification and other domains may include:

- ✓ Algorithmic Enhancements: Developing novel variants of the FCM algorithm that address specific limitations such as computational efficiency, robustness to initialization, and scalability.

- ✓ Integration with Deep Learning: Exploring the integration of FCM clustering with deep learning techniques such as neural networks and CNNs to leverage the power of hierarchical feature representations and end-to-end learning.

- ✓ Hybrid Approaches: Investigating hybrid clustering approaches that combine the strengths of FCM with other clustering algorithms or machine learning methods to overcome individual limitations and achieve improved clustering performance.

- ✓ Domain-Specific Applications: Tailoring FCM clustering algorithms and methodologies to suit specific application domains, such as leaf classification, medical image analysis, and remote sensing, by incorporating domain knowledge and expertise.

- ✓ Evaluation and Benchmarking: Conducting comprehensive evaluations and benchmarking studies to assess the performance of FCM clustering algorithms under various conditions and compare them against state-of-the-art clustering methods.

In conclusion, while FCM clustering offers a flexible and versatile approach to data clustering, it is essential to recognize its limitations and explore avenues for improvement. By addressing these challenges and capitalizing on emerging research trends, FCM clustering can continue to play a significant role in advancing the field of pattern recognition and data mining, particularly in leaf classification and related applications.

CHAPTER 4

4. PROPOSED SYSTEM

Our model is proposed primarily based on certain standards follows:

- Pre processing
- Feature extraction
- RCNN

Pre-processing:

Pre-processing serves as a fundamental step inside the pipeline of photo evaluation, specialLy in the domain of plant disorder detection. In the context of detecting pepper leaf illnesses using image information, pre-processing plays a crucial role in improving the first-rate and usability of the photographs before feeding them into the following tiers of analysis. This text elaborates at the importance of pre-processing inside the detection of pepper leaf diseases and outlines not unusual strategies hired on this phase.

Feature Extraction:

Function analysis emphasizes the importance of person traits or capabilities in item popularity. These features can be visible cues such as shade, texture, form, or spatial relationships. With the aid of specializing in discriminative capabilities, we can lessen the complexity of the information and improve the efficiency of machine studying algorithms.

RCNN:

To deal with the computational inefficiency related to processing a large range of vicinity proposals, we delivered a breakthrough approach in object detection called rapid R-CNN.

In contrast to its predecessor, R-CNN, which required each location notion to be personally fed right into a CNN for function extraction and type, fast R-CNN operates extra effectively through processing the whole photo straight away.

In fast R-CNN, the process begins with the input photograph being passed through a CNN to generate a convolutional function map. This feature map encodes rich data about the spatial layout and content material of the image. As opposed to producing region proposals one at a time, the function map is used to identify areas of hobby ROI directly. Once the ROIs are recognized, they are converted into rectangular regions and resized to a fixed size the use of an ROI pooling layer.

4.1 Advantages:

- High Accuracy
- Handling Multiple Classes
- Automation and Efficiency
- probability

4.2 Functional Requirements:

- Data Collection
- Data Pre-processing
- Training And Testing
- Modelling
- Predicting

Data Collection:

Data collection is the initial step in developing a leaf identification system. It involves gathering a diverse set of leaf images to ensure the model's robustness and accuracy. The data collection process may include the following steps:

Selection of Leaf Species:

- Determine the target leaf species that the system will identify.
- Choose a variety of species to ensure comprehensive coverage.

Acquisition of Leaf Images:

- Collect high-quality images of leaves representing various angles, lighting conditions, and backgrounds.
- Ensure that images capture different stages of leaf growth and variations within the same species.

Dataset Annotation:

- Annotate each leaf image with corresponding labels indicating the species or relevant attributes.
- Ensure consistency and accuracy in annotation to facilitate model training.

Data Augmentation:

- Augment the dataset by applying transformations such as rotation, scaling, and flipping to increase its diversity.
- Maintain a balance between augmentation techniques to avoid overfitting.

Quality Control:

- Perform quality checks to remove blurry, distorted, or irrelevant images from the dataset.
- Ensure uniformity in image resolution, format, and orientation.

Data Pre-processing:

Data pre-processing is crucial for enhancing the quality of input data and improving model performance. It involves several steps to prepare the dataset for training and testing.

Image Pre-processing:

- Resize images to a standard size to facilitate computational efficiency.
- Normalize pixel values to a common scale to mitigate variations in illumination and contrast.
- Apply noise reduction techniques to improve image clarity and reduce background interference.

Feature Extraction:

- Extract relevant features from leaf images, such as shape, texture, and colour characteristics.
- Use appropriate feature extraction algorithms to capture distinctive attributes of different leaf species.

Data Splitting:

- Divide the dataset into training, validation, and testing sets to assess model performance effectively.
- Ensure that each set contains a representative sample of leaf images from all species.

Label Encoding:

- Encode categorical labels into numerical representations for compatibility with machine learning algorithms.
- Use one-hot encoding or label encoding techniques based on the nature of the classification problem.

Training and Testing:

Training and testing are iterative processes aimed at building and evaluating the performance of the leaf identification model.

Model Selection:

- Choose an appropriate machine learning or deep learning architecture for leaf classification tasks.
- Consider factors such as model complexity, interpretability, and computational requirements.

Training Process:

- Train the selected model using the pre-processed dataset to learn the underlying patterns and relationships.
- Optimize model hyper parameters through techniques like grid search or random search to improve performance.

Evaluation Metrics:

- Define evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's performance.
- Use cross-validation techniques to obtain reliable estimates of the model's generalization ability.

Model Validation:

- Validate the trained model using the validation dataset to ensure it does not over fit or under fit the data.
- Fine-tune model parameters based on validation results to achieve optimal performance.

Modelling:

Modelling involves the development and refinement of algorithms for leaf identification based on the extracted features and training data.

Algorithm Implementation:

- Implement machine learning or deep learning algorithms for leaf classification using libraries such as Tensor Flow.
- Customize algorithms to address specific challenges or requirements of the leaf identification task.

Ensemble Techniques:

- Explore ensemble learning methods such as random forests or gradient boosting to enhance model robustness and accuracy.
- Combine multiple base classifiers to leverage their individual strengths and mitigate weaknesses.

Transfer Learning:

- Apply transfer learning techniques to leverage pre-trained models and adapt them to the leaf identification domain.
- Fine-tune pre-trained models on the target dataset to expedite training and improve performance.

Predicting:

Predicting involves using the trained model to classify new leaf images and make accurate predictions.

