MONITORING, LOGGING IN SERVER AND STORAGE USING PROMETHEUS AND GRAFANA

Submitted by

BARANI DHARAN A

(921623405002)

In partial fulfilment for the award of the degree of

MASTER OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SBM COLLEGE OF ENGINEERING & TECHNOLOGY,

DINDIGUL-5

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING ANNA UNIVERSITY: CHENNAI 600 025

NOV-DEC 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"MONITORING, LOGGING IN SERVER AND STORAGE USING PROMETHEUS AND GRAFANA"**is the bonafide work of **"BARANI DHARAN A (921623405002)"**who carried out the project work under my supervision.

SIGNATURE

Mr.S.SASIKUMAR, M.E., (Ph.D).,

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering, SBM College of Engineering and Technology Dindigul-5.

SIGNATURE

Mr.RUBANCHAKRAVARTHY,

M.E.,SUPERVISOR/ASST.PROF,

Department of Computer Science and Engineering, SBM College of Engineering and Technology, Dindigul-5.

Submitted for the Anna University, Project Phase-I Viva Voice Examination held on _____at SBM College of Engineering and Technology, Dindigul.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it Possible, whose constant guidance and encouragement crown all efforts with success. We express our deepest gratitude to our **Chairman Dr.N.JAYARAJ, MJF, D.Lit., Ph.D., D.HONS**, for his invaluable guidance and blessings.

I extend my warm gratitude to our **Principal Dr.G.VIJAY CHAKARAVARTHY, B.E., M.E., Ph.D.,** who is always with me in all my activities that I have in my college premises.

I extend my warm gratitude to our **CEO & Managing Trustee Mr.J.DINESH @ JAYASIMHA., B.E., M.E., (Ph.D).,** who is always with me in all my activities that I have in my college premises.

I extend my gratitude to my **Head of the Department Mr.S.SASIKUMAR., M.E., (Ph.D)** for her grateful support to complete this project.

I extend my gratitude to my **Supervisor Mr.S.SASIKUMAR., M.E., (Ph.D)** in Computer Science and Engineering Department, for his valuable guidance, encouragement, constructive criticism and unreserved co-operation extended at each stage to complete this project successfully.

Last, I am extremely grateful to my parents and friends for their constant encouragement and moral support throughout my venture.

BARANI DHARAN A

ABSTRACT

Since the inception of civilization, human beings have been the target of the effects of substances on the liver. The liver is one of the most important organs, being responsible for creating nutrients, metabolizing them and maintaining a non-toxic body condition. Liver disease is one of the most dangerous human diseases and has a very serious impact on human life. It predicts liver disease accurately and efficiently by using deep learning algorithms (DNN). Liver diseases, encompassing various conditions such as tumours, cirrhosis, and hepatitis, pose significant challenges to global healthcare systems due to their prevalence and complexity. Early detection of liver tumours, in particular, is crucial for timely intervention and improved patient outcomes. In this project, we propose an innovative approach to liver tumour detection leveraging cutting-edge deep learning techniques, specifically the YOLO (You Only Look Once) object detection framework. Our system aims to analyze scanned liver images with high accuracy and efficiency, enabling the identification and localization of tumour regions within the liver. By harnessing the capabilities of YOLO, the system can rapidly process large volumes of medical images and provide real-time feedback to healthcare professionals. The proposed system offers a promising solution for enhancing diagnostic accuracy, facilitating prompt treatment decisions, and ultimately improving the management of liver diseases. We propose the integration of a cloud-based storage solution to facilitate seamless access to input and output images and to store the results of tumor detection. This enhancement aims to improve the accessibility, scalability, and reliability of the system by leveraging cloud technology. Through the integration of advanced technology with medical imaging, our system contributes to the advancement of liver disease diagnosis and patient care, offering hope for better health outcomes and quality of life for individuals affected by liver tumours.

