Project Report on

Computer Aided Segmentation of Liver Lesions in CT SCANS Using Cascade Convolution Neural Network and Adam Optimised Classifier

Submitted to OSMANIA UNIVERSITY for the partial fulfilment of the requirement for the award of the degree for

MASTER OF COMPUTER APPLICATIONS

SUBMITTED BY

BONGURAM ARUN KUMAR REDDY

HT.NO: 151221862047

DEPARTMENT OF INFORMATICS

OSMANIA UNIVERSITY

HYDERABAD-500007

September 2023

Project Report on

Computer Aided Segmentation of Liver Lesions in CT SCANS Using Cascade Convolution Neural Network and Adam Optimised Classifier

Submitted to OSMANIA UNIVERSITY for the partial fulfilment of the requirement for the award of the degree for

MASTER OF COMPUTER APPLICATIONS

SUBMITTED BY

BONGURAM ARUN KUMAR REDDY

HT.NO: 151221862047

RAJA BAHADHUR VENKATA RAMA REDDY INSTITUTE OF TECHNOLOGY

Hanuman Tekdi, Abids, Hyderabad-500001

(Affiliated to Osmania University, Approved by AICTE & Govt. of TS)

September 2023

RAJA BAHADUR VENKATA RAMA REDDY
INSTITUTE OF TECHNOLOGY
(Sponsored by RBVRR Educational Society, Registered No.17/56F)
(Affiliated to Osmania University & Approved by AICTE)
Hanuman Tekdi, Abids, Hyderabad – 500 001.
Phone No. 040-2475 2118, Website: www.rbvrrit.com, Email : info@rbvrrit.com

CERTIFICATE

This is to certify that **Mr. Bonguram Arun Kumar Reddy** **Bearing Roll No. 151221862047** have developed software Project Titled "**Computer Aided Segmentation of Liver Lesions in CT SCANS Using Cascade Convolution Neural Network and Adam Optimised Classifier"** as Partial fulfilment for the award of the Degree of **MASTER OF COMPUTER APPLICATIONS** from **OSMANIA UNIVERSITY.**

Place: Hyderabad **PRINCIPAL**

Date:

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I hereby declare that the results embodied in this project Entitled "**Computer Aided Segmentation of Liver Lesions in CT SCANS Using Cascade Convolution Neural Network and Adam Optimised Classifier"** is carried out by me during the period from June to September 2023 in partial fulfilment of the degree of Master of Computer Applications from Osmania University, and I have not submitted the same to any other University/Institute for the award of any other degree.

Place: Hyderabad **By**

Date: **B. Arun Kumar Reddy**

1512-21-862-047

ACKNOWLEDGEMENT

I express my gratitude to lord for showering his grace and blessings upon me for the completion of this project.

Although our name appears on the cover of this book, many people has contributed in some form of the other to this project development. I would not have completed this project without the assistance or support of each of the following. I thank you all.

Whole heartedly I would like to thanks to my project guide, **Mrs P.Ashalatha**, Assistant Professor of **RBVRRIT** College, for her constant motivation and valuable help through the project work. I also extend my thanks to **Mr. P. Madhu Kumar Reddy**, Vice Principal of **RBVRRIT** College, for his help through the course. I express my gratitude to **Professor G.Tirupathi**, Principal of **RBVRRIT** for his valuable suggestions and advices throughout the MCA course. Also, I would like to extend my thanks to other Faculties and staff for their Co-operation during my Course.

Finally, I would like to thank my friends for their cooperation to complete this project.

ABSTRACT

Automatic liver segmentation not only plays an important role in the analysis of liver disease, but also reduces the cost and humanity's impact in segmentation. In addition, liver segmentation is a very challenging task due to countless anatomical variations and technical difficulties. Many methods have been designed to overcome these challenges, but these methods still need to be improved to obtain the liver disease using CNN. In this paper, a fast algorithm is proposed for liver extraction from CT images with single-block linear detection. The proposed method does not require iteration; thus, the computational time and complexity are decreased enormously. In addition, the initialization is not crucial in the algorithm, so the algorithm's robustness and specificity are improved. The experimental evaluation of the proposed method revealed effective segmentation in normal and abnormal (liver haemangioma and liver cancer) abdominal CT images.

The project aims to develop a robust computer-aided system for the automated segmentation of liver lesions in CT scans. To achieve this, a Cascade Convolutional Neural Network (CNN) architecture is proposed, incorporating an Adam-optimized classifier. The cascade architecture consists of multiple stages of CNNs, with each stage refining the lesion segmentation progressively. The Adam optimizer is employed to fine-tune the classifier's parameters for improved accuracy and convergence speed. The proposed system leverages deep learning techniques to enhance the precision and efficiency of liver lesion detection, potentially aiding radiologists in diagnosing and monitoring liver diseases more effectively. The project's implementation and evaluation demonstrate promising results in terms of segmentation accuracy and efficiency.

INDEX

INTRODUCTION

Liver disease is a significant global health concern, affecting millions of people worldwide. Early and accurate detection of liver diseases plays a crucial role in effective treatment and improved patient outcomes. In recent years, advancements in deep learning, particularly Convolutional Neural Networks (CNNs), have shown tremendous potential in medical image analysis, including the detection of liver diseases from medical images.

This project aims to develop a CNN-based system for liver disease detection from medical images, particularly focusing on analysing liver images obtained through medical imaging modalities like computed tomography (CT) scans or magnetic resonance imaging (MRI) scans.

The use of CNNs for liver disease detection offers several advantages over traditional methods. CNNs can automatically learn discriminative features from the input images, allowing them to identify complex patterns and anomalies associated with liver diseases. Moreover, CNNs can handle large amounts of medical image data efficiently and generalize well to unseen cases, enhancing the system's diagnostic capabilities.

The proposed system will follow a supervised learning approach, where it will be trained on a labeled dataset of liver images, where each image is annotated with its corresponding disease status (e.g., normal, fatty liver, cirrhosis, or tumors). The CNN will be designed to classify liver images into different disease categories or perform binary classification, depending on the specific problem setting.

Key components of the project include data preprocessing, CNN architecture design, model training, and evaluation. The dataset will be carefully curated and pre-processed to ensure consistency and reliability during training. The CNN architecture will be selected or designed to suit the liver disease detection task, considering factors such as image resolution, depth of the network, and use of transfer learning.

During model training, an appropriate loss function and optimization algorithm will be employed to fine-tune the CNN's parameters. Evaluation metrics such as accuracy, precision, recall, and F1-score will be used to assess the model's performance on a separate test dataset.

The successful implementation of the CNN-based liver disease detection system holds the potential to revolutionize the way liver diseases are diagnosed and monitored.

It can assist medical professionals in making more accurate and timely diagnoses, leading to improved patient care and better management of liver diseases. Furthermore, the project contributes to the growing field of AI-powered healthcare, showcasing the benefits of deep learning in medical image analysis.

DOMAIN OVERVIEW:

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.

PROBLEM STATEMENT:

Infection. Parasites and infections can taint the liver, causing aggravation that diminishes liver capability. The infections that cause liver harm can be spread through blood or semen, defiled food or water, or close contact with a contaminated. individual.

OBJECTIVE:

The objective of this project is to develop a CNN-based system for liver disease detection from medical images, particularly focusing on analyzing liver images obtained through medical imaging modalities such as computed tomography (CT) scans or magnetic resonance imaging (MRI) scans. The system aims to assist medical professionals in accurately diagnosing liver diseases at an early stage, leading to better patient outcomes and effective treatment plans.

1.1. COMPANY PROFILE

PANTECH E LEARNING

Pantech E Learning Pvt Ltd: Empowering Technical Excellence in Education Pantech eLearning is a technical training and design solutions company that focuses on providing technical training in various domains such as Machine Learning, Deep Learning, IoT, Embedded Systems, Robotics, Android Development, Open CV, Python, VLSI, Image Processing, Network Security, Power Electronics, Power Systems, Renewable Energy, Big data and Data Science. The company also provides design and R&D solutions, project designs, and DIY kits. Pantech eLearning was launched with the intention of making the process of digital e-learning simplified and to make the students and staff become technology-savvy, creative, and active contributors to the development of technology solutions.

Pantech eLearning has been successful in inculcating technical education cum services for more than a decade. The company has a commitment to quality, constant innovation, and exceeding expectations, always in pursuit of technological updates and transfer of knowledge to the staff and student community. Pantech R&D covers everything from basic engineering to advanced research in electronics, electrical, and software domains. The company provides 2 customized product design and manufacture for customers, project designs and installation of project prototypes in colleges, design, testing, and manufacturing of lab equipment to the engineering colleges, manufacturing and sales of project kits, on campus technical training at college premises, in-house trainings, short-term and long term course offerings, courses with placements, two/three days workshop programs and hands-on sessions, faculty development programs (FDP) and STTPs, internships/in-plant trainings, workshop and the internship for both the faculties and the students, guidance for projects for final years/pre-final students of engineering and diploma stream, research project solutions to colleges as a whole and to corporates, conducting faculty development programs and hands-on sessions in the project domains, demo of the lab kits to the staff of engineering colleges, and working on the global market through e commerce.

LITERATURE SURVEY

1.TITLE: Fuzzy Logic for Child-Pugh classification of patients with Cirrhosis of Liver

AUTHORS: Anu Sebastian, Surekha Mariam Varghese

Survival analysis is a common procedure in the medical field. Predicting life expectancy is a highly significant factor in the decision making of both patients and doctors. In addition to the correct diagnosis of the ailment, accurate classification and assessment is necessary to commence appropriate and effective treatment. The Child-Pugh classification has been used for many years for the assessment of severity and prognosis of chronic liver disease and cirrhosis of liver. It is considered as a standard for classifying the patients with end–stage liver disease. Fuzzy logic is a clever technique that can be used to model imprecise and complex systems efficiently. Intelligent procedure based on Fuzzy logic could be used for the Child-Pugh classification of the patients suffering from liver cirrhosis. For different classes of cirrhosis patients, the life expectancy also differs. This paper presents a fuzzy intelligent technique to classify and predict the life expectancy of patients with cirrhosis of liver.

2.TITLE: Liver Disease Detection Due to Excessive Alcoholism Using Data Mining Techniques

AUTHORS: Insha Arshad, Chiranjit Dutta.

Alcohol is consumed in excess by millions of people across the world. Alcohol consumption is directly linked to life threatening liver diseases such as cirrhosis which may ultimately lead to death. Early detection of liver disease caused by over consumption of alcohol would help in saving lives of many people. By detecting liver disease in its early stage, it can be diagnosed in time and may lead to full recovery in some patients. This paper proposes detection as well as to predict the presence of liver disease using data mining algorithms. We will make a decision tree for the dataset and then the rules will be generated. After determining the rules, we will use different data mining algorithms to train and test the dataset to detect the liver disease. The data was collected from UCI repository and our training dataset was developed. It consists of 7 different attributes having 345 instances.

In the dataset, different categories of blood tests are taken into considerations which are directly linked to liver diseases that may arise due to excessive alcohol consumption along with frequency of alcohol consumption. Based on the type of liver disease detected, prognosis may be suggested.

3.TITLE: Prediction of liver cancer using Conditional probability Bayes theorem

AUTHORS: N. Ramkumar; S. Prakash; S. Ashok Kumar; K Sangeetha

Cancer is the one of the hazardous diseases in the world. Cancer spreads in lungs, liver, breast, bones etc. Liver cancer is the most dangerous and it will continue lifelong. The symptoms of liver cancer are Jaundice, loss of weight, yellow coloured urine, vomiting, pain in the upper right abdomen, sweats, fever and enlarged liver. The liver cancer which begins in the liver apart from moving from other part of the body is called as a primary liver cancer. Cancer which spreads all other part of the body and finally it reaches liver is called as secondary liver cancer. Liver is one of the important parts of the human. WHO surveys say out of 100,000 people, around 30 people are suffered from liver cancer and mostly it affects the African and Asian countries earlier. Nowadays it became a popular disease The most common kind of a liver cancer is called as hepatocellular carcinoma, this particular affects male rather than female. The liver cancer occurs mainly due to the more alcohol consumption. Many data mining algorithms, Artificial intelligence concepts are used to predict the liver cancer. The probability of predicting the liver cancer is performed using the Bayes theory with the WEKA tool.

4.TITLE: Rule Optimization of Boosted C5.0 Classification Using Genetic Algorithm for Liver disease Prediction

AUTHORS: Mafazalyaqeen Hassoon; Mikhak Samadi Kouhi ; Mariam Zomorodi Moghadam ; Moloud Abdar

One of the interesting and important subjects among researchers in the field of medical and computer science is diagnosing illness by considering the features that have the most impact on recognitions. The subject discusses a new concept which is called Medical Data Mining (MDM). Indeed, data mining methods use different ways such as classification and clustering to classify diseases and their symptoms which are helpful for diagnosing. This paper introduces a new method of liver disease diagnosis to help doctors and their patients in finding the disease symptoms and reduce a long time of diagnosing and prevent deaths. The proposed method will optimize the rules released from Boosted C5.0 classification method with the Genetic Algorithm (GA), to increase the diagnosis time and accuracy. So instead of using an evolutionary algorithm for producing rules, the genetic algorithm is used for improving and reducing rules of another algorithm. We show that our proposed approach has better performance and throughput in comparison with other work in the field. The accuracy is improved from 81% in [1] to 93% in our work

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Typically, the existing mechanisms assumed that the accuracy of prediction was achieved. But this wasn't the case then, hence, it must be improved further to increase the classification accuracy. Also, other research works addressed these issues by introducing efficient combination. Existing Models based on feature selection and classification raised some issues regarding with training dataset and Test dataset.

LIMITATION

- Certain approaches being applicable only for small data.
- Certain combination of classifier over fit with data set while others are under fit.
- Some approaches are not adoptable for real time collection of database implementation.

3.2 PROPOSED SYSTEM

In proposed system, we have to import the liver patient dataset CT images. Then the dataset should be pre-processed and remove the anomalies and full up empty cells in the dataset, so the we can further improve the effective Liver diseases detection. Normal and abnormal images

ADVANTAGES

- The performance classification of liver-based diseases is further improved.
- Time complexity and accuracy can be measured by various machine learning models, so that we can measures different.
- Different machine learning having high accuracy of result.
- Risky factors can be predicted early by machine learning models.

3.3 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

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

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.4 FUNCTIONAL REQUIREMENTS

1.Data Collection

2.Data Preprocessing

3.Training and Testing

4.Modiling

5.Predicting

3.5 NON-FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) 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 nonfunctional 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 is > 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

SYSTEM REQUIREMENT SPECIFICATION

SYSTEM REQUIREMENTS

4.1 Hardware System Configuration: -

- ❖ Processor - Intel i3 and above
- ❖ RAM - 4 GB (min)
- ❖ Hard Disk - 20 GB
- ❖ Key Board - Standard Windows Keyboard
- ❖ Mouse - Two or Three Button Mouse
- ❖ Monitor - SVGA

4.2 Software Requirements:

- ❖ Operating System - Windows 7 or above
- ❖ Coding Language - Python 3.6 or 3.9
- ❖ IDE - Python IDLE
- ❖ Tools - Anaconda Navigator
- ❖ Library - NumPy,
  sklearn,
  seaborn,
  matplotlib,
  Open cv.