Inference:

- Deploy the trained model to classify unseen leaf images based on learned patterns and features.
- Implement efficient inference pipelines to handle real-time classification requests with minimal latency.

Confidence Estimation:

- Provide confidence scores or probability distributions for predicted classes to quantify the model's uncertainty.
- Enable users to interpret and trust model predictions based on confidence levels.

Error Analysis:

- Conduct error analysis to identify misclassified or ambiguous leaf samples and understand underlying causes.
- Refine the model based on error patterns to enhance its performance over time.

In summary, the leaf identification system's functional requirements encompass various stages, including data collection, pre-processing, training, modelling, and prediction. By meticulously addressing each step's intricacies and challenges, developers can design robust and accurate systems capable of identifying diverse leaf species with high precision and reliability.

4.3 Non Functional Requirements

Non-Functional Requirement specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of Non Functional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

CHAPTER 5

5. SYSTEM ARCHITECTURE

By using appearing the convolution operation best once according to image, fast R-CNN extensively reduces computational overhead compared to R-CNN. This efficiency benefit lets in for quicker inference times without sacrificing accuracy. Additionally, the usage of ROI pooling ensures that each area proposal contributes precious facts to the final classification decision.

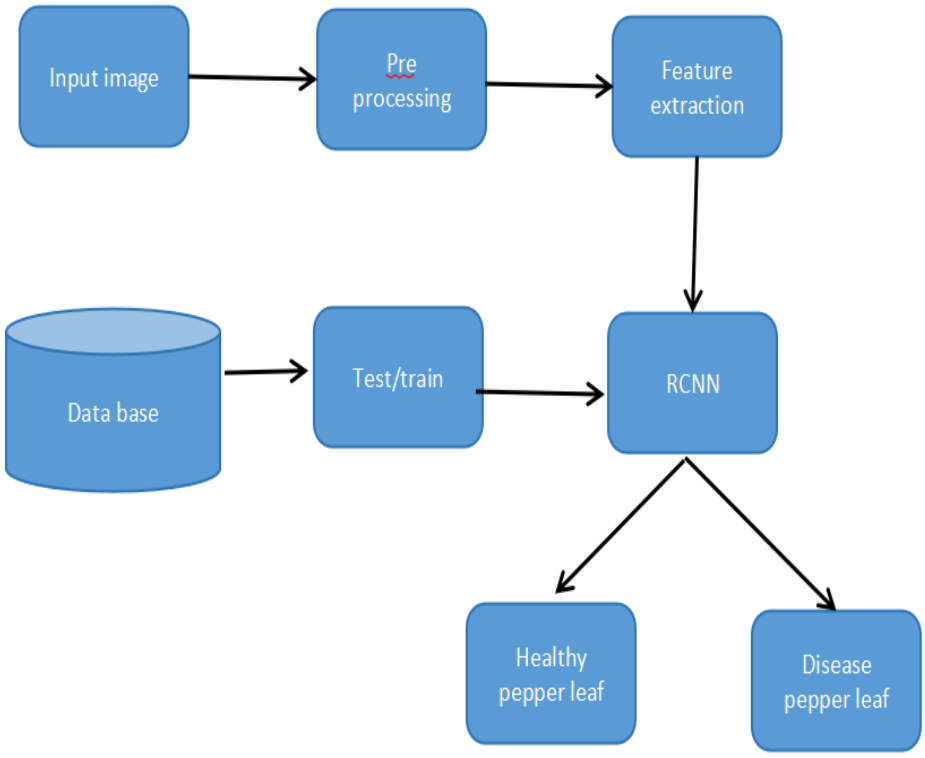

Fig.No.5.0:-System Architecture

5.1 Data Flow Diagram:

1. DFDs are commonly used during the system analysis and design phases of software development to understand and document the flow of data within a system.

2. They provide a visual representation of how data moves through a system, making it easier for stakeholders to understand the system's functionality and structure.

3. DFDs consist of four main components: processes, data stores, data flows, and external entities. Processes represent the functions or actions performed on the data, data stores represent where data is stored within the system, data flows represent the movement of data between processes and data stores, and external entities represent sources or destinations of data outside the system boundary.

4. DFDs are typically drawn using standardized symbols and notation, making them easily understandable by both technical and non-technical stakeholders.

5. There are different levels of DFDs, ranging from a high-level overview of the entire system to more detailed diagrams that focus on specific aspects or subsystems of the system.

6. DFDs help identify potential sources of data redundancy, inconsistency, or inefficiency within a system, allowing designers to make improvements or optimizations.

7. They facilitate communication and collaboration among project stakeholders, such as developers, designers, users, and managers, by providing a common language and visual representation of the system.

8. DFDs can be used iteratively throughout the software development lifecycle, from initial requirements gathering and analysis to system testing and maintenance, to ensure that the system meets the needs of its users and stakeholders.

9. DFDs can be converted into other types of diagrams or models, such as entity-relationship diagrams or object-oriented models, to provide different perspectives on the system and aid in its understanding and development.

10. DFDs are an integral part of structured systems analysis and design methodologies, such as SSADM and DFD approach, which emphasize a systematic and disciplined approach to software development.