TABLE OF CONTENTS

LIST OF FIGURE

LIST OF ABBREVATION

ACRONYMS	ABBREVIATIONS
YOLO	You Only Look Once
CNN	Convolutional Neural Networks
SVM	Support Vector Machine
GUI	Graphical User Interface
KNN	K Nearest Neighbor
ANN	Artificial Neural Networks
ILPD	Indian Liver Patient Dataset
SRS	Software Requirement Specification
CLAH	Contrast Limited adaptive histogram equalization
FPN	Feature Pyramids Network
HTML	Hyper Text Markup Language
CAD	Computer-aided Design
NLP	Natural Language Processing

CHAPTER 1

1. INTRODUCTION

The liver is one of the largest organs in our body. Untreated liver infections can lead to liver failure and malignancy. Liver disease is the leading cause of death worldwide . According to the World Health Organization around 58 million people having hepatitis C infection .Hepatitis C was an main cause of death for around 290,000 people in 2019 ,mainly due to the cancer of the liver and scarring. The liver is responsible for metabolic, strength-storing, and waste-filtering functioning in your body. The liver helps to eliminating hazardous compounds from the body, such as drugs and alcohol. It may also store many types of vital elements, such as vitamins, minerals, and glucose, and transport them into circulation as needed. Cirrhosis and hepatitis kill around 2 million people worldwide. ML is widely used in healthcare industry for analyse various types of disease. When someone is currently in bad health they must schedule an expensive and time-consuming doctor appointment. Also, it can be not easy for the user if they are far from health facilities because the condition cannot be recognized.

So, it can be better for the patient and make the process run more smoothly if the aforementioned operation can be carried out utilising an automated software that saves time and money . The aim of this study seeks to to conquer these challenges by developing a deep learning based technique for liver disease prediction in people. Jaundice, liver enlargement, swelling in the legs and ankles, itchy skin ,vomiting ,dark urine color, weight loss and weakness in the muscles are some of the typical liver symptoms. Inflammation is the initial stage of liver disease. The liver helps to eliminate toxic waste in our body. When the liver is unable to handle this toxic waste, the body respond by enlarging the liver. If the swelling is not treated properly, it can lead to scarring.

This stage is called fibrosis and at that stage your liver does not work properly. Cirrhosis is the third stage of liver disease .At Cirrhosis stage liver become much scarred and area around the liver itches. Liver failure is the final stage of liver disease .In this stage of liver failure, the liver loses its ability to function and chances of liver cancer is increasing at this stage

Problems with liver patients are not easily discovered in an early stage as it will be functioning normally even when it is partially damaged. An early diagnosis of liver problems will increase patient's survival rate. Liver failures are at high rate of risk among Indians. It is expected that by 2025 India may become the World Capital for Liver Diseases. The widespread occurrence of liver infection in India is contributed due to deskbound lifestyle, increased alcohol consumption and smoking. There are about 100 types of liver infections. Therefore, developing a deep that will enhance in the diagnosis of the disease will be of a great advantage in the medical field. These systems will help the physicians in making accurate decisions on patients and also with the help of Automatic classification tools for liver diseases (probably mobile enabled or web enabled), one can reduce the patient queue at the liver experts such as endocrinologists.

Classification techniques are much popular in medical diagnosis and predicting diseases. Michael J Sorich reported that SVM classifier produces best predictive performance for the chemical datasets. Lung-Cheng Huang reported that Naïve Bayesian classifier produces high performance than SVM and C 4.5 for the CDC Chronic fatigue syndrome dataset. Paul R Harper reported that there is not necessary a single best classification tool but instead the best performing algorithm will depend on the features of the dataset to be analyzed. The main objective of this research is to use classification algorithms to identify the liver patients from healthy individuals. In this study, FOUR classification algorithms Logistic Regression, Support Vector Deeps (SVM), K Nearest Neighbor (KNN) and artificial neural networks (ANN) have been considered for comparing their