SYSTEM DESIGN

Here are some essential modules for this project:

Data Preprocessing Module:

Responsible for loading, cleaning, and normalizing CT scan data.

May involve data augmentation techniques to increase the diversity of the training dataset.

Ensure compatibility with the input requirements of the neural network.

Cascade Convolutional Neural Network (CNN) Architecture:

Design the cascade architecture with multiple stages.

Each stage of the CNN should be a separate module responsible for feature extraction and lesion segmentation.

Define the connections and flow between these stages.

Adam-Optimized Classifier Module:

Implement the classifier module using the Adam optimizer.

Train the classifier to make decisions about lesion segmentation.

Ensure that the classifier's weights are optimized for accurate segmentation.

User Interface Module:

Develop a user-friendly interface for interacting with the system.

Allow radiologists to input CT scan data and view segmentation results.

Include features for reviewing and fine-tuning the segmentation if necessary.

Post-Processing Module:

After lesion segmentation, this module can perform additional processing steps.

It may involve noise reduction, smoothing, or refining the segmented regions for better accuracy.

Real-Time Processing Module (Optional):

If real-time processing is a requirement, design a module that optimizes the model for fast inference.

Consider techniques like model quantization or hardware acceleration.

Data Storage and Retrieval Module:

Implement a module for storing and retrieving segmented lesion data alongside patient records.

Ensure compatibility with Picture Archiving and Communication Systems (PACS) if necessary.

Model Training and Update Module:

Design a module that facilitates model training using new data.

Implement mechanisms for continuous learning and model updates to adapt to evolving datasets.

Explainability and Uncertainty Estimation Module:

Incorporate a module that provides explanations for the model's decisions and estimates the uncertainty in predictions.

This module can enhance the system's transparency and reliability.

Integration with Clinical Decision Support Systems (CDSS) Module:

Design a module to seamlessly integrate the segmentation results with CDSS, enabling comprehensive diagnostic assistance.

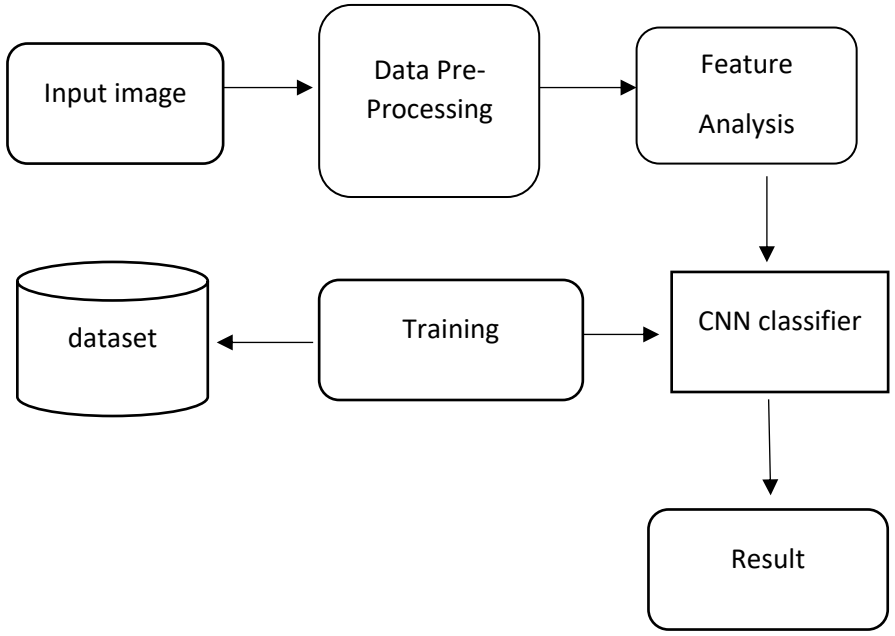
Regulatory Compliance and Security Module:

Ensure that the system complies with medical device standards and data security regulations. Implement encryption, access controls, and auditing mechanisms as needed.

Collaborative Research and Validation Module:

Create a module for collaborative research efforts to validate the system's performance across various clinical scenarios and patient populations.

ARCHITECHTURE DIAGRAM



System Design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements**.** System Design for tech interviews is something that can't be ignored! Almost every IT giant whether it be Facebook, Amazon, Google, or any other ask various questions based on System Design concepts such as scalability, load-balancing, caching, etc. in the interview.

This specifically designed System Design tutorial will help you to learn and master System Design concepts in the most efficient way from basics to advanced level.

It's a wide field of study in Engineering and includes various concepts and principles that will help you in designing scalable systems. These concepts are extensively asked in the Interview Rounds for SDE 2 and SDE 3 Positions at various tech companies. These senior roles demand a better understanding of how you solve a particular design problem,how you respond when there is more than expected traffic on your system, how you design the database of your system and many more. All these decisions are required to be taken carefully keeping in mind Scalability, Reliability, Availability, and Maintainability

Approaching a Design Problem

Breaking Down the Problem: When you are given a Design Problem start breaking it down to small components. These components can be Services or Features which you need to implement in the System. Initially, a System given to be designed can have a large number of features and you are not expected to design

Communicating your ideas: Communicate well with the Interviewer. While designing the system keep him in the loop. Discuss your process out loud. Try to demonstrate your design clearly on the whiteboard with flowcharts and diagrams. Describe your ideas to your interviewer, how you have planned to tackle the problem of scalability, how you will be designing your database and many others.

Assumptions that make sense: Make some reasonable assumptions while you are designing the System. Suppose you have to assume the number of requests the system will be processing per day, the number of database calls made in a month, or theefficiency rate of your Caching System. These are some of the numbers you need to assume while designing. Try to keep these numbers as reasonable as possible.

Reliability in System Design –

A system is Reliable when it can meet the end-user requirement. When you are designing a system, you should have planned to implement a set of features and services in your system. If your system can serve all those features without wearing out then your System can be considered to be Reliable.

A Fault Tolerant system can be one that can continue to be functioning reliably even in the presence of faults. **Faults** are the errors that arise in a particular component of the

system. An occurrence of fault doesn't guarantee Failure of the System. **Failure** is the state when the system is not able to perform as expected. It is no longer able to provide certain services to the end-users.

Availability in System Design –

Availability is a characteristic of a System which aims to ensure an agreed level of Operational Performance, also known as uptime. It is essential for a system to ensure high availability in order to serve the user's requests.

The extent of Availability varies from system to system. Suppose you are designing a Social Media Application then high availability is not much of aneed. A delay of a few seconds can be tolerated. Getting to view the post of your favorite celebrity on Instagram with a delay of 5 to 10 seconds will not be much of an issue. But if you are designing a system for hospitals, Data Centers, or Banking, then you should ensure that your system is highly available. Because a delay in the service can lead to a huge loss.

There are various principles you should follow in order to ensure the availability of your system.

- Your System should not have a Single Point of Failure. Basically, your system should not be dependent on a single service in order to process all of its requests. Because when that service fails then your entire system can be jeopardized and endup becoming unavailable.
- Detecting the Failure and resolving it at that point.

Scalability in System Design –

Scalability refers to the ability of the System to cope up with the increasing load. While designing the system you should keep in mind the load experienced by it. It's said that if you have to design a system for load **X** then you should plan to design it for **10X** and test it for **100X**. There can be a situation where your system can experience an increasing load. Suppose you are designing an E-commerce application then you can expect a spike in the load during a Flash Sale or when a new Product is Launched for sale. In that case, your system should be smart enough to handle the increasing load efficiently and that makes it **Scalable**.

In order to ensure scalability, you should be able to compute the load that your system will experience. There are various factors that describe the Load on the System:

- Number of requests coming to your system for getting processed per day
- Number of Database calls made from your system
- Amount of Cache Hit or Miss requests to your system
- Users currently active on your system

A good system design is to organize the program modules in such a way that are easy to develop and change. Structured design techniques help developers to deal with the size and complexity of programs. Analysts create instructions for the developers about how code should be written and how pieces of code should fit together to form a program.

1. If any pre-existing code needs to be understood, organized, and pieced together.
2. It is common for the project team to have to write some code and produce original programs that support the application logic of the system.

5.1 UML DIAGRAMS

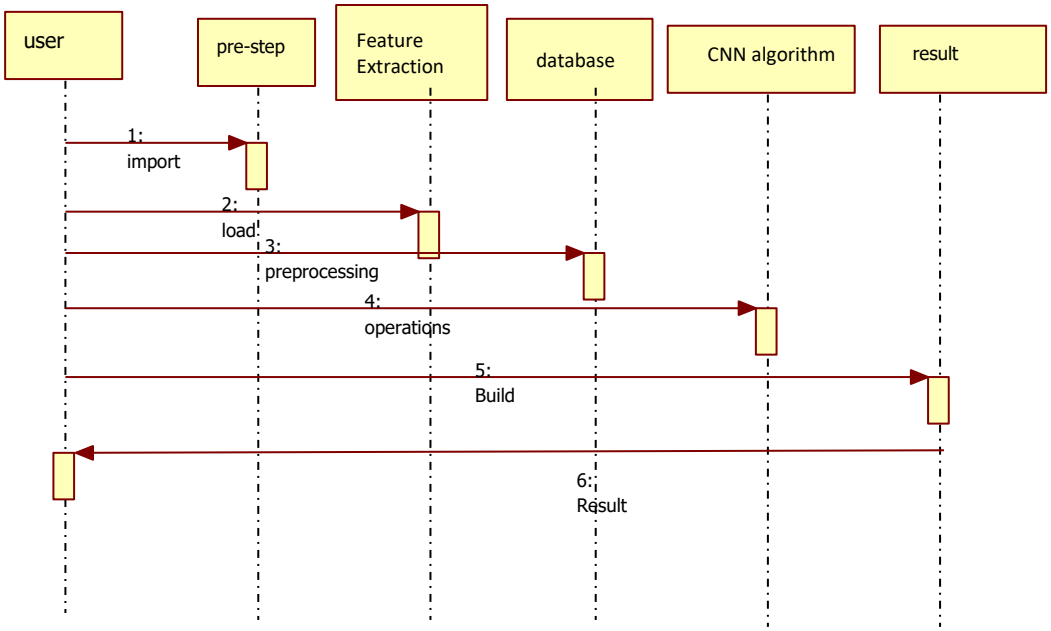
The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

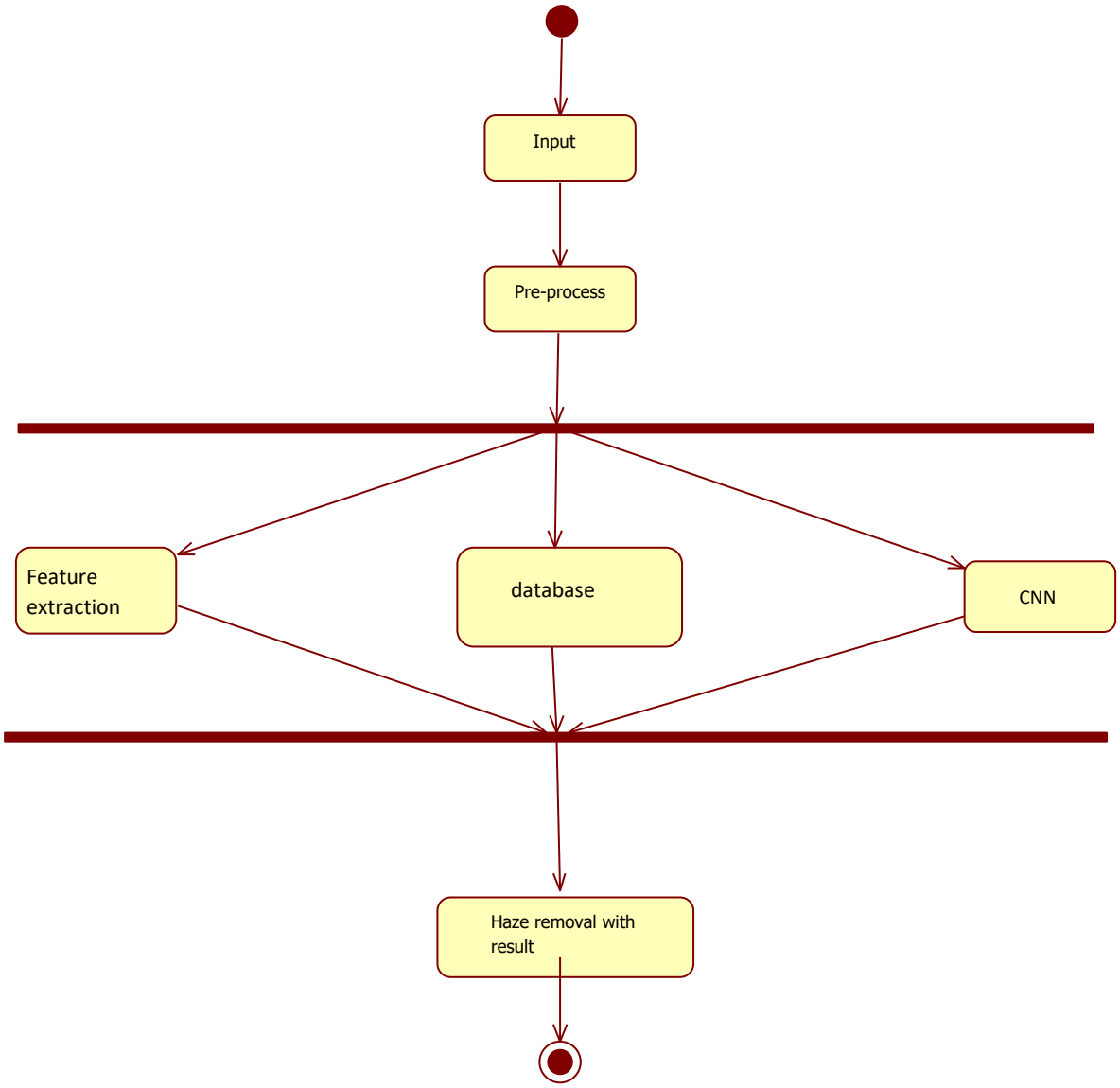
Sequence Diagram:

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, 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 behavior, including exceptional behavior and error handling.