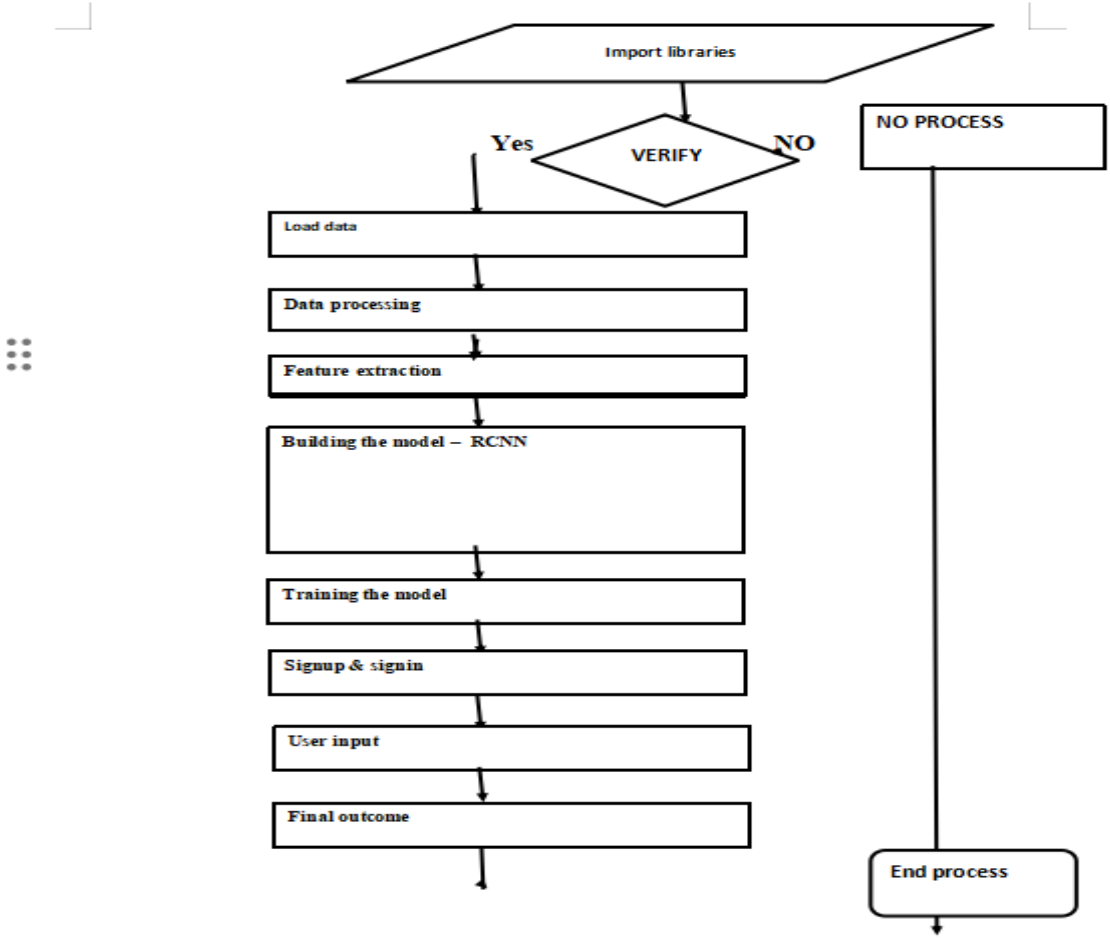

Fig.No.5.1:-Data Flow Diagram

5.2 Sequence Diagram:

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioural classifier that represents a declaration of an offered behaviour. Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behaviour of the subject without reference to its internal structure. These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behaviour, including exceptional behaviour and error handling.

SEQUENCE DIAGRAM:

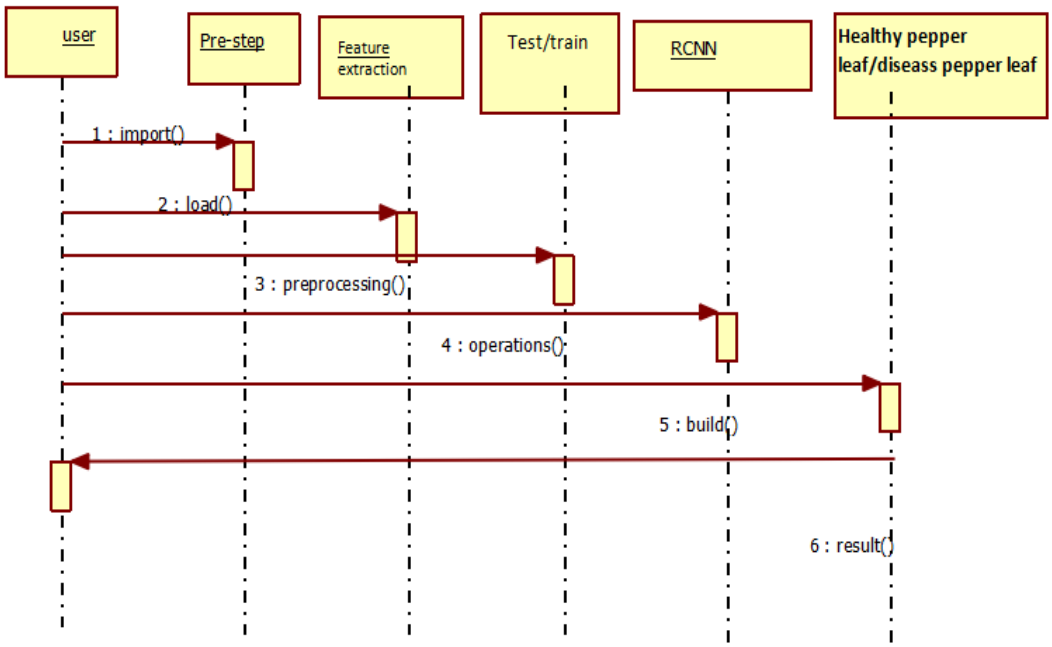

Fig.No.5.2:- Sequence Diagram

5.3 Activity Diagram:

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

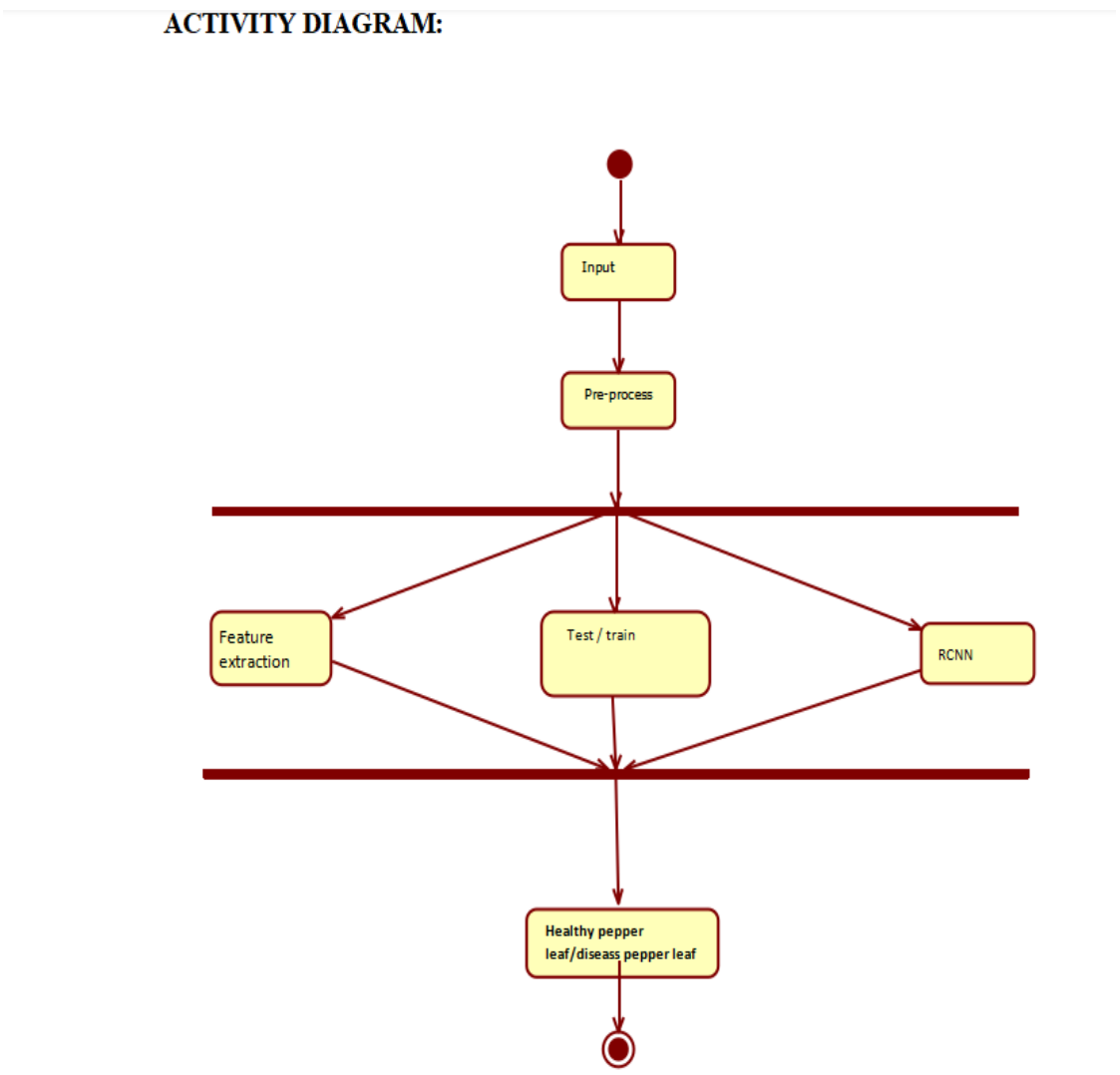

Fig.No.5.3:-Activity Diagram

5.4 Use-case Diagram:

- UML is a standard language for specifying, visualizing, constructing, and documenting the software systems.
- UML was created by OMG and UML 1.0 specification draft was proposed to the OMG in January 1997.
- OMG is continuously putting effort to make a truly industry standard.
- UML stands for Unified Modelling Language.
- UML is a pictorial language used to make software blue prints

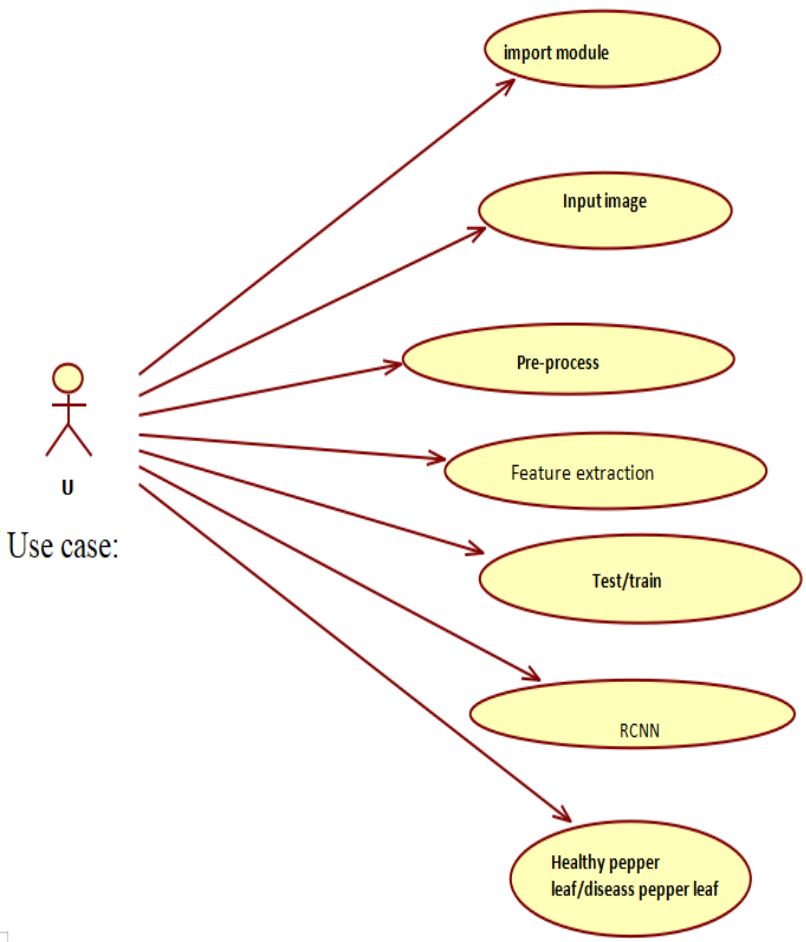

Fig.No.5.4:-Use Case Diagram

5.5 Collaboration Diagram:

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

COLLABORATION:

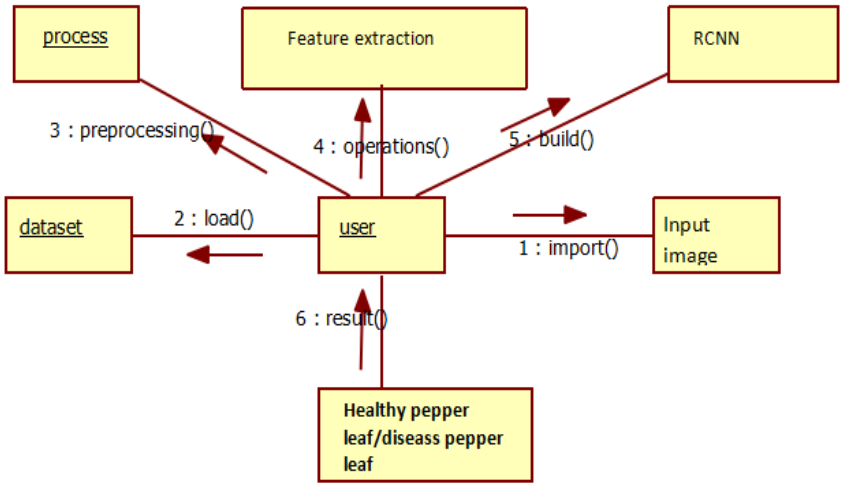

Fig.No.5.5:-Collaboration Diagram

5.6 Class Diagram:

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

- ✓ The top compartment contains the name of the class. It is printed in bold and centred, and the first letter is capitalized.
- ✓ The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- ✓ The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