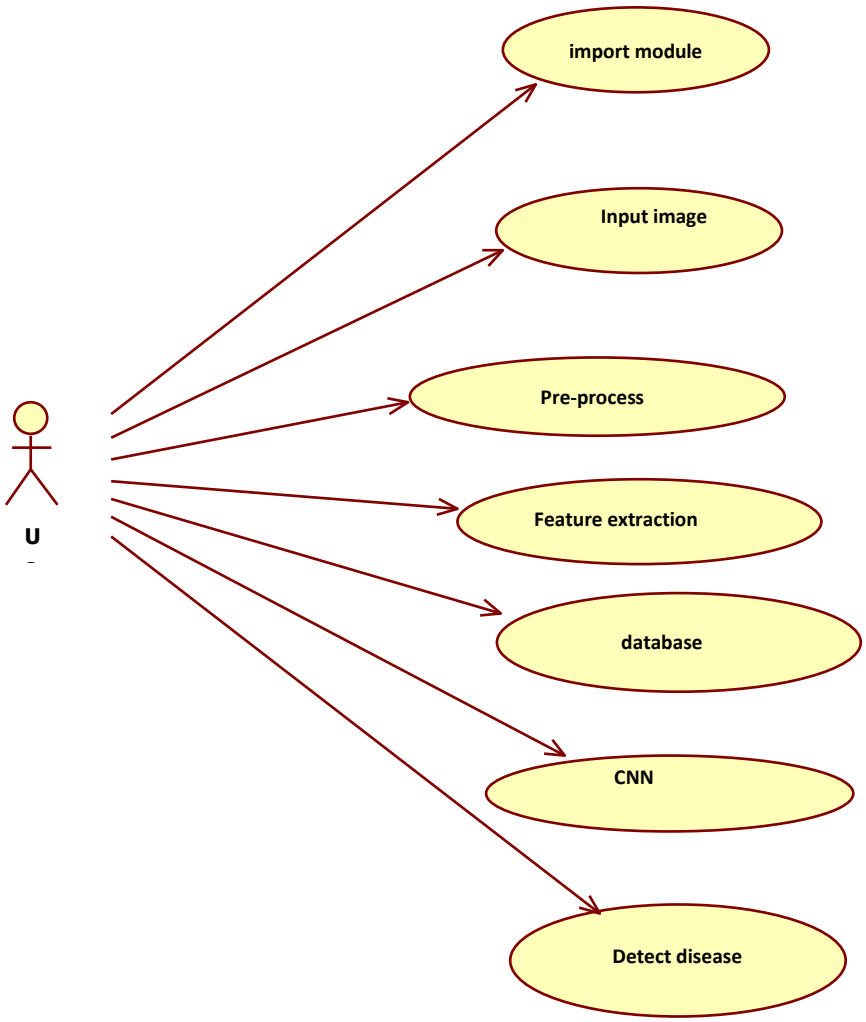
Activity Diagrams-:

- Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling 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.



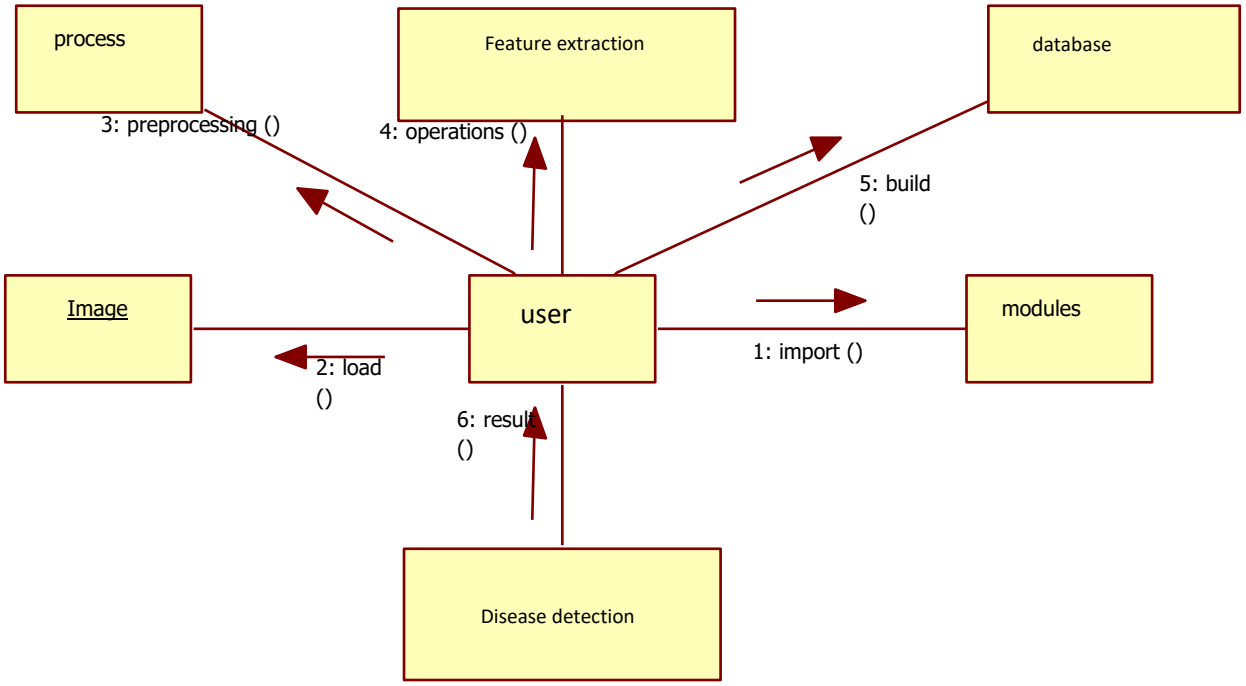
Usecase diagram:

- UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
- UML was created by Object Management Group (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
- UML stands for Unified **M**odeling **L**anguage.
- UML is a pictorial language used to make software blue prints



Collaboration

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior 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



Class diagram

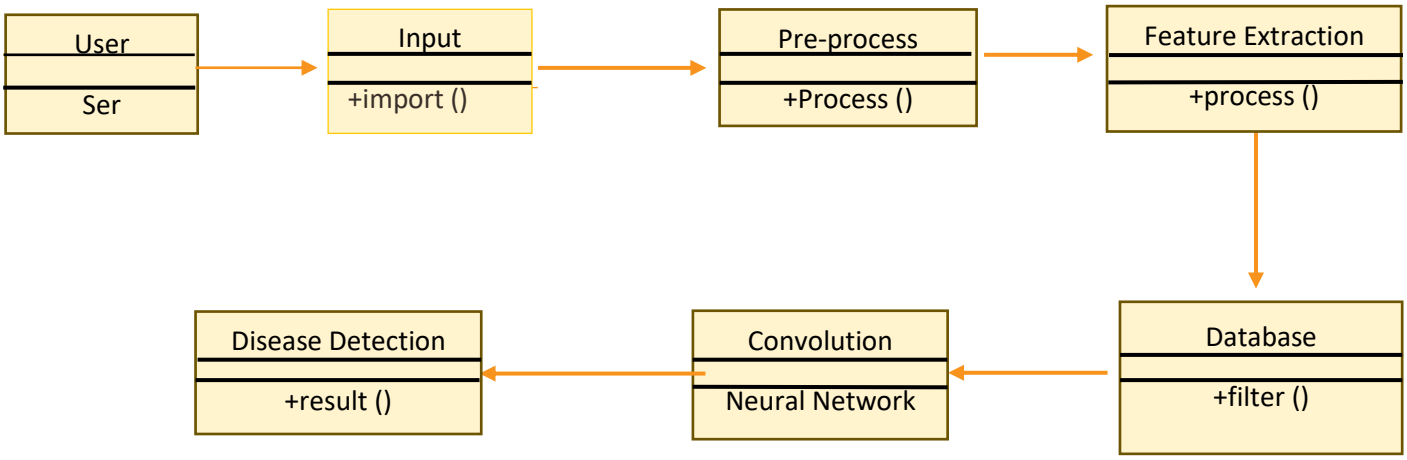
The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class 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 centered, 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.



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.

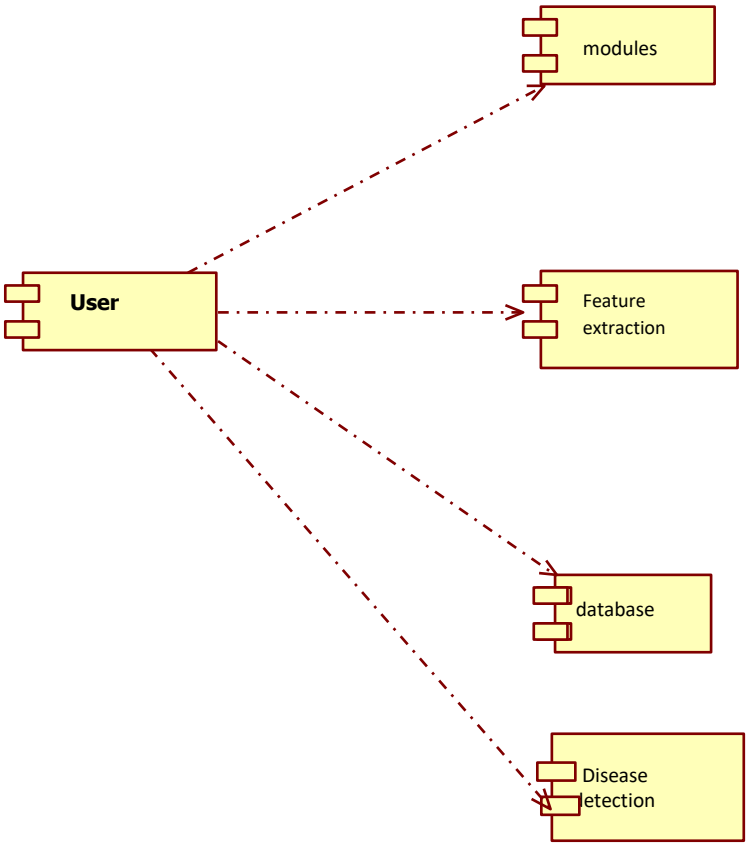
Thus, from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

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.



ER Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.

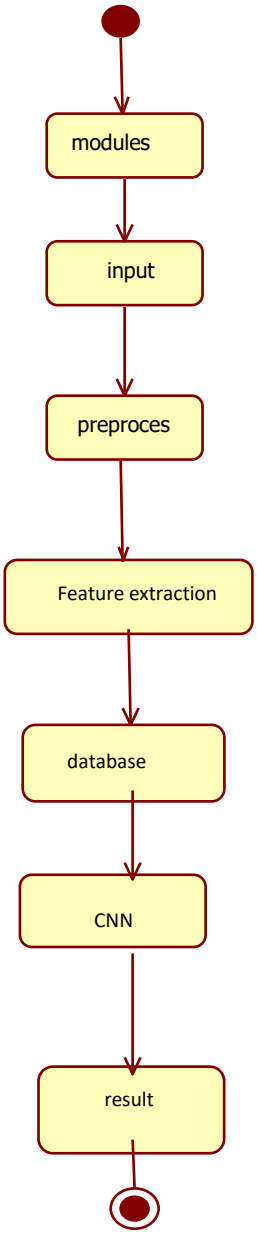
ER diagrams are used to sketch out the design of a database.

Data flow diagram

Also known as DFD, Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow

State Chart



OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correctinformation from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data.

3. The goal of designing input is to make data entry easier and to be free from errors.The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

4. When the data is entered it will check for its validity. Data can be entered with thehelp of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

5.2 Input Design and Output Design

Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things.

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct

source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

IMPLEMENTATION

6.1 MODULES:

Input image acquisition: Image Acquisition? According to Raghava Kashyapa (Machine Vision Expert), In image processing and machine vision, image acquisition is the action of retrieving an image from a source, usually hardware systems like cameras, sensors, etc.

Pre-process: Image preprocessing are the steps taken to format images before they are used by model training and inference. This includes, but is not limited to, resizing, orienting, and color corrections.

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.

Feature analysis: 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. These components are understood as three-dimensional shapes called geons.

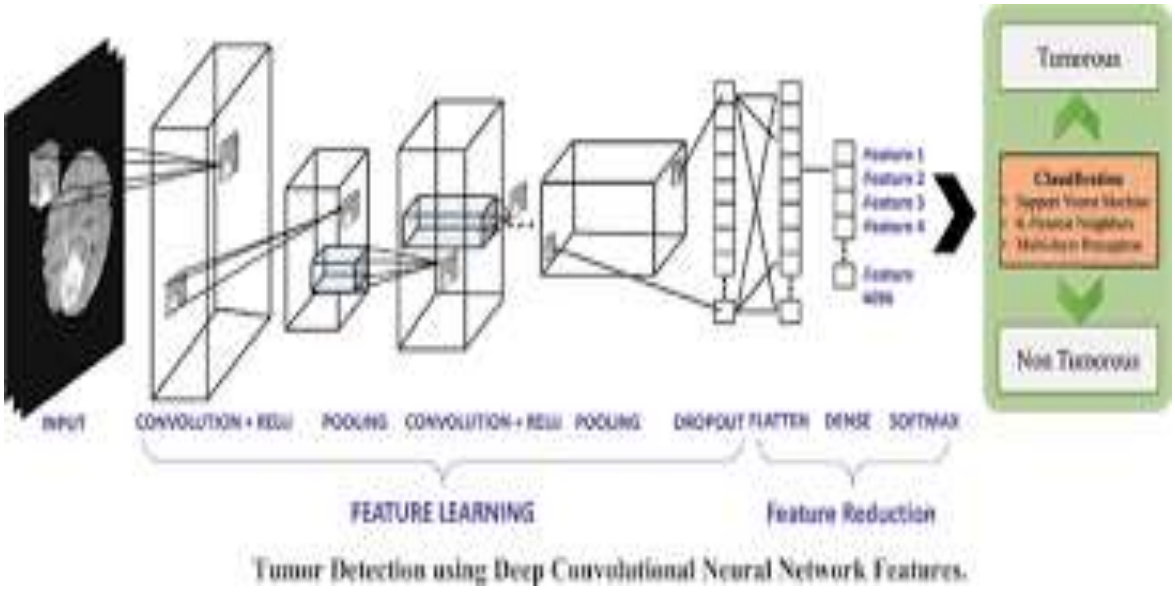
It yields better results than applying machine learning directly to the raw data.

ALGORITHMS:

- **CONVOLUTION NEURAL NETWORK**

INTRODUCTION:

Convolutional neural networks. Sounds like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the most influential innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competitions (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to 15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.



Tumor Detection using Deep Convolutional Neural Network Features.

System description:

Image Acquisition is to acquire a leaf tomato 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 (Gray 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.

RGB TO GRAY:

So, to convert a color image to a grayscale image in OpenCV, we can have two solutions

- Convert image to grayscale with imread() function
- Convert image to grayscale using cvtColor() function

Let's discover how to do it with above mentioned functions...

considerd, that is flagwhich decides the way image is read. There three flags defined in OpenCV...

1. cv2.IMREAD_COLOR or 1
2. cv2.IMREAD_GRAYSCALE or 0
3. cv2.IMREAD_UNCHANGED or -1

So, to convert the color image to grayscale we will be using cv2.imread("image-name.png",0) or you can also write cv2.IMREAD_GRAYSCALE in the place of 0 as it also denotes the same constant.

- resize: mage interpolation occurs when you resize or distort your image from one pixel grid to another. image resizing is necessary when you need to increase or decrease the total number of pixels, whereas remapping can occur when you are correcting for lens distortion or rotating an image. zooming refers to increase the quantity of pixels, so that when you zoom an image, you will see more detail.

interpolation works by using known data to estimate values at unknown points. image interpolation works in two directions, and tries to achieve a best approximation of a pixel's intensity based on the values at surrounding pixels. common interpolation algorithms can be grouped into two categories: adaptive and non-adaptive. adaptive methods change depending on what they are interpolating, whereas non-adaptive methods treat all pixels equally. non-adaptive algorithms include: nearest neighbor, bilinear, bicubic, spline, sinc, lanczos and others. adaptive algorithms include many proprietary algorithms in licensed software such as: qimage, photozoom pro and genuine fractals.

many compact digital cameras can perform both an optical and a digital zoom. a camera performs an optical zoom by moving the zoom lens so that it increases the magnification of light. however, a digital zoom degrades quality by simply interpolating the image. even though the photo with

digital zoom contains the same number of pixels, the detail is clearly far less than with optical zoom. in this lecture zooming and shrinking will be introduced and for this purpose interpolation is introduced and discussed. many various interpolation techniques will be briefly introduced and three of them namely, nearest neighbour, bilinear, and bicubic interpolations will be discussed in more details with visual examples. also required matlab comments for generating the shown examples. will be provided.

Conversion: color image to grayscale image conversion c. saravanan, image processing is a vital research area and the utilization of images increases in various applications.

On different research areas, scientists are working on such as image compression, image restoration, image segmentation etc. To enhance the existing image processing techniques and invent new method of solving image processing problems. the latest image processing applications such as medical image processing, satellite image processing, and molecular image processing uses various image processing techniques. conversion of color image to grayscale image is one of the image processing applications used in different fields effectively. in publication organizations' printing, a color image is expensive compared to a grayscale image. thus, color images have converted to grayscale image to reduce the printing cost for low priced edition books. similarly, color deficient viewer requires good quality of grayscale image to perceive the information, as the normal people perceive the color picture. likewise, various image processing applications require conversion of color image to grayscale image for different purpose. conversion of a color image to a grayscale image requires more knowledge about the color image. a pixel color in an image is a combination of three colors red, green, and blue (rgb). The rgb color values are represented in three dimensions xyz, illustrated by the attributes of lightness, chroma, and hue. quality of a color image depends on the color represented by the number of bits the digital device could support. the basic color image represented by 8 bits, the high color image represented using 16 bits, the true color image represented by 24 bits, and the deep color image is represented by 32 bits. the number of bits decides the maximum number of different colors supported by the digital device. if each red, green, and blue occupies 8 bits then the combination of rgb occupies 24 bit and supports 16,777,216 different colors. the 24 bit represents the color of a pixel in the color image. the grayscale image has represented by luminance using 8 bits value. the luminance of a pixel value of a grayscale image ranges from 0 to 255. the conversion of a color image into a grayscale image is converting the rgb values (24 bit) into grayscale value (8 bit). various image processing techniques and software applications converts color image to grayscale image. however, the image processing techniques or applications are unable to handle the disparity in the chromaticity and the luminance. in the literature, several linear and non-linear techniques had discussed for converting color image to grayscale image. The recent techniques handle these disparities much better than the earlier techniques. nevertheless, the techniques involve several computational procedures such as conversion of rgb space to xyz space then approximations then mapping or other related techniques. Grayscale mappings of color images that are constructed by approximating spectral uniformity are often inadequate.the recent technique used to convert from color image to gray image highly consumes computational time and memory. thus, a new algorithm proposed to convert color image to grayscale image in a minimum amount of time.

There are several issues related to conversion of color image to grayscale image and different solutions to address these issues have addressed in the literature. the software such as adobe photoshop devised custom non-linear projections and required users to set image dependent parameters by trial and error. the following writings discuss recent six prime research works focusing on the conversion of color image to grayscale image. a technique proposed has utilized the l*a*b luminance chrominance representation.the proposed technique introduces an additive correction term for spatial chrominance variations. the first step of this algorithm computes high pass filtered versions of all three channels, and the high-pass content from the two chrominance channels combined into a single signal that represents high frequency chrominance information. another alternative, used in the implementation, is the slightly computationally simpler 1-norm metric. the main feature of this technique is 2010 second international conference on computer engineering and applications 978-0-7695-3982-9/10 $26.00 © 2010 ieee doi 10.1109/iccea.2010.192 196 the same color in the input image can map to different grayscale values depends on the spatial surround. another novel technique proposed to handle fluorescent colors effectively. source color image converted to uniform color space, then target differences were calculated, and finally least squares optimization technique has applied. the experiment shows that the isoluminant colors handled perfectly. the cost of setting up and solving the optimization problem is proportional to the size of the image. the proposed technique is highly resource (time and memory) consumable.

In addition, the technique has not provided large improvements for scenes with high dynamic luminance range like natural scenes. a technique proposed for re-coloring of images for color-deficient viewers without introducing visual artifacts. the mapping of color to grayscale preserves contrasts and maintains luminance consistency.

The quadratic objective function has defined for contrast preservation. further, constraints added to enforce luminance consistency within narrow chrominance bands. the technique performed well for certain images and as standard for other images. another technique proposed enhances the contrast and converts color to grayscale. the proposed technique used gaussian pairing technique for image sampling, dimensionality reduction, and sampling color differences. the predominant component analysis used for analyzing color differences.

The technique has satisfied continuous mapping, global consistency, grayscale preservation, luminance ordering, saturation ordering, and hue ordering.

the process controlled by three parameters: the degree of image enhancement; the typical size of relevant image features in pixels; and the proportion of image pixels assumed to be outliers. first,

the algorithm converts the rgb values into ypq color space. further, to analyze the distribution of color contrasts between image features, color differences between pixels considered using gaussian pairing. dimensionality reduction by predominant component analysis performed to find the color axis that best represents the chromatic contrasts lost when the luminance channel supplies the color to grayscale mapping. next, has combined luminance and chrominance information.

The final step used saturation to calibrate luminance while adjusting its dynamic range and compensatingfor image noise. the decolorize algorithm is effective at enhancing contrast. the algorithm avoids the noise, contouring, and halo artifacts. however, tuning on parameters required individually to suit each image. a recent technique demonstrated color to grayscale conversion based on the experimental background of the coloroid system observations. a survey of the coloroid system to and from cie xyz system formulas completed. observations based on the coloroid system discussed. the seven basic coloroid hues fixed. relative gray-equivalent differences of the basic hue pairs calculated. proposed two formulas based on the cielab color space and the coloroid color space for building the gradient field. further, the inconsistency of gradient field corrected. finally, 2d integration applied to get the grayscale image. from the demonstration noted that the isoluminant colors and bluish colors transformed to grayscale more realistic. the technique preserves overall appearance of the color image. a most recent work converted the color image and video to grayscale. the proposed technique converted the image and video perceptually accurate. first, h-k (helmoltz-kohlrausch) phenomenon predicted by a chromatic lightness term that corrects perceived lightness based on the color's chromatic component. the color image converted to linear rgb by inverse gamma mapping, then transformed to cieluv color space. its apparent chromatic object lightness channel calculated. lightness channel to grayscale values mapped using reference white chromatic values. gamma mapping applied to move from a linear space to a gamma-corrected space. local contrast increased in the grayscale image to represent better the local contrast of original image. the work carried out using cielab and cieluv color spaces. This two-step approaches a good compromise between a fully automatic technique (first step) and user control (second step) making this approach well suited for natural images, photographs, artistic reproductions as well as business graphics. the main limitation of the approach is the locality of the second step. It cannot restore chromatic contrast between non-adjacent regions. end the steps 1 to 3 calculate the luminance and chrominance values of the source color image.

represents the resulted gray color image. the color image, 3d plot of the rgb values of the color image, and 3d plot of the reduced rgb values of the color image have shown and compared. 197

table 1. comparison of source image rgb and reduced rgb values. source color image 3d plot of the source color image rgb values 3d plot of the reduced rgb values the 3d plot of the source image and reduced rgb values compared. the rgb values of the source image ranging from 0 to 255 were reduced to a range of 0 to 85. the reduction enhances the color range and helps to calculate the grayscale in a better way. in the above picture observed that the color ranges are in the 3d plot of the source image rgb values are enhanced many colors are highly visible in the 3d plot of the approximated and reduced rgb values without any major changes in the colors and structure. the reduction process of rgb values retained the major values of the rgb at most of the points observed. the reductions made at the rgb color level so that the resultant grayscale image to retain the luminance and chrominance property at the maximum.the proposed algorithm tested on thirty-four number of different eight-bit color images published in the recent research publications. the color values of the color images are in the range of 0 to 255 for each red, green, and blue. in the images, seven are jpg in format and twenty-seven are png in type. the mat lab software has used for testing the algorithm.

The results of the experiment carefully examined. the difference between the results of the proposed technique and recent techniques identified and discussed. the result revealed that the proposed algorithm yields grayscale image with better luminance and chrominance property for most of the cases and as standard for other few cases. however, there is a minimum amount of loss in the grayscale image due to reduction the algorithm preserved contrasts, sharpness, shadow, and structure of the color image in the reproduced grayscale image. the following table shows four of the different color images used in the experiment and respective reproduced grayscale images using the proposed algorithm. table 2. experiment results color image grayscale image using luminance components in the first grayscale image the letters a, b, and c are sharp as in the color image whereas it is not so, in other earlier techniques. in the second image, the shades of the different color reproduced as sharp as in the color image in the grayscale image than the recent algorithms. the conversion of color image to grayscale image using the proposed algorithm uses approximation of rgb values using luminance rgb components approximated rgb reduced by three, added with chrominance value and average of these four value results perceptually and structurally good quality of grayscale images. First, the luminance and chrominance values are calculated. further, the rgb values of the source color image reduced. finally, the reduced rgb values have added chrominance values.

the resulted grayscale images confirm that the luminance and chrominance properties and structure of the color images 198 retained well in the grayscale image. the results confirm that the isoluminant images have handled as handled by other recent techniques. the proposed

algorithm is helpful for different applications where good quality of grayscale image is highly required. the proposed algorithm results best quality of grayscale images.

Training: Create model of convoltutional neural network using sequential with dense and dropout models. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. We use the 'add ()' function to add layers to our model. Our first 2 layers are Conv2D layers. These are convolution layers that will deal with our input images, which are seen as 2-dimensional matrices. After that sequential add convolutional and max-pooling and flatten layer. onvolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel. After that max pooling layer for extraction of leaf of tomato image. Max pooling is a type of operation that is typically added to CNNs following individual convolutional layers. When added to a model, max pooling reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer. After that flatten layer Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4) Flatten has one argument as follows keras.layers.Flatten(data_format = None). dropout is placed on the fully connected layers only because they are the one with the greater number of parameters and thus, they're likely to excessively co-adapting themselves causing overfitting. The output generated by the dense layer is an 'm' dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also apply operations like rotation, scaling, translation on the vector.

Mobilenet of CNN: MobileNet is a CNN architecture that is much faster as well as a smaller model that makes use of a new kind of convolutional layer, known as Depth wise Separable convolution. Because of the small size of the model, these models are considered very useful.

Dense of CNN: Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer, thus called as dense. Dense Layer is used to classify image based on output from convolutional layers. Working of single neuron. A layer contains multiple number of such neurons. Using above CNN algorithm, we trained dataset and more Python code is simple and easy to run. Here is a simple Python code that will print "Welcome to Python". A simple python example is given below.

Python 3.4 Example

In python 3.4 version, you need to add parenthesis () in a string code to print it.

```
>>> a= ("Welcome to Python Example")
>>> print a
Welcome To Python Example
>>>
```

How to execute python

There are three different ways of working in Python:

1) Interactive Mode:

You can enter python in the command prompt and start working with Python.

Press Enter key and the Command Prompt will appear like:

Now you can execute your Python commands.

2) Script Mode:

Using Script Mode, you can write your Python code in a separate file using any editor of your Operating System.

Save it by .py extension.

Now open Command prompt and execute it by:

3) Using IDE: (Integrated Development Environment)

You can execute your Python code using a Graphical User Interface (GUI).

All you need to do is:

Click on Start button -> All Programs -> Python -> IDLE (Python GUI)

You can use both Interactive as well as Script mode in IDE.

1) Using Interactive mode:

Execute your Python code on the Python prompt and it will display result simultaneously.

2) Using Script Mode:

i) Click on Start button -> All Programs -> Python -> IDLE (Python GUI)

ii) Python Shell will be opened. Now click on File -> New Window.

A new Editor will be opened. Write your Python code here.

Run then code by clicking on Run in the Menu bar.

Run -> Run Module

Result will be displayed on a new Python shell as:

- Opencv:

TECHNOLOGY DESCRIPTION

Basics of Image Processing: -

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: Colour image to gray 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 color.

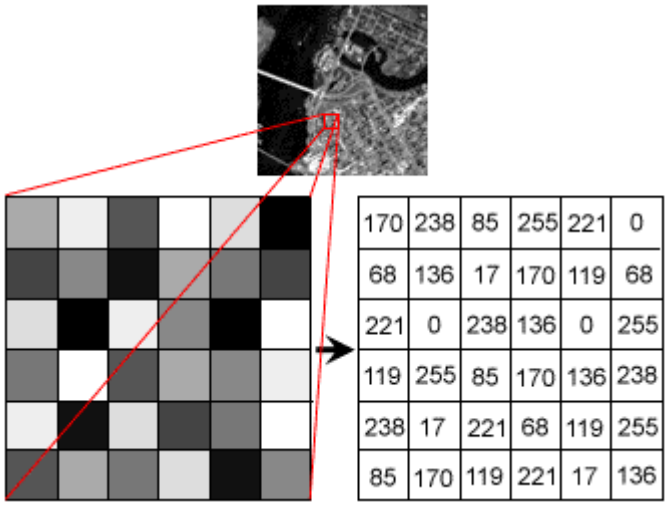


Fig: Gray Scale Image Pixel Value Analysis

Each pixel has a color. The color 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.

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 color 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 color depth increases, an 8-bit pixel (1 byte) stores 256 colors, a 24-bit pixel (3 bytes) stores 16 million colors, the latter known as true color.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 color. For example, an image recorded by a 12 MP camera; since each pixel uses 3 bytes to record true color, 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 images file types. The PNG, JPEG, and GIF formats are most often used to display images on the Internet.

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, mis-connect ions, mis-understandings and mis-information. It is vast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology. It is truly multidisciplinary endeavor ploughed with imprecise jargon.

Several factors 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 (CCDs) for digitizing, storage during processing and display and large low cost of image storage arrays.

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 color 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: Digital camera

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

Fig: 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 Gray 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: 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 deblurring function is considered a restoration technique.

Color image processing:

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

Fig: gray Scale image Colour Image

INTRODUCTION TO COMPUTER VISION

Using software to parse the world's visual content is as big of a revolution in computing as mobile was 10 years ago, and will provide a major edge for developers and businesses to build amazing products.

Computer Vision is the process of using machines to understand and analyze imagery (both photos and videos). While these types of algorithms have been around in various forms since the 1960's, recent advances in Machine Learning, as well as leaps forward in data storage, computing capabilities, and cheap high-quality input devices, have driven major improvements in how well our software can explore this kind of content.

What is Computer Vision?

Computer Vision is the broad parent's name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. But within this parent idea, there are a few specific tasks that are core building blocks:

- In object classification, you train a model on a dataset of specific objects, and the model classifies new objects as belonging to one or more of your training categories.
- For object identification, your model will recognize a specific instance of an object – for example, parsing two faces in an image and tagging one as Tom Cruise and one as Katie Holmes.

A classical application of computer vision is handwriting recognition for digitizing handwritten content (we'll explore more use cases below). Outside of just recognition, other methods of analysis include:

- Video motion analysis uses computer vision to estimate the velocity of objects in a video, or the camera itself.
- In image segmentation, algorithms partition images into multiple sets of views.
- Scene reconstruction creates a 3D model of a scene inputted through images or video (check out Selva).
- In image restoration, noise such as blurring is removed from photos using Machine Learning based filters.