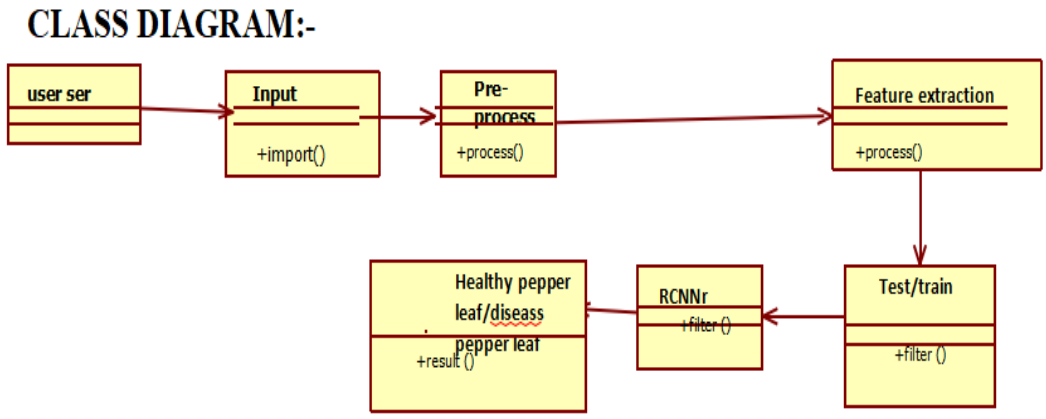

Fig.No.5.6:-Class Diagram

5.7 Component Diagram:

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as −

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

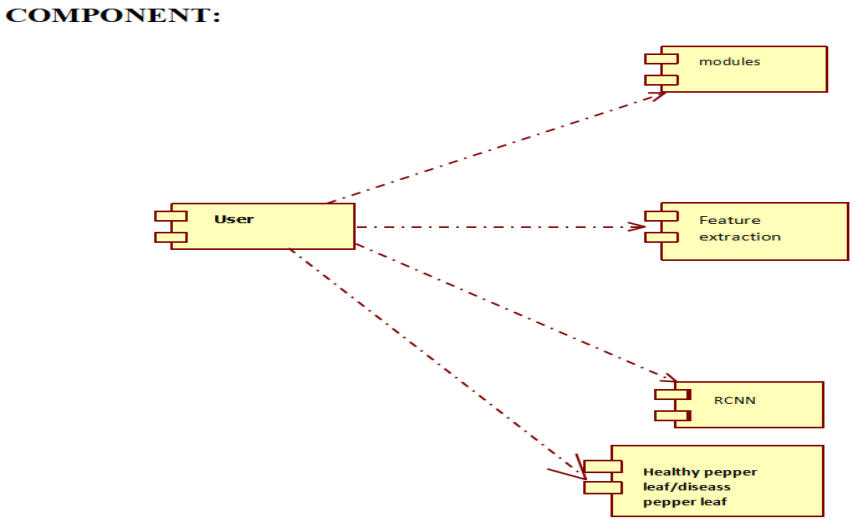

Fig.No.5.7:-Component Diagram

5.8 Deployment Diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

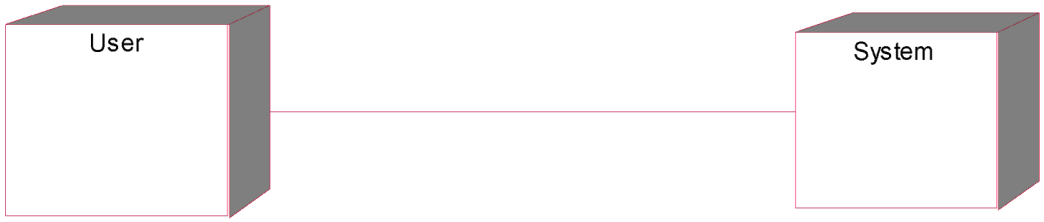

Fig.No.5.8:-Deployment Diagram

CHAPTER 6

6. EXPERIMENTAL ANALYSIS AND DISCUSSION

Image Acquisition is to acquire a leaf pepper image. A digital image is produced by one or several image sensors, which, besides various types of light-sensitive cameras, include range sensors, tomography devices, radar, ultra-sonic cameras, etc. Depending on the type of sensor, the resulting image data is an ordinary 2D image, a 3D volume, or an image sequence. The pixel values typically correspond to light intensity in one or several spectral bands (grey images or colour images), but can also be related to various physical measures, such as depth, absorption or reflectance of sonic or electromagnetic waves, or nuclear magnetic resonance.

Pre-processing: Pre-processing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

Feature extraction:-Feature analysis argues that we observe individual characteristics, or features, of every object and pattern we encounter. Recognition-by-components theory maintains that we sort objects into their component parts as a way of recognizing them. It yields better results than applying machine learning directly to the raw data.

Train and test:-Datasets are split into two subsets. The first subset is known as the training data - it's a portion of our actual dataset that is fed into the machine learning model to discover and learn patterns. In this way, it trains our model. The other subset is known as the testing data.

RCNN:

To bypass the problem of selecting a huge number of regions, Ross Girshick.et al. proposed a method where we use selective search to extract just 2000 regions from the image and he called them region proposals. Therefore, now, instead of trying to classify a huge number of regions, you can just work with 2000 regions. These 2000 region proposals are generated using the selective search algorithm which is written below.

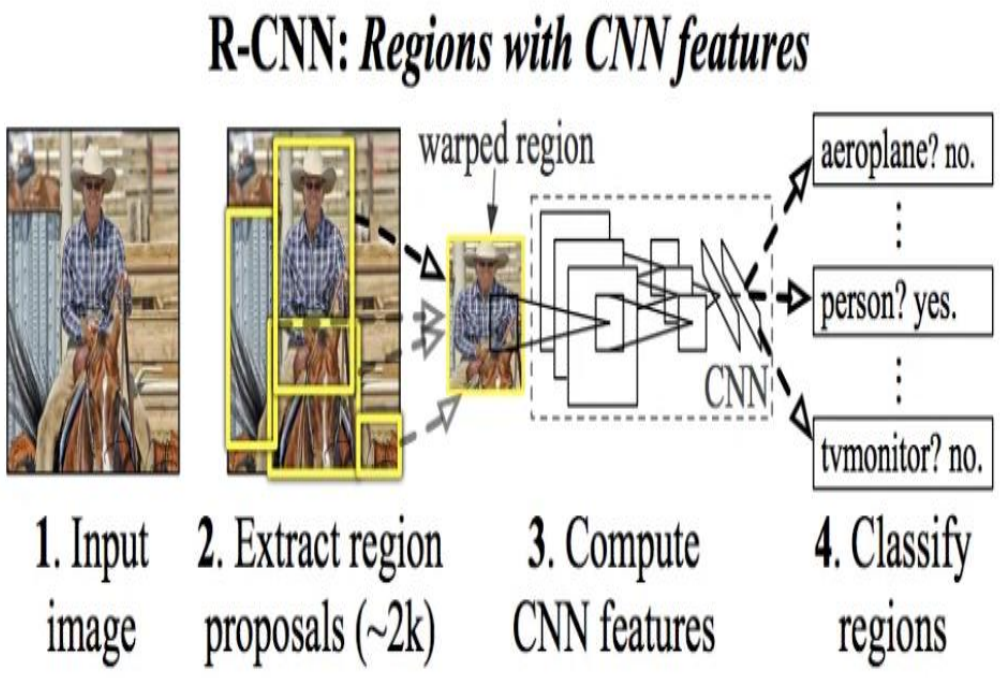

Fig.No.6.0:-RCNN

The same author of the previous paper R-CNN solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN. The approach is similar to the R-CNN algorithm.

But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a ROI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer. From the ROI feature vector, we use a soft max layer to predict the class of the proposed region and also the offset values for the bounding box.

The reason "Fast R-CNN" is faster than R-CNN is because you don't have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

Model Creation and Training:

- Facts series: acquire a dataset of high-resolution images containing pepper leaves affected, including wholesome leaves.
- Pre-processing: Pre-process the images to standardize their size, beautify comparison, and do away with noise. This ensures consistency and improves the model's potential to examine relevant capabilities.
- Version choice: pick out a suitable RCNN structure for the undertaking. Popular selections encompass faster RCNN, which integrates region concept networks for green place notion era, and masks RCNN, which extends faster RCNN to carry out instance segmentation further to object detection.
- Information Augmentation: increase the schooling dataset with strategies together with random rotations, translations, flips, and colour jittering to boom dataset variability and enhance model generalization. Observe augmentation judiciously to preserve the semantic meaning of pepper leaf pix and ensure that augmented samples continue to be representative of actual-world scenarios.

- Education: exceptional-song the pre-skilled RCNN model on the annotated dataset of pepper leaf pics. At some stage in schooling, the version learns to localize and classify disorder instances within the leaf pics. Utilize strategies which includes records augmentation (e.g., rotation, flipping, and scaling) to growth the variety of schooling examples and improve the version's robustness to variations in disorder look and leaf situations. Train the model the usage of a gradient-based totally optimization set of rules including SGD or Adam, adjusting the gaining knowledge of fee and different hyper parameters as needed to optimize performance.
- Assessment: To reveal generalization capability and come across ability overfitting, examine the performance of the trained RCNN version on a separate validation dataset. Use widespread metrics such as precision, remember, F1-score, and suggest common precision to quantitatively compare the model's accuracy in ailment detection and localization.
- Model Deployment: as soon as best overall performance is completed at the validation dataset, examine the skilled RCNN version on a separate test dataset to assess its actual-international overall performance. Deploy the skilled model for inference on new, unseen pepper leaf pixel, offering accurate and efficient disease detection capabilities for agricultural programs.

Loss	Accuracy	Val Loss	Val Accuracy
3.5680	0.0717	2.7247	0.0598
2.6660	0.0933	2.5799	0.2060
2.4816	0.2317	2.3525	0.3056
2.1220	0.3417	2.0172	0.3652
1.6762	0.4983	1.7834	0.3953
1.4743	0.5208	1.5787	0.5116
1.2470	0.6133	1.4353	0.5615

Table No.6.1:-Accuracy and Loss

6.1 Software Environment:

Python language:

- **Opencv:**

Introduction to Computer Vision:

Computer vision represents a significant advancement in computing, akin to the impact of mobile technology a decade ago. It involves using software to interpret and analyse visual content, including images, videos, and icons. While the concept dates back to the 1960s, recent strides in machine learning, data storage, computing power, and affordable high-quality input devices have greatly enhanced our software's ability to comprehend visual data.

What is Computer Vision?

Computer vision encompasses computations related to visual content, covering a wide range of tasks such as object classification, object identification, handwriting recognition, video motion analysis, image segmentation, scene reconstruction, and image restoration. Essentially, any software process that involves understanding pixels falls under the domain of computer vision.

How Computer Vision Works:

Computer vision processes images by interpreting them as grids of pixels, each represented by numerical values. Algorithms then analyse these pixels to derive insights. Despite advancements, there remains uncertainty about how our eyes and brains process images, which makes it challenging to replicate these processes accurately in algorithms.

Challenges and Computational Complexity:

Representing images computationally requires significant memory and processing power. For example, storing a typical image involves millions of bits, and training models for tasks like deep learning demands large datasets and extensive computational resources.

Business Applications of Computer Vision:

Major tech companies like Google and Facebook leverage computer vision for tasks such as mapping, facial recognition, and image analysis. Beyond tech giants, industries like automotive and healthcare are integrating computer vision into products and services, such as autonomous vehicles and medical diagnostics.

Computer Vision Resources:

Various frameworks and libraries facilitate computer vision tasks. OpenCV, SimpleCV, and Mahotas are popular choices, offering functionalities for image processing and analysis. Additionally, NumPy provides essential tools for numerical operations and array manipulation, making it a fundamental component for implementing computer vision algorithms in Python.

Conclusion:

Computer vision plays a crucial role in modern technology, enabling diverse applications across industries. With ongoing advancements and the availability of powerful tools and frameworks, integrating computer vision into applications is becoming increasingly accessible and impactful.

Imutils:

Imutils is a collection of convenient functions designed to simplify basic image processing tasks in Python using OpenCV. It offers a range of functionalities including translation, rotation, resizing, skeletonization, and displaying images with Matplotlib.

Translation:

Translation involves shifting an image along the X or Y direction. In OpenCV, this requires constructing a translation matrix (denoted as M) with the (X, Y) shift values and applying the `cv2.warpAffine` function. Imutils provides a `translate` function, eliminating the need for manual matrix construction and function calls.

Rotation:

Rotating an image in OpenCV typically involves calling `cv2.getRotationMatrix2D` and `cv2.warpAffine`. Imutils simplifies this process with its `rotate` function, which handles the rotation operation efficiently, including the calculation of rotation matrix and point coordinates.

Resizing:

Resizing an image in OpenCV can be complex, especially when maintaining aspect ratio. Imutils' `resize` function streamlines this task by preserving aspect ratio and offering keyword arguments for specifying width and height. Additionally, it eliminates the need for explicit dimension calculations.