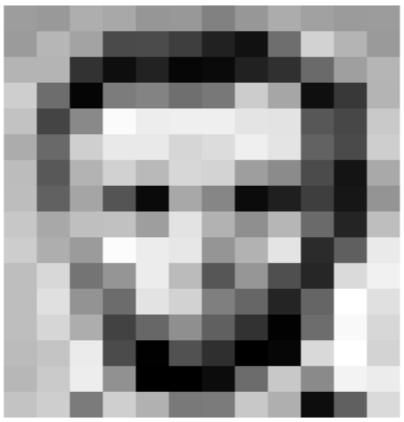
Any other application that involves understanding pixels through software can safely be labeled as computer vision.

How Computer Vision Works

One of the major open questions in both Neuroscience and Machine Learning is: how exactly do our brains work, and how can we approximate that with our own algorithms? The reality is that there are very few working and comprehensive theories of brain computation; so, despite the fact that Neural Nets are supposed to "mimic the way the brain works," nobody is quite sure if that's actually true. Jeff Hawkins has an entire book on this topic called On Intelligence.

The same paradox holds true for computer vision – since we're not decided on how the brain and eyes process images, it's difficult to say how well the algorithms used in production approximate our own internal mental processes. For example, studies have shown that some functions that we thought happen in the brain of frogs actually take place in the eyes. We're a far cry from amphibians, but similar uncertainty exists in human cognition.

Machines interpret images very simply: as a series of pixels, each with their own set of color values. Consider the simplified image below, and how grayscale values are converted into a simple array of numbers:



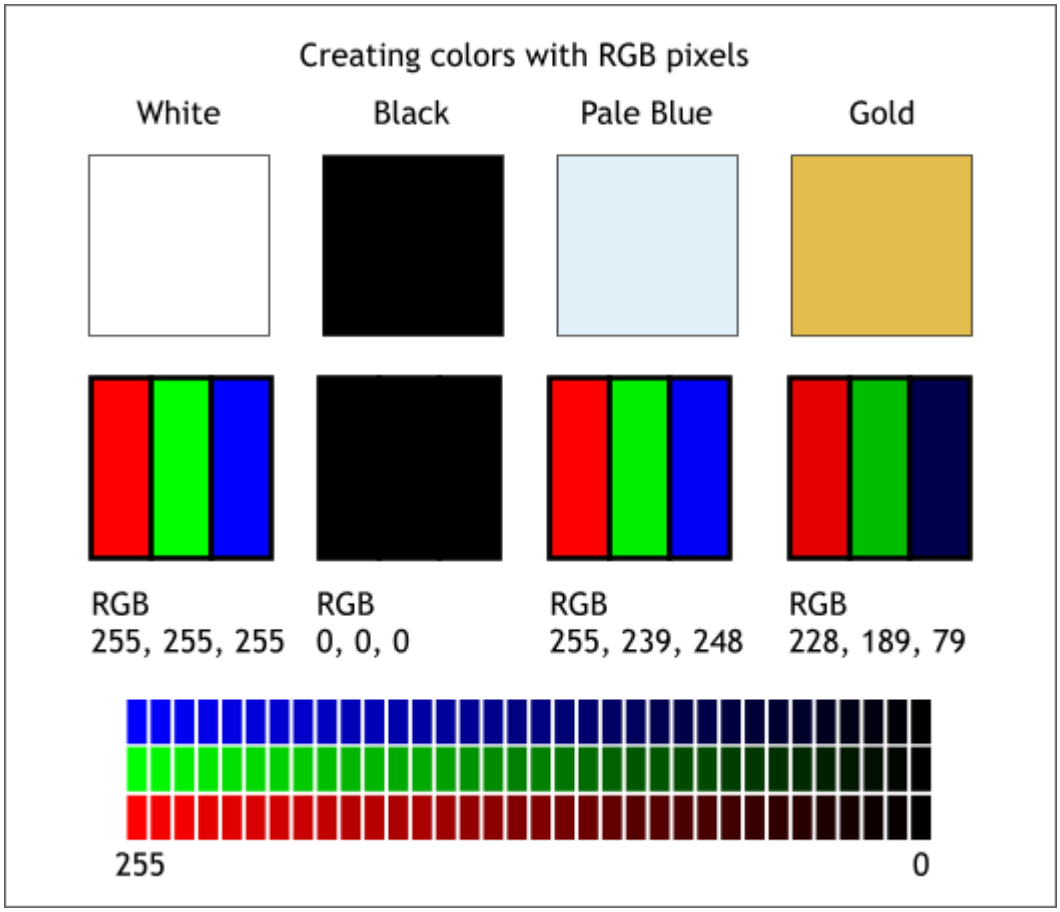
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Source: Openframeworks

Think of an image as a giant grid of different squares, or pixels (this image is a very simplified version of what looks like either Abraham Lincoln or a Dementor). Each pixel in an image can be represented by a number, usually from 0 – 255. The series of numbers on the right is what software sees when you input an image. For our image, there are 12 columns and 16 rows, which means there are 192 input values for this image.

When we start to add in color, things get more complicated. Computers usually read color as a series of 3 values – red, green, and blue (RGB) – on that same 0 – 255 scale. Now, each pixel actually has 3 values for the computer to store in addition to its position. If we were to colorize President Lincoln (or Harry Potter's worst fear), that would lead to 12 x 16 x 3 values, or 576 numbers.



Source: Xaraxone

For some perspective on how computationally expensive this is, consider this tree:

- Each color value is stored in 8 bits.
- 8 bits x 3 colors per pixel = 24 bits per pixel.
- A normal sized 1024 x 768 image x 24 bits per pixel = almost 19M bits, or about 2.36 megabytes.

That's a lot of memory to require for one image, and a lot of pixels for an algorithm to iterate over. But to train a model with meaningful accuracy – especially when you're talking about Deep Learning – you'd usually need tens of thousands of images, and the more the merrier. Even if you were to use Transfer Learning to use the insights of an already trained model, you'd still need a few thousand images to train yours on.

With the sheer amount of computing power and storage required just to train deep learning models for computer vision, it's not hard to understand why advances in those two fields have driven Machine Learning forward to such a degree.

Business Use Cases for Computer Vision

Computer vision is one of the areas in Machine Learning where core concepts are already being integrated into major products that we use every day. Google is using maps to leverage their image data and identify street names, businesses, and office buildings. Facebook is using computer vision to identify people in photos, and do a number of things with that information.

But it's not just tech companies that are leverage Machine Learning for image applications. Ford, the American car manufacturer that has been around literally since the early 1900's, is investing heavily in autonomous vehicles (AVs). Much of the underlying technology in AVs relies on analyzing the multiple video feeds coming into the car and using computer vision to analyze and pick a path of action.

Another major area where computer vision can help is in the medical field. Much of diagnosis is image processing, like reading x-rays, MRI scans, and other types of diagnostics. Google has been working with medical research teams to explore how deep learning can help medical workflows, and have made significant progress in terms of accuracy. To paraphrase from their research page:

"Collaborating closely with doctors and international healthcare systems, we developed a state-of-the-art computer vision system for reading retinal fundus images for diabetic retinopathy and determined our algorithm's performance is on par with U.S. board-certified ophthalmologists. We've recently published some of our research in the Journal of the American Medical Association and summarized the highlights in a blog post."

But aside from the groundbreaking stuff, it's getting much easier to integrate computer vision into your own applications. A number of high-quality third-party providers like Clarifai offer a simple API for tagging and understanding images, while Kairos provides functionality around facial recognition. We'll dive into the open-source packages available for use below.

Computer Vision on Algorithmia

Algorithmia makes it easy to deploy computer vision applications as scalable microservices. Our marketplace has a few algorithms to help get the job done:

- SalNet automatically identifies the most important parts of an image
- Nudity Detection detects nudity in pictures
- Emotion Recognition parses emotions exhibited in images
- DeepStyle transfers next-level filters onto your image
- Face Recognition…recognizes faces.
- Image Memorability judges how memorable an image is.

A typical workflow for your product might involve passing images from a security camera into Emotion Recognition and raising a flag if any aggressive emotions are exhibited, or using Nudity Detection to block inappropriate profile pictures on your web application.

For a more detailed exploration of how you can use the Algorithmia platform to implement complex and useful computer vision tasks,

Computer Vision Resources

Packages and Frameworks

OpenCV – "OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics."

SimpleCV – "SimpleCV is an open-source framework for building computer vision applications. With it, you get access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap storage."

Mahotas – “Mahotas is a computer vision and image processing library for Python. It includes many algorithms implemented in C++ for speed while operating in numpy arrays and with a very clean Python interface. Mahotas currently has over 100 functions for image processing and computer vision and it keeps growing.

7.1NUMPY:

- NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.
- Audience
- This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.
- Prerequisites
- You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.
- NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open-source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

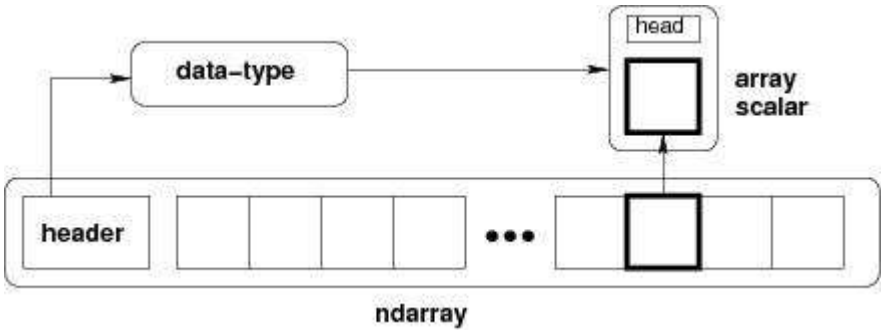
NumPy is often used along with packages like SciPy (Scientific Python) and Mat−plotlib (plotting library). This combination is widely used as a replacement for

MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in a ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type −



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
```

It creates a ndarray from any object exposing array interface, or from any method that returns an array.

Imutils:A series of convenience functions to make basic image processing operations such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python.

Transalation

Translation is the shifting of an image in either the x or y direction. To translate an image in OpenCV you need to supply the (x, y)-shift, denoted as (t_x, t_y) to construct the translation matrix M:

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

And from there, you would need to apply the cv2.warpAffine function.

Instead of manually constructing the translation matrix M and calling cv2.warpAffine, you can simply make a call to the translate function of imutils

Rotation care has to be taken to supply the (x, y)-coordinate of the point the image is to be rotated about. These calculation calls can quickly add up and make your code bulky and less readable. The rotate function inimutils

Rotating an image in OpenCV is accomplished by making a call tocv2.getRotationMatrix2D and cv2.warpAffine. Further helps resolve this problem.

Resizing

Resizing an image in OpenCV is accomplished by calling the cv2.resize function. However, special care needs to be taken to ensure that the aspect ratio is maintained. Thisresize function of imutils maintains the aspect ratio and provides the keyword arguments width and height so the image can be resized to the intended width/height while (1) maintaining aspect ratio and (2) ensuring the dimensions of the image do not have to be explicitly computed by the developer.

Another optional keyword argument, inter, can be used to specify interpolation method as well.

Skeletonization is the process of constructing the “topological skeleton” of an object in an image, where the object is presumed to be white on a black background. OpenCV does not provide a function to explicity construct the skeleton, but does provide the morphological and binary functions to do so.

For convenience, the skeletonize function of imutils can be used to construct the topological skeleton of the image.The first argument, size is the size of the structuring element kernel. An optional argument,structuring , can be used to control the structuring element — it defaults tocv2.MORPH_RECT , but can be any valid structuring element.

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms,basic linear algebra, basic statistical operations, random simulation and much more. At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences: • NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original. • The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements. • NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using python's built-in sequences. A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays. The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, a and b, we could iterate over each element.

The Numeric Python extensions (NumPy henceforth) is a set of extensions to the Python programming language which allows Python programmers to efficiently manipulate large sets of objects organized in grid-like fashion. These sets of objects are called arrays, and they can have any number of dimensions: one dimensional array is similar to standard Python sequences, two-dimensional arrays are similar to matrices from linear algebra. Note that one-dimensional arrays are also different from any other Python sequence, and that two-dimensional matrices are also different from the matrices of linear algebra, in ways which we will mention later in this text. Why are these extensions needed? The core reason is a very prosaic one, and that is that anipulating a set of a million numbers in Python with the standard data structures such as lists,

tuples or classes is much too slow and uses too much space. Anything which we can do in NumPy we can do in standard Python – we just may not be alive to see the program finish. A more subtle reason for these extensions however is that the kinds of operations that programmers typically want to do on arrays, while sometimes very complex, can often be decomposed into a set of fairly standard operations. This decomposition has been developed similarly in many array languages. In some ways, NumPy is simply the application of this experience to the Python language – thus many of the operations described in NumPy work the way they do because experience has shown that way to be a good one, in a variety of contexts. The languages which were used to guide the development of NumPy include the infamous APL family of languages, Basis, MATLAB, FORTRAN, S and S+, and others. This heritage will be obvious to users of NumPy who already have experience with these other languages. This tutorial, however, does not assume any such background, and all that is expected of the reader is a reasonable working knowledge of the standard Python language. This document is the "official" documentation for NumPy. It is both a tutorial and the most authoritative source of information about NumPy with the exception of the source code. The tutorial material will walk you through a set of manipulations of simple, small, arrays of numbers, as well as image files. This choice was made because:

• Aconcrete data

set makes explaining the behavior of some functions much easier to motivate than simply talking about abstract operations on abstract data sets; • Every reader will at least an intuition as to the meaning of the data and organization of image files, and • The result of various manipulations can be displayed simply since the data set has a natural graphical representation. All users of NumPy, whether interested in image processing or not, are encouraged to follow the tutorial with a working NumPy installation at their side, testing the examples, and, more importantly, transferring the understanding gained by working on images to their specific domain. The best way to learn is by doing – the aim of this tutorial is to guide you along this "doing.

7.2 DISPLAYING WITH MATPLOTLIB

In the Python bindings of OpenCV, images are represented as NumPy arrays in BGR order. This works fine when using the cv2.imshow function. However, if you intend on using Matplotlib, the plt.imshow function assumes the image is in RGB order. A simple call tocv2.cvtColor will resolve this problem, or you can use the opencv2matplotlib convenience function.

Tensorflow:

But before we look at the steps, a quick look at what is Tensorflow. TensorFlow is basically a free and open-source software library for dataflow and differentiable programming across a variety of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. Developed by the Google Brain team, it is being used for both research and production at Google. Tensorflow can be installed either through CPU installation or GPU installation.

Below are the instructions to be followed for installing Tensorflow CPU on a Windows computer:

If Python is installed, open IDLE and goto command prompt type pip install tensorflow

Twilio is a customer engagement platform used by hundreds of thousands of businesses and more than ten million developers worldwide to build unique, personalized experiences for their customers.

The most famous deep learning library in the world is Google's TensorFlow. 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 TensorFlow was built to scale. TensorFlow is a library developed by the Google Brain 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

TensorFlow Architecture

Tensorflow architecture works in three parts:

- Preprocessing the data
- Build the model
- Train and estimate the model

It is called Tensorflow 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 TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Where can Tensorflow run?

TensorFlow 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 Tensorflow can be run on many different platforms. You can run it on

- Desktop running Windows, macOS 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. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of TensorFlow is the TensorBoard. The TensorBoard enables to monitor graphically and visually what TensorFlow is doing.

List of Prominent Algorithms supported by TensorFlow

- Linear regression: tf.estimator.LinearRegressor
- Classification:tf.estimator.LinearClassifier
- Deep learning classification: tf.estimator.DNNClassifier
- Deep learning wipe and deep: tf.estimator.DNNLinearCombinedClassifier
- Booster tree regression: tf.estimator.BoostedTreesRegressor
- Boosted tree classification: tf.estimator.BoostedTreesClassifier

7.3 PYTHON OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python Features

Python's features include:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly

- Easy-to-read: Python code is more clearly defined and visible to the eyes.

- Easy-to-maintain: Python's source code is fairly easy-to-maintain.

- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- Databases: Python provides interfaces to all major commercial databases.

- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- IT supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.

- IT supports automatic garbage collection.

- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

History and Versions:

Python is predominantly a dynamic typed programming language which was initiated by Guido van Rossum in the year 1989. The major design philosophy that was given more importance was the readability of the code and expressing an idea in fewer lines of code rather than the verbose way of expressing things as in C++ and Java [K-8] [K-9]. The other design philosophy that was worth mentioning was that, there should be always a single way and a single obvious way to express a given task which is contradictory to other languages such as C++, Perl etc. [K-10]. Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The initial versions of Python were heavily

inspired from lisp (for functional programming constructs). Python had heavily borrowed the module system, exception model and also keyword arguments from Modula-3 language [K-10]. Pythons' developers strive not to entertain premature optimization, even though it might increase the performance by a few basis points [K-9]. During its design, the creators had conceptualized the language as being a very extensible language, and hence they had designed the language to have a small core library which was extended by a huge standard library [K-7]. Thus, as a result, python is used as a scripting language as it can be easily embedded into any application, though it can be used to develop a full-fledged application. The reference implementation of python is CPython. There are also other implementations like Jython, Iron Python which can use python syntax as well as can use any class of Java (Jython) or .Net class (Iron Python). Versions: Python has two versions 2.x version and 3.x version. The 3.x version is a backward incompatible release was released to fix many design issues which plagued the 2.x series. The latest in the 2.x series is 2.7.6 and the latest in 3.x series is 3.4.0. 1.5.2 Paradigms: Python supports multi-paradigms such as: Object-Oriented, Imperative, Functional, Procedural, and Reflective. In Object-Oriented Paradigm, Python supports most of the OOP's concepts such as Inheritance (It also has support for Multiple Inheritance), Polymorphism but its lack of support for encapsulation is a blatant omission as Python doesn't have private, protected members: all class members are public [K-11]. Earlier Python 2.6 versions didn't support some OOP's concepts such as Abstraction through Interfaces and Abstract Classes [K-19]. It also supports Concurrent paradigm, but with Python we will not be able to make truly multitasking applications as the inbuilt threading API is limited by GIL (Global Interpreter Lock) and hence applications that use the threading API cannot run on multi-core parallelly [K-12].The only remedy is that, the user has to either use the multi-processing module which would fork processes or use Interpreters that haven't implemented GIL such as Jython or Iron Python [K-12]. 1.5.3 Compilation, Execution and Memory Management: Compilation, Execution and Memory Management: 21 A Comparative Studies of Programming Languages (Comparative Studies of Six Programming Language) Just like the other Managed Languages, Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The reference implementation (i.e., CPython) doesn't come with a JIT compiler because of which the execution speed is slow compared to native programming languages [K-17]. We can use PyPy interpreter as it includes a JIT compiler rather than using the Python interpreter that comes by default with the python language, if speed of execution is one of the important factors [K-18]. The Python Runtime Environment also takes care of all the allocation and deallocation of memory through the Garbage Collector. When a new object is created, the GC allocates the necessary memory,

and once the object goes out of its scope, the GC doesn't release memory immediately but instead it becomes eligible for Garbage Collection, which would eventually release the memory. Typing

Strategies: Python is a strongly dynamic typed language. Python 3 also supports optional static typing [K-20]. There are a few advantages in using a dynamic typed language, the most prominent one would be that the code is more readable as there is less code (in other words has less boiler-plate code). But the main disadvantage in having python as a dynamic programming language is that there would be no way to guarantee that a particular piece of code would run successfully for all the different data-types scenarios simply because it had run successfully with one type. Basically, we don't have any means to find out an error in the code, till the code has started running. 1.5.4 Strengths and Weaknesses and Application Areas: Python is predominantly used as a scripting language used in developing standalone applications that are being developed with Static-Typed languages, because of the flexibility it provides due to its dynamic typed nature. Python favours rapid application development, which qualifies it to be used for prototyping. To a certain extent, Python is also used in developing websites. Due to its dynamic typing and of the presence of a Virtual Machine, there is a considerable overhead which translates to weigh less performance when we compare with native programming languages [K-13]. And hence it is not suited

7.4 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the

packages you want, install them in an environment, run the packages and update them, all inside Navigator.

WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QTConsole
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

Advanced conda users can also build your own Navigator applications

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What’s new in 1.9?

- Add support for Offline Mode for all environment related actions.
- Add support for custom configuration of main windows links.
- Numerous bug fixes and performance enhancements.

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to soitware development and scientific applications.

It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

CODING

```
import numpy as np # linear algebra

import os

import cv2

import random

import shutil

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

#import shutil

from glob import glob

# Helper libraries

import matplotlib.pyplot as plt

#import math

%matplotlib inline

#print(tf. __version__)

data_root='C:/Users/B ARUNREDDY/Music/liver disease/dataset'

path_abnormal_image = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal')

path_normal_image = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset/normal')

DataSets Overview

# jpg and png files

abnormal_images_ls = glob(os.path.join(path_abnormal_image,"*.jpg"))

normal_images_ls = glob(os.path.join(path_normal_image,"*.jpg"))

normal_images_ls.extend(glob(os.path.join(path_normal_image,"*.jpg")))
```

```
len(abnormal_images_ls)

len(normal_images_ls)

46

abnormal = {'class': 'abnormal',

        'path': path_abnormal_image,

        'images': abnormal_images_ls}

normal = {'class': 'normal',

            'path': path_normal_image,

            'images': normal_images_ls}

total_abnormal_image = len(abnormal_images_ls)

total_normal_image = len(normal_images_ls)

print("Total path_abnormal_image: {}".format(total_abnormal_image))

print("Total path_normal_image: {}".format(total_normal_image))

Total path_abnormal_image: 52

Total path_normal_image: 46

image_abnormal = cv2.imread(os.path.join(abnormal_images_ls[1]))

image_normal = cv2.imread(os.path.join(normal_images_ls[5]))

operatedImage = cv2.cvtColor(image_normal, cv2.COLOR_BGR2GRAY)


# modify the data type

# setting to 32-bit floating point

operatedImage = np.float32(operatedImage)

# apply the cv2.cornerHarris method

# to detect the corners with appropriate

# values as input parameters
```

```
dest = cv2.cornerHarris(operatedImage, 2, 5, 0.07)
f = plt.figure(figsize=(8, 8))
f.add_subplot(1, 2, 1)
plt.imshow(image_normal)
f.add_subplot(1,2, 2)
plt.imshow(image_abnormal)
```

```
<matplotlib.image.AxesImage at 0x20f79aecbd0>
```



```
<Figure size 800x800 with 2 Axes>
```

```
print("Image abnormal Shape {}".format(image_abnormal.shape))
print("Image normal Shape {}".format(image_normal.shape))
```

```
Image abnormal Shape (210, 240, 3)
Image normal Shape (195, 259, 3)
```

Create Train-Test Directory

```
# Create Train-Test Directory
subdirs  = ['train/', 'test/']
for subdir in subdirs:
    labeldirs = ['abnormal', 'normal']
    for labldir in labeldirs:
        newdir = subdir + labldir
        os.makedirs(newdir, exist_ok=True)
        print(newdir)
```

```
train/abnormal
train/normal
test/abnormal
test/normal
```

```
# Copy Images to test set

# seed random number generator

random.seed(237)

# define ratio of pictures used for testing

test_ratio = 0.2

for cases in [abnormal, normal]:

    total_cases = len(cases['images']) #number of total images

    num_to_select = int(test_ratio * total_cases) #number of images to copy to test set

    print(cases['class'], num_to_select)

    list_of_random_files = random.sample(cases['images'], num_to_select) #random files selected

 for files in list_of_random_files:

        shutil.copy2(files, 'test/' + cases['class'])

abnormal 10

normal 9

# Copy Images to train set

for cases in [abnormal, normal]:

    image_test_files = os.listdir('test/' + cases['class']) # list test files

    for images in cases['images']:

        if images.split('/')[-1] not in (image_test_files): #exclude test files from shutil.copy

            shutil.copy2(images, 'train/' + cases['class'])

total_train_abnormal = len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal'))

total_train_normal = len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/normal'))
```

```
total_test_abnormal= len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal'))

total_test_normal = len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/normal'))

print("Train sets images abnormal: {}".format(total_train_abnormal))

print("Train sets images normal: {}".format(total_train_normal))

print("Test sets images abnormal: {}".format(total_test_abnormal))

print("Test sets images normal: {}".format(total_test_normal))
```

```
Train sets images abnormal: 52

Train sets images normal: 23

Test sets images abnormal: 52

Test sets images normal: 23
```

CNN Model

```
batch_size =15

epochs = 10

IMG_HEIGHT = 150

IMG_WIDTH = 150

train_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our training data

test_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our validation data

train_dir = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset')

test_dir = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset')

total_train = total_train_abnormal + total_train_normal

total_test = total_test_abnormal + total_test_normal

train_data_gen = train_image_generator.flow_from_directory(batch_size=batch_size,

                                                           directory=train_dir,
```

```
shuffle=True, target_size=(IMG_HEIGHT, IMG_WIDTH), class_mode=('binary')

Found 75 images belonging to 2 classes.

test_data_gen = test_image_generator.flow_from_directory(batch_size=batch_size,
                                                         directory=test_dir,
                                                         target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                         class_mode='binary')

Found 75 images belonging to 2 classes.

model = Sequential([
    Conv2D(32, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2, 2),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.summary()

Model: "sequential"

Layer (type)                 Output Shape              Param #
```

conv2d (Conv2D) (None, 150, 150, 32) 896

max_pooling2d (MaxPooling2 (None, 75, 75, 32) 0

D)

conv2d_1 (Conv2D) (None, 75, 75, 64) 18496

max_pooling2d_1 (MaxPoolin (None, 37, 37, 64) 0

g2D)

conv2d_2 (Conv2D) (None, 37, 37, 64) 36928

max_pooling2d_2 (MaxPoolin (None, 18, 18, 64) 0

g2D)

flatten (Flatten) (None, 20736) 0

dense (Dense) (None, 512) 10617344

dense_1 (Dense) (None, 1) 513

...

Total params: 10674177 (40.72 MB)

Trainable params: 10674177 (40.72 MB)

Non-trainable params: 0 (0.00 Byte)

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=test_data_gen,
    validation_steps=total_test // batch_size
)Epoch 1/10
```

C:\Users\B ARUNREDDY\AppData\Local\Temp\ipykernel_18284\2517996987.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

history = model.fit_generator(

5/5 [==============================] - 4s 518ms/step - loss: 0.7212 - accuracy: 0.6667 - val_loss: 0.6247 - val_accuracy: 0.6933

Epoch 2/10

5/5 [==============================] - 2s 418ms/step - loss: 0.5953 - accuracy: 0.6933 - val_loss: 0.5095 - val_accuracy: 0.6933

Epoch 3/10

5/5 [==============================] - 2s 388ms/step - loss: 0.4780 - accuracy: 0.6933 - val_loss: 0.3987 - val_accuracy: 0.8667

Epoch 4/10

5/5 [==============================] - 2s 396ms/step - loss: 0.4008 - accuracy: 0.8533 - val_loss: 0.4086 - val_accuracy: 0.7333

Epoch 5/10

5/5 [==============================] - 2s 390ms/step - loss: 0.3376 - accuracy: 0.7733 - val_loss: 0.2502 - val_accuracy: 0.9200

Epoch 6/10

5/5 [==============================] - 2s 392ms/step - loss: 0.2232 - accuracy: 0.9600 - val_loss: 0.1236 - val_accuracy: 0.9467

Epoch 7/10

5/5 [==============================] - 2s 390ms/step - loss: 0.1091 - accuracy: 0.9467 - val_loss: 0.0588 - val_accuracy: 0.9733

Epoch 8/10

5/5 [==============================] - 2s 391ms/step - loss: 0.0531 - accuracy: 0.9867 - val_loss: 0.0166 - val_accuracy: 1.0000

Epoch 9/10

```
5/5 [==============================] - 2s 376ms/step - loss: 0.0114 - accuracy: 1.0000 - val_loss: 0.0043 - val_accuracy: 1.0000

Epoch 10/10

5/5 [==============================] - 2s 380ms/step - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.0012 - val_accuracy: 1.0000

import matplotlib.image as mpimg

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing import image

img=mpimg.imread('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal/download (13).jpg')

imgplot = plt.imshow(img, interpolation='none')

test_image = image.load_img('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal/download (13).jpg', target_size = (150, 150))

#C:\Users\HP\Desktop\covid 19\dataset\normal

test_image = image.img_to_array(test_image)

test_image = np.expand_dims(test_image, axis = 0)

result = model.predict(test_image)

print(result)

if result>=1:

   print( 'normal')

else:

   print( 'LIVER DISEASE')

1/1 [==============================] - 0s 124ms/step

[[-1560.253]LIVER DISEASE

<Figure size 640x480 with 1 Axes>
```

SYSTEM TESTING

TESTING METHODOLOGIES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that theSoftware system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests,

must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g., Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

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 number 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.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

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.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

TESTING STRATEGIES

UNIT TESTING

1.Unit is smallest compatible component.

2.In unit testing called communicating components are replaced with stubs, simulators, or trusted companies

3.Calling components are replaced with drivers or trusted super-components.

4.The unit is tested in isolation.

TEST CASES

Test case1:(packages testing)

Input: downloading packages in interactive mode

Output: download proper tools

Test case2: (load input)

Input: load input dataset images of liver disease from medical images

Output: read input data in images of natural pesticide

Test case3:(take steps for using pre-process)

Input: take input process from the images of liver disease from medical images.

Output: pre-process to do resize images of liver disease from medical images.

Test case 4:(get the features)

Input: get feature from dataset and compare

Output: got feature from input and data base features compared.

Test case 5:(take input and predict)

Input: take input and check output

Output: selected input and checked out liver disease or normal.