Skeletonization:

Skeletonization is the process of extracting the topological skeleton of an object in an image.

While OpenCV lacks a dedicated skeletonization function, Imutils provides a convenient `skeletonize` function to construct the skeleton using morphological and binary operations. It offers control over the structuring element and defaults to a rectangular kernel.

Displaying with Matplotlib:

OpenCV represents images as NumPy arrays in BGR order, which may cause compatibility issues with Matplotlib's `plt.imshow` function. Imutils addresses this by offering a simple solution: either using `cv2.cvtColor` to convert the image to RGB order or utilizing the `opencv2matplotlib` convenience function for seamless display. Overall, Imutils simplifies common image processing tasks, making them more accessible and efficient for Python developers using OpenCV. Tensor Flow:

The most famous deep learning library in the world is Google's Tensor Flow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations. To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

- Researchers
- Data scientists
- Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency. Google does not just have any data; they have the world's most massive computer, so Tensor Flow was built to scale. Tensor Flow is a library developed by the Google Offense and non-offense Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

In this tutorial, you will learn

Tensor Flow Architecture:

Tensor flow architecture works in three parts:

- Pre-processing the data
- Build the model
- Train and estimate the model

It is called Tensor flow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output. This is why it is called Tensor Flow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Where can Tensor flow run?

Tensor Flow can hardware, and software requirements can be classified into Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop. Run Phase or Inference Phase: Once training is done Tensor Flow can be run on many different platforms.

You can run it on Desktop running Windows, mac OS or Linux Cloud as a web service Mobile devices like iOS and Android You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. Tensor Flow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, Tensor Flow can be accessed and controlled by other languages mainly, Python. Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what Tensor Flow is doing.

- List of Prominent Algorithms supported by Tensor Flow
- Linear regression:
- Classification :
- Deep learning classification:
- Booster tree regression:
- Boosted tree classification:

6.2 System Testing:

System testing, also referred to as system-level tests or system-integration testing, is the process in which a QA team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

Software Testing Strategies:

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.

Testing:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software Testing can also be stated as the process of validating and verifying that a software program/application/product:

- Meets the business and technical requirements that guided its design and Development.
- Works as expected and can be implemented with the same characteristics.

6.3 Testing Methods:

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

- Functions: Identified functions must be exercised.
- Output: Identified classes of software outputs must be exercised.
- Systems/Procedures: system should work properly

Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

Test Case for Excel Sheet Verification:

Here in machine learning we are dealing with dataset which is in excel sheet format so if any test case we need means we need to check excel file. Later on classification will work on the respective columns of dataset.

TEST CASES:

Test case1: (packages testing)

Input: downloading packages in interactive mode

Output: importing packages in script mode

Test case2: (data process)

Input: load data

Output: load data and display data in output code

Test case 3: (pre-process)

Input: do pre-process

Output: did pre-process using resize and conversion

Test case 4: (output)

Input: find output

Output: do the training part with algorithm and check leaf image healthy pepper leaf or disease pepper leaf

6.4 Result:

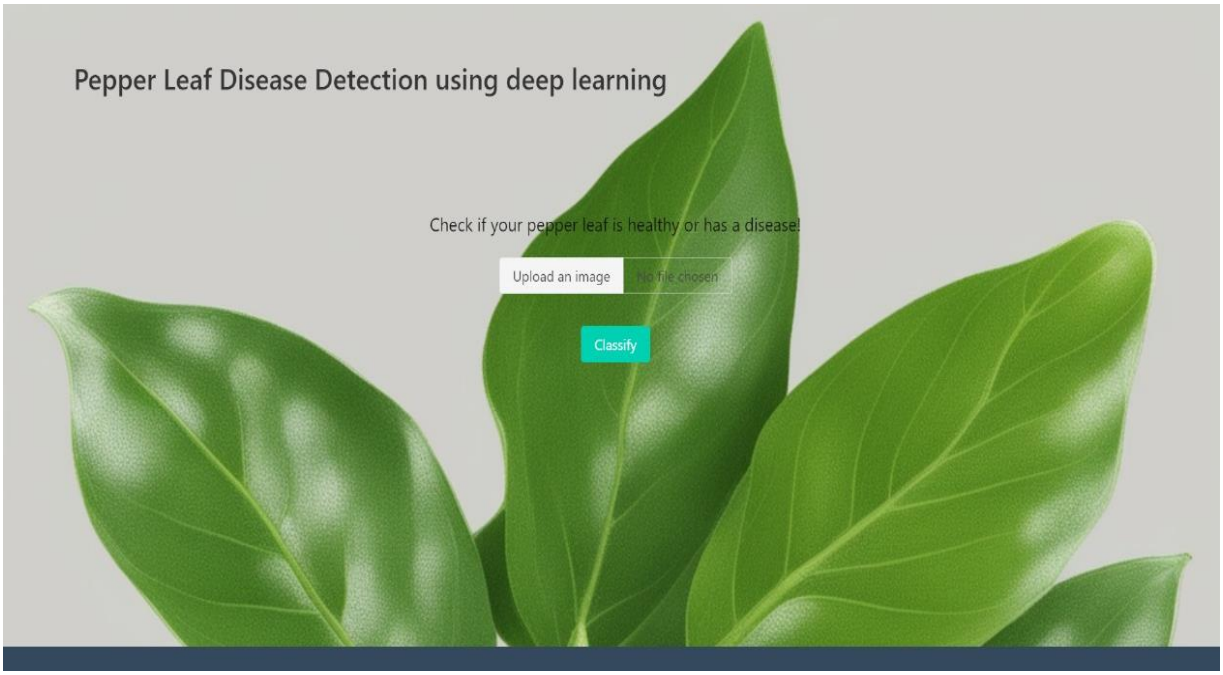

Fig.No.6.4.1:-Home Page

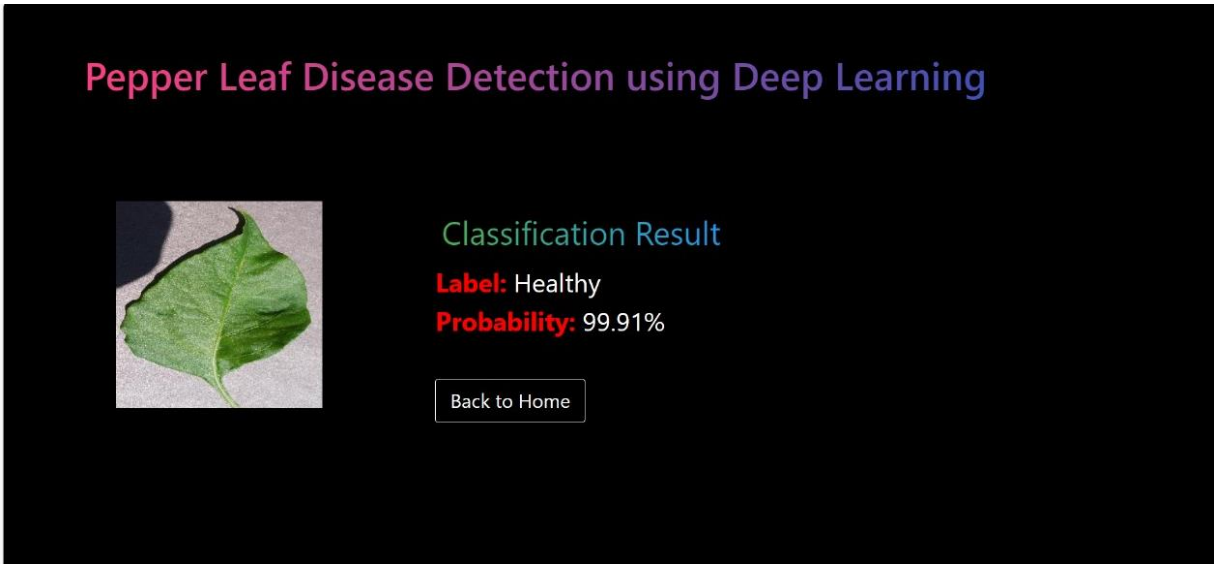

Fig.No.6.4.2:-Healthy Leaf

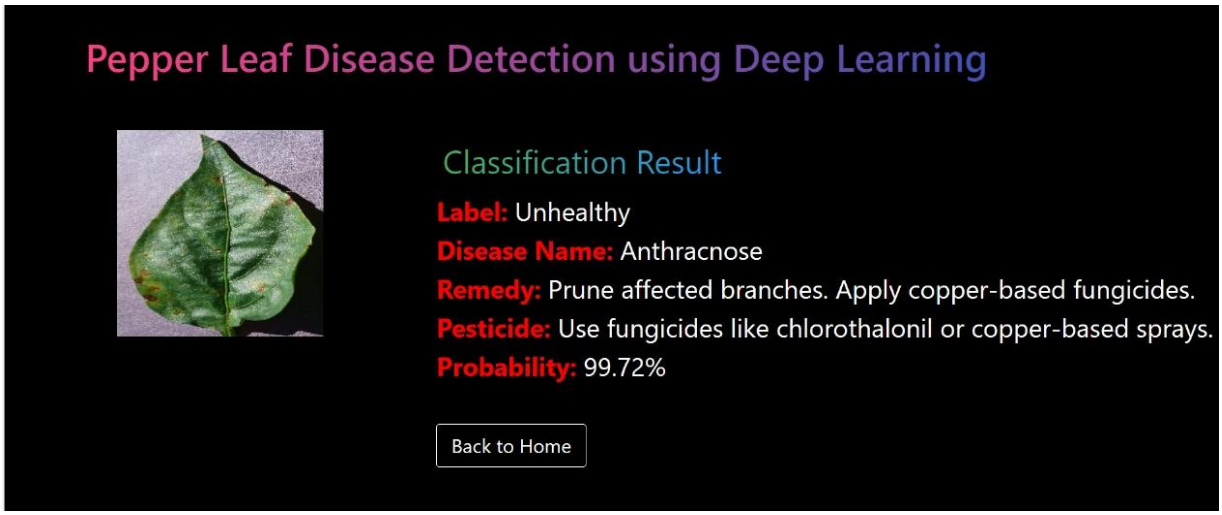

Fig.No.6.4.3:-Un Healthy Leaf

CHAPTER 7

7. CONCUSION & FUTURE SCOPE:

This paper presents a technique for detecting diseases on pepper leaves the use of location RCNNs, with a focus on achieving high accuracy. The method entails training the gadget on a various dataset comprising both healthful and diseased pepper leaves. By using leveraging system gaining knowledge of strategies, the machine learns to differentiate among distinct disorder classes and wholesome states. Upon receiving an input image of a pepper leaf, the device suits it towards the discovered styles and identifies the maximum likely state. Furthermore, this research paves the way for future improvements, inclusive of real-time sickness detection in pepper flora thru video analysis. Moreover, there may be ability to increase the device's competencies to expect the susceptibility of pepper flowers to sicknesses based on their modern-day situation.

In end, this paper gives a strong technique for detecting illnesses on pepper leaves the usage of region-primarily based convolutional neural networks. The gadget demonstrates excessive accuracy and holds promise for actual-world programs in agricultural settings. By way of leveraging gadget mastering strategies, this studies contributes to improvements in ailment control. The overall accuracy rating of 0.Ninety one displays the version's strong performance in detecting multiple diseases simultaneously, thinking about the magnificence imbalance and complexity of the detection mission. By using utilising device getting to know techniques, this studies contributes to advancements in disorder management. The overall accuracy rating of zero. Ninety one reflects the model's robust overall performance in detecting multiple illnesses simultaneously, thinking about the elegance imbalance and complexity of the detection project. To hit upon whether pepper leaves are healthy or unhealthy the usage of the model that we skilled and tested at the Plant Village dataset.

1. REFERENCES:

[1] Midhun P Mathew, Therese Yamuna Mahesh, Determining The Region of Apple Leaf Affected by Disease Using YOLO V3. In 2021 International Conference on Communication, Control and Information Sciences (ICCISc-2021)

[2] Abirami Devaraj, Karunya Rathan, Sarvepalli Jaahnavi and K Indira. Identification of Plant Disease using Image Processing Technique. International Conference on Communication and Signal Processing, April 4-6, 2019, India

[3] Surampalli Ashok , Gemini Kishore , Velpula Rajesh , S. Suchitra, S.G.Gino Sophia, B.Pavithra. Tomato Leaf Disease Detection Using Deep Learning Techniques. Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020)IEEE Conference Record # 48766; IEEE Xplore ISBN: 978-1-7281-5371-1

[4] Anand R,Veni S,Aravinth J. An Application of image processing techniques for Detection of Diseases on Brinjal Leaves Using K-Means Clustering Method. 2016 Fifth International Conference On Recent Trends In Information Technology

[5] Ch. Usha Kumari, S. Jeevan Prasad, G. Mounika. Leaf Disease Detection: Feature Extraction with K-means clustering and Classification with ANN. 2019 3rd International Conference on Computing Methodologies and Communication.