OUTPUT SCREENS

In [1]:
```
import numpy as np # linear algebra

import os
import cv2
import random
import shutil
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

#import shutil
from glob import glob
# Helper libraries
import matplotlib.pyplot as plt
#import math
%matplotlib inline
#print(tf.__version__)
```

In [2]:
```
data_root='C:/Users/B ARUNREDDY/Music/liver disease/dataset'
path_abnormal_image = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal')
path_normal_image = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset/normal')
```

Datasets Overview

In [3]:
```
# jpg and png files
abnormal_images_ls = glob(os.path.join(path_abnormal_image,"*.jpg"))

normal_images_ls = glob(os.path.join(path_normal_image,"*.jpg"))
normal_images_ls.extend(glob(os.path.join(path_normal_image,"*.jpg")))
len(abnormal_images_ls)
len(normal_images_ls)
```

Out[3]: 46

jupyter liver disease Last Checkpoint: 08/07/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
len(normal_images_ls)
```

Out[3]: 46

In [4]:
```
abnormal = {'class': 'abnormal',
            'path': path_abnormal_image,
            'images': abnormal_images_ls}

normal = {'class': 'normal',
              'path': path_normal_image,
              'images': normal_images_ls}
```

In [5]:
```
total_abnormal_image = len(abnormal_images_ls)
total_normal_image = len(normal_images_ls)
print("Total path_abnormal_image: {}".format(total_abnormal_image))
print("Total path_normal_image: {}".format(total_normal_image))
```

```
Total path_abnormal_image: 52
Total path_normal_image: 46
```

In [6]:
```
image_abnormal = cv2.imread(os.path.join(abnormal_images_ls[1]))
image_normal = cv2.imread(os.path.join(normal_images_ls[5]))
operatedImage = cv2.cvtColor(image_normal, cv2.COLOR_BGR2GRAY)

# modify the data type
# setting to 32-bit floating point
operatedImage = np.float32(operatedImage)

# apply the cv2.cornerHarris method
# to detect the corners with appropriate
# values as input parameters
dest = cv2.cornerHarris(operatedImage, 2, 5, 0.07)

f = plt.figure(figsize=(8, 8))
f.add_subplot(1, 2, 1)
plt.imshow(image_normal)
f.add_subplot(1,2, 2)
plt.imshow(image_abnormal)
```

Out[6]: <matplotlib.image.AxesImage at 0x20f79aecbd0>

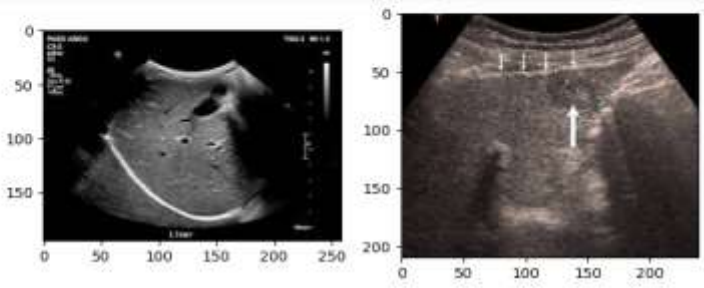
jupyter liver disease Last Checkpoint: 08/07/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code



In [7]:
```
print("Image abnormal Shape {}".format(image_abnormal.shape))
print("Image normal Shape {}".format(image_normal.shape))
```

```
Image abnormal Shape (210, 240, 3)
Image normal Shape (195, 259, 3)
```

Create Train-Test Directory

In [8]:
```
# Create Train-Test Directory
subdirs  = ['train/', 'test/']
for subdir in subdirs:
    labeldirs = ['abnormal', 'normal']
    for labldir in labeldirs:
        newdir = subdir + labldir
        os.makedirs(newdir, exist_ok=True)
        print(newdir)
```

```
train/abnormal
train/normal
test/abnormal
test/normal
```

jupyter liver disease Last Checkpoint: 08/07/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [9]:
```
# Copy Images to test set

# seed random number generator
random.seed(237)
# define ratio of pictures used for testing
test_ratio = 0.2


for cases in [abnormal, normal]:
    total_cases = len(cases['images']) #number of total images
    num_to_select = int(test_ratio * total_cases) #number of images to copy to test set

    print(cases['class'], num_to_select)

    list_of_random_files = random.sample(cases['images'], num_to_select) #random files selected

    for files in list_of_random_files:
        shutil.copy2(files, 'test/' + cases['class'])
```

```
abnormal 10
normal 9
```

In [10]:
```
# Copy Images to train set
for cases in [abnormal, normal]:
    image_test_files = os.listdir('test/' + cases['class']) # list test files
    for images in cases['images']:
        if images.split('/')[-1] not in (image_test_files): #exclude test files from shutil.copy
            shutil.copy2(images, 'train/' + cases['class'])
```

In [11]:
```
total_train_abnormal = len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal'))
total_train_normal = len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/normal'))
total_test_abnormal= len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal'))
total_test_normal = len(os.listdir('C:/Users/B ARUNREDDY/Music/liver disease/dataset/normal'))

print("Train sets Images abnormal: {}".format(total_train_abnormal))
print("Train sets Images normal: {}".format(total_train_normal))
print("Test sets Images abnormal: {}".format(total_test_abnormal))
print("Test sets Images normal: {}".format(total_test_normal))
```

jupyter liver disease Last Checkpoint: 08/07/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

CNN Model

In [12]:
```
batch_size =15
epochs = 10
IMG_HEIGHT = 150
IMG_WIDTH = 150
```

In [13]:
```
train_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our training data
test_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our validation data
```

In [14]:
```
train_dir = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset')
test_dir = os.path.join('C:/Users/B ARUNREDDY/Music/liver disease/dataset')


total_train = total_train_abnormal + total_train_normal
total_test = total_test_abnormal + total_test_normal
```

In [15]:
```
train_data_gen = train_image_generator.flow_from_directory(batch_size=batch_size,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                           class_mode='binary')
```

```
Found 75 images belonging to 2 classes.
```

In [16]:
```
test_data_gen = test_image_generator.flow_from_directory(batch_size=batch_size,
                                                         directory=test_dir,
                                                         target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                         class_mode='binary')
```

```
Found 75 images belonging to 2 classes.
```

In [17]:
```
model = Sequential([
    Conv2D(32, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(2, 2),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(64, 3, padding='same', activation='relu'),
```

jupyter liver disease Last Checkpoint: 08/07/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

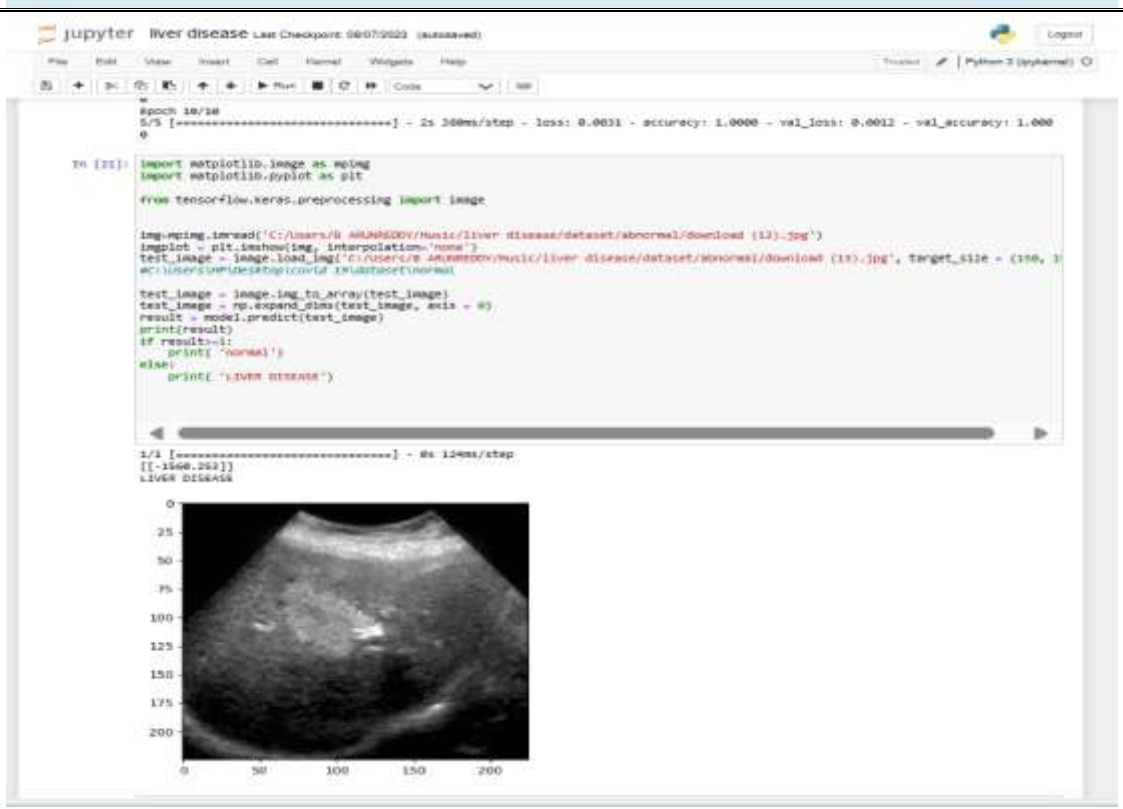
Trusted | Python 3 (ipykernel)

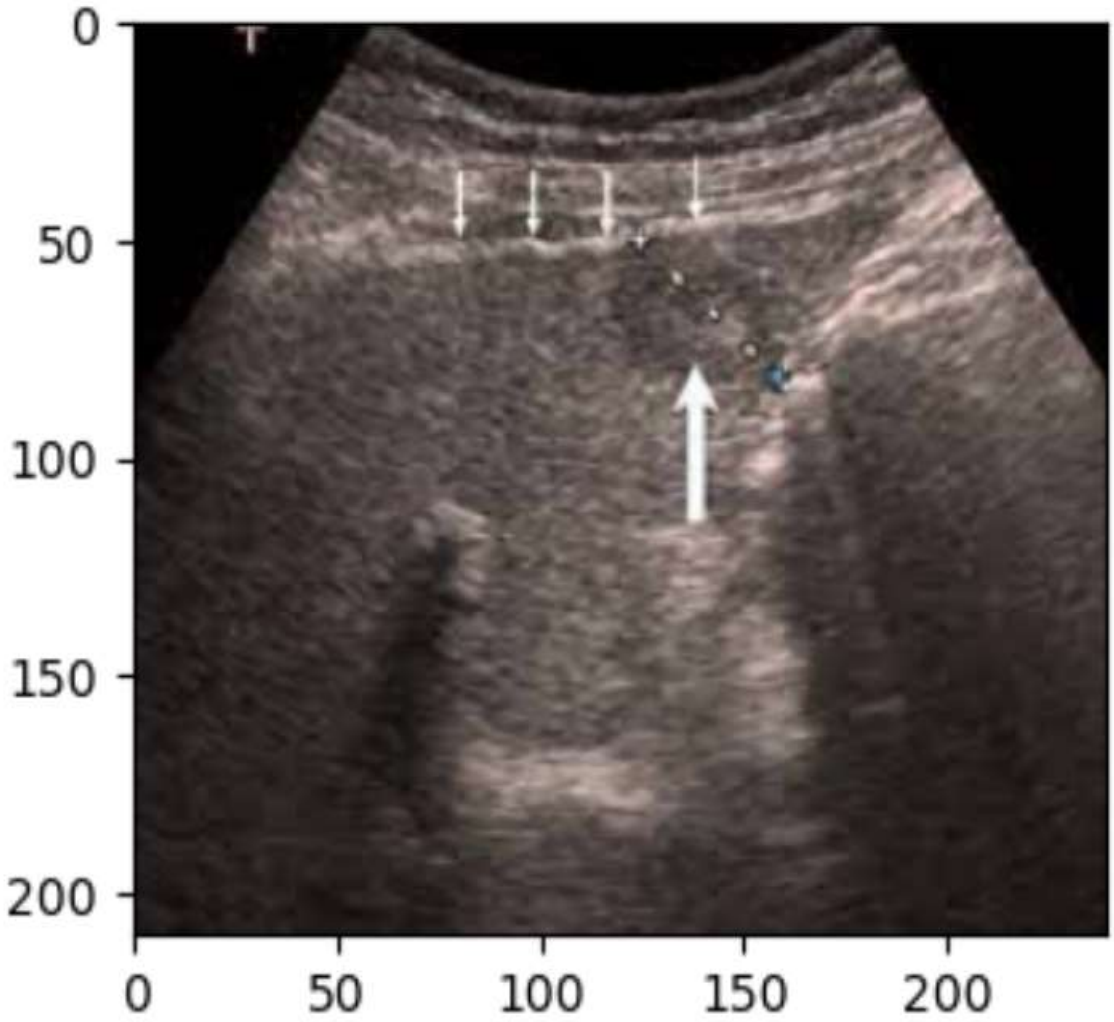
Run Code

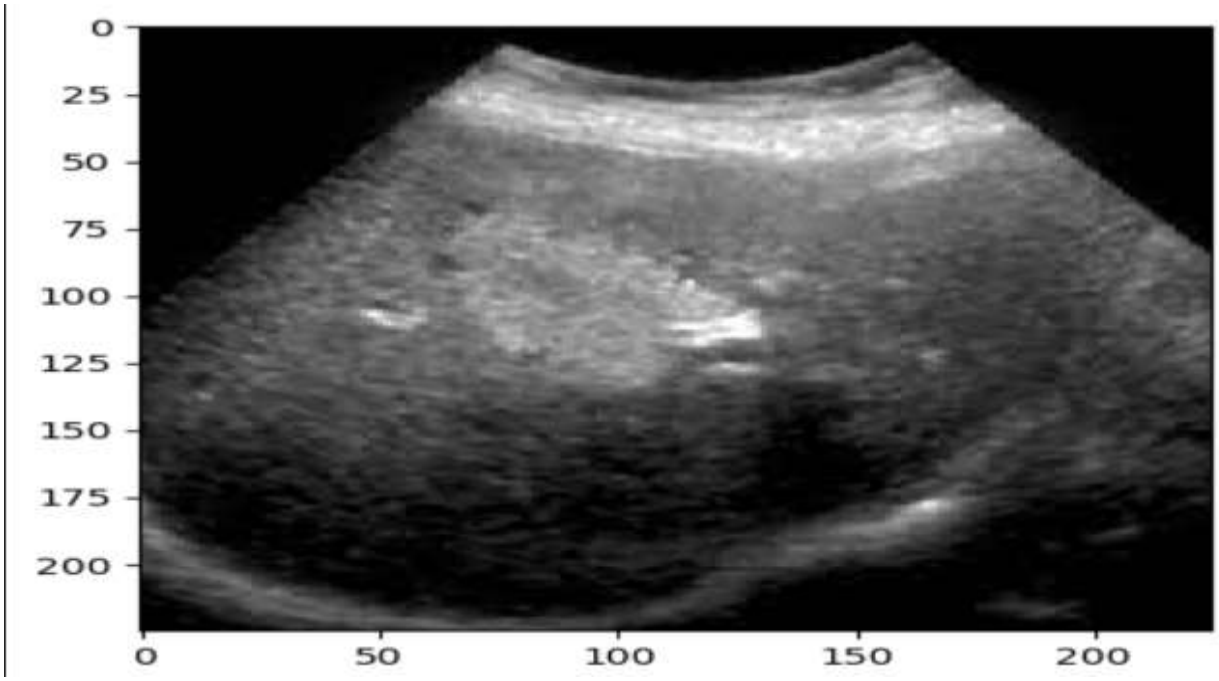
In [19]:
```
model.summary()
```

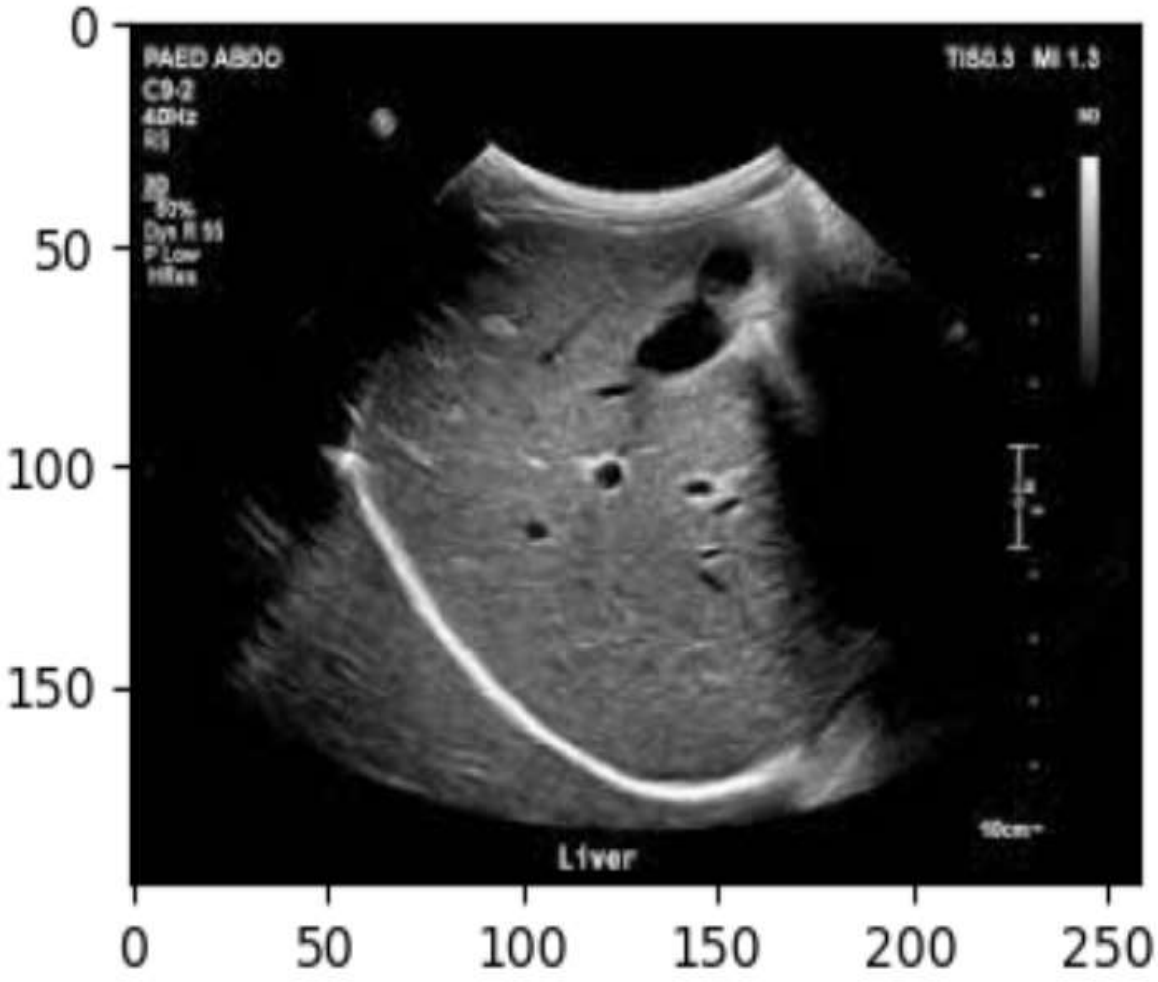
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896

 max_pooling2d (MaxPooling2  (None, 75, 75, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 37, 37, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 64)        36928

 max_pooling2d_2 (MaxPoolin  (None, 18, 18, 64)        0
 g2D)

 flatten (Flatten)           (None, 20736)             0

 dense (Dense)               (None, 512)               10617344

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 10674177 (40.72 MB)
Trainable params: 10674177 (40.72 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [20]:
```
history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=test_data_gen,
    validation_steps=total_test // batch_size
)
```

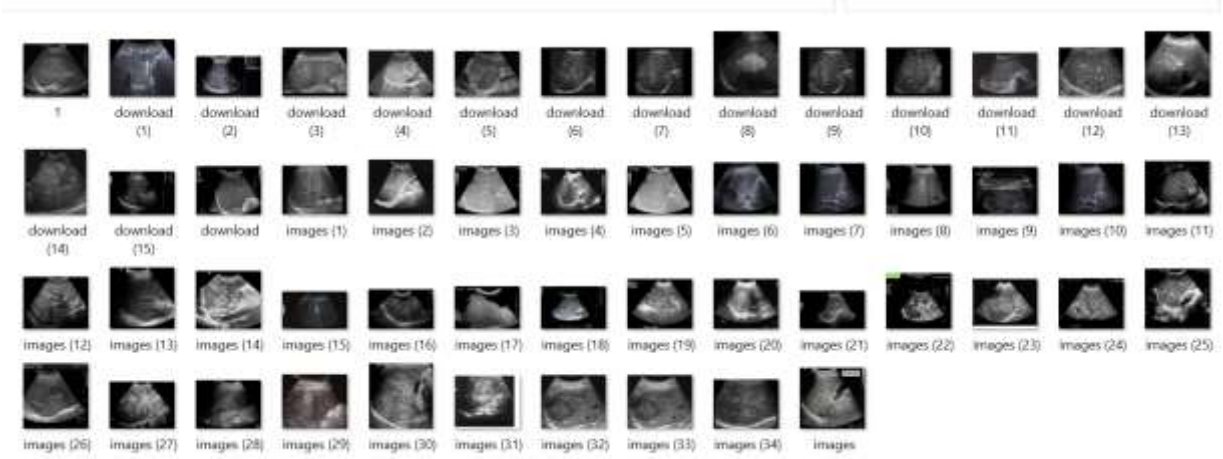

jupyter liver disease
Epoch 10/10
5/5 [==========] - 2s 388ms/step - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.0012 - val_accuracy: 1.000
0
In [21]:
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
img=mpimg.imread('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal/download (13).jpg')
imgplot = plt.imshow(img, interpolation='none')
test_image = image.load_img('C:/Users/B ARUNREDDY/Music/liver disease/dataset/abnormal/download (13).jpg', target_size = (150, 1
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = model.predict(test_image)
print(result)
if result==1:
print('normal')
else:
print('LIVER DISEASE')
1/1 [==========] - 0s 124ms/step
[[-1660.252]]
LIVER DISEASE
0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
50
100
150
200
0
50
100
150
200

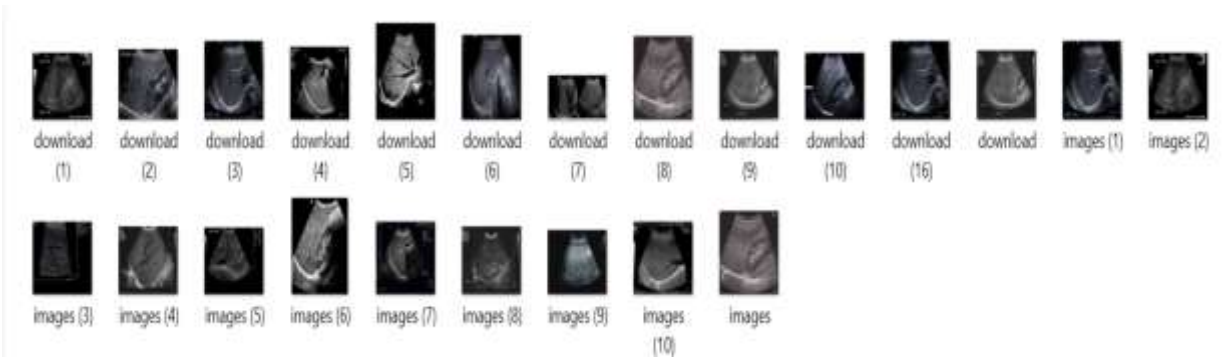

0
25
50
75
100
125
150
175
200
0
50
100
150
200


PAED ABDO
C9-2
Liver
0
50
100
150
0
50
100
150
200
250

Abnormal cases



Normal cases



FUTURE ENHANCEMENT

Future enhancements for liver disease detection using CNNs could focus on improving various aspects of the system to make it more accurate, efficient, and clinically relevant.

In the future, expanding the system's capabilities to perform multi-class segmentation can enhance its clinical utility. This would involve differentiating between various types of liver lesions, providing radiologists with a more detailed diagnostic picture. Additionally, extending the project to work with 3D CT scans could offer a comprehensive perspective for lesion identification and growth assessment. Real-time processing optimizations would make the system applicable during surgeries, while an interactive user interface would facilitate seamless integration into the radiologist's workflow. Leveraging advanced data augmentation techniques, transfer learning, and incorporating explainability and uncertainty estimation methods would further enhance the system's performance and usability. These ongoing refinements will position the project as a valuable tool in the medical field, aiding in accurate and efficient liver lesion diagnosis and treatment planning.

Finally, incorporating radiomics analysis to extract quantitative lesion features could provide deeper insights into lesion characteristics, adding another layer of information for radiologists to make more informed decisions.

CONCLUSION

In conclusion, the application of Convolutional Neural Networks (CNNs) for liver disease detection from medical images shows great promise in advancing the field of medical diagnostics. The use of CNNs in liver disease detection offers several advantages, including automated and accurate analysis, the ability to handle large and diverse datasets, and the potential for real-time diagnosis. This conclusion summarizes the key findings and contributions of using CNNs for liver disease detection.

This project represents a significant step forward in the field of medical imaging and computer-aided diagnosis. The development of a Cascade Convolutional Neural Network (CNN) with an Adam-optimized classifier for the segmentation of liver lesions in CT scans demonstrates the potential for leveraging deep learning techniques to enhance the precision and efficiency of diagnosis.

In summary, this project lays a solid foundation for the ongoing development and integration of AI-driven tools in the healthcare sector. By addressing the challenges of liver lesion segmentation, it contributes to the broader mission of improving patient care, early disease detection, and more informed clinical decisions. As this technology matures and incorporates future enhancements, it has the potential to significantly impact the field of radiology and, ultimately, patient outcomes.

REFERENCES

1.Denbow, D.M. Gastrointestinal Anatomy and Physiology. In Sturkie's Avian Physiology, 6th ed.; Academic Press: Cambridge, MA, USA, 2015; pp. 337–366.

2. WHO. Global Health Estimates 2015: Deaths by Cause, Age, Sex, by Country and by Region; WHO: Geneva, Switzerland, 2015.

3.Rahimian, J.; Wilson, T.; Oram, V.; Holzman, R.S. Pyogenic liver abscess: Recent trends in etiology and mortality. Clin. Infect. Dis.2004,39, 1654–1659. [CrossRef] [PubMed]

4.Akhondi, H.; Sabih, D. Liver Abscess. 2021. Available online: https://www.ncbi.nlm.nih.gov/books/NBK538230/ (accessed on15 October 2021).

5.Burt, A.; Ferrell, L.; Hubscher, S. MacSween's Pathology of the Liver E-Book; Elsevier Health Sciences: Amsterdam, The Netherlands, 20

6.Pere, G.; Graupera, I.; Lammert, F.; Angeli, ., Caballeria, L. Screening for liver fibrosis in the general population: A call for action.Lancet Gastroenterol. Hepatol. 2016,1, 236–264.

7.Anthony, P.P.; Ishak, K.G.; Nayak, N.C.; Poulsen, H.E.; Scheuer, P.J. The morphology of Cirrhosis. Recommendations on definition,nomenclature and classification by a working group sponsored by the World Health Organization. J. Clin. Pathol.1978,31,395–414. [CrossRef] [PubMed]

8.Goldfarb, G.; Nouel, O.; Poynard, T.; Rueff, B. Efficiency of respiratory assistance in cirrhotic patients with liver failure. IntensiveCare Med. 1983,9, 271–273. [CrossRef] [PubMed]

9.Garcia-Doval, I.; Hernandez, M.V.; Vanaclocha, F.; Sellas, A.; Montero, D. Should tumor necrosis factor antagonist safetyinformation be applied from patients with rheumatoid arthritis to psoriasis? Rates of serious adverse events in the prospectiverheumatoid arthritis BIOBADASER and psoriasis BIOBADADERM cohorts. Br. J. Dermatol.2017,176, 643–649. [CrossRef][PubMed]

10.Hossein, A.; Sabih, D.E. Liver Abscess. 2019. Available online: https://europepmc.org/article/NBK/nbk538230 (accessed on 15October 2021).

11. Scheuer, J.H. Liver Biopsy Interpretation E-Book; Elsevier Health Sciences: Amsterdam, The Netherlands, 2020.

12.Siriwardena, K.; Mason, J.M.; Mullamitha, S.; Hancock, H.C.; Jegatheeswaran, S. Management of colorectal cancer presentingwith synchronous liver metastases. Nat. Rev. Clin. Oncol. 2014,8, 446–459. [CrossRef] [PubMed]

13.Azer, S.A. Deep learning with convolutional neural networks for identification of liver masses and hepatocellular carcinoma: Asystematic review. World J. Gastrointest. Oncol. 2019,12, 1218–1230. [CrossRef]

14.Xu, J.; Jing, M.; Wang, S.; Yang, C.; Chen, X. A review of medical image detection for cancers in digestive system based on artificialintelligence. Expert Rev. Med. Devices 2019,10, 877–889. [CrossRef]

15.Faust, O.; Acharya, U.R.; Meiburger, K.M.; Molinari, F.; Koh, J.E. Comparative assessment of texture features for the identificationof cancer in ultrasound images: A review. Biocybern. Biomed. Eng. 2018,2, 275–296. [CrossRef]

16.Moghbel, M.; Mashohor, S.; Mahmud, R.; Saripan, M.I.B. Review of liver segmentation and computer assisted detection/diagnosismethods in computed tomography. Artif. Intell. Rev. 2018,4, 497–537. [CrossRef]

17.Nayantara, P.V.; Kamath, S.; Manjunath, K.; Rajagopal, K. Computer-aided diagnosis of liver lesions using CT images: Asystematic review. Comput. Biol. Med. 2020,3, 234–248. [CrossRef]

18.Kaur, R.; Juneja, M.; Mandal, A.K. A hybrid edge-based technique for segmentation of renal lesions in CT images. Multimed. ToolsAppl. 2018,6, 12917–12937. [CrossRef]

19.Jeffers, A.M.; Sieh, W.; Lipson, J.A.; Rothstein, J.H.; McGuire, V. Breast cancer risk and mammographic density assessed withsemiautomated and fully automated methods and BI-RADS. Radiology 2016,2, 348–355. [CrossRef] [PubMed]

20.Muthuselvan, S.; Rajapraksh, S.; Somasundaram, K.; Karthik, K. Classification of Liver Patient Dataset Using Machine LearningAlgorithms. Int. J. Eng. Technol. 2018,7, 323–326. [CrossRef]

21.Idris, K.; Bhoite, S. Applications of Machine Learning for Prediction of Liver Disease. Int. J. Comput. Appl. Technol. Res.2019,8,394–396. [CrossRef]22. Murty, S.V.; Kumar, R.K. Enhanced classifier accuracy in liver disease diagnosis using a novel multi-layer feed forward deepneural network. Int. J. Recent Technol. Eng. 2019,8, 1392–1400.