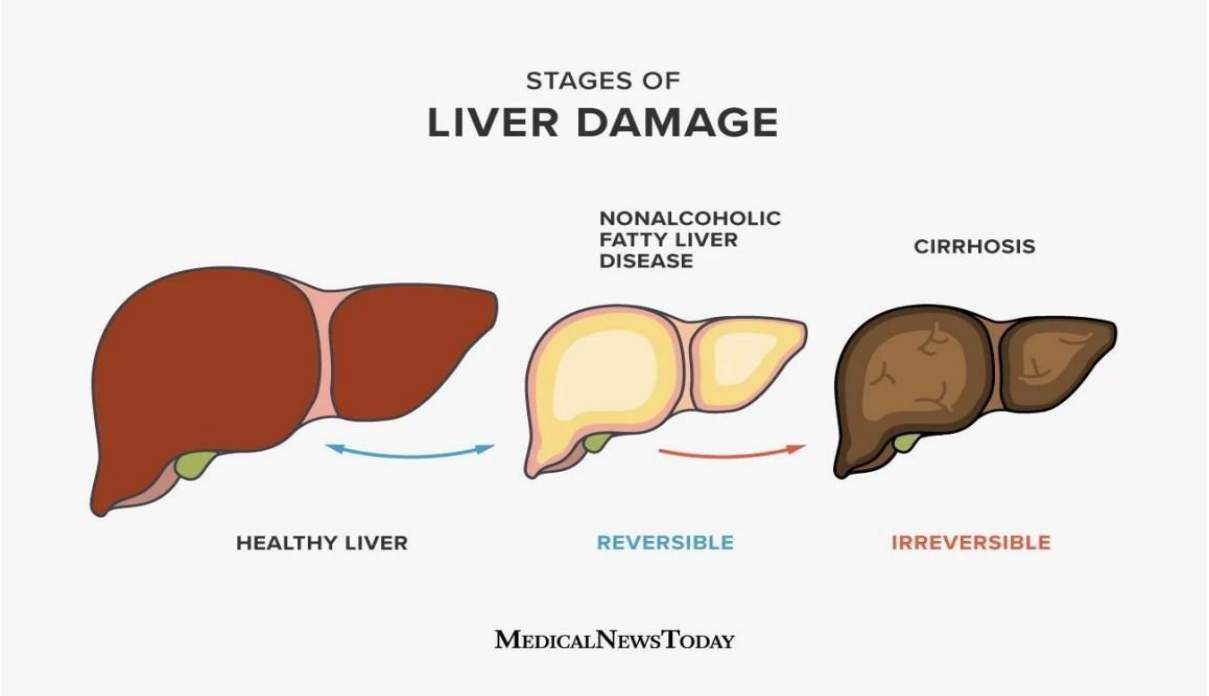
performance based on the liver patient data. Further, the model with the highest accuracy is implemented as a user friendly Graphical User Interface (GUI) using Tkinter package in python. The GUI can be readily utilized by doctors and medical practitioners as a screening tool for liver disease.

Millions of individuals worldwide suffer from liver problems, which are a global health concern. Successful treatment and the avoidance of problems with these illnesses depend on early discovery and correct diagnosis. Traditional diagnostic techniques are inaccurate, frequently intrusive, and time-consuming. Deep Learning (ML) approaches have been created as a potential remedy for the identification of liver disorders. Liver disease prediction is the process of determining the likelihood of anindividual developing liver disease based on various factors such as medical history, lifestyle habits, genetics, and other relevant information. The aim of liver disease prediction is to identify individuals who are at high risk of developing liver disease and to implement early interventions to prevent or slow down the progression of thedisease. There are several methods used for liver disease prediction including statistical models, deep learning algorithms and others. These models use various factors such as age, sex, alcohol consumption, body mass index, and others to make predictions about the likelihood of developing liver disease. To accurately anticipate and diagnose liver problems, ML systems can evaluate large datasets. In this work, the effectiveness of three widely used ML techniques are examined: logistic regression, support vector deeps, and decision trees. In order to diagnose liver problems, the study will evaluate the efficacy, sensitivity, specificity, and overall performance of these procedures.

Deep Learning has become one of the most evolving technologies in the current period. Deep learning can be simply explained as scientific study of algorithms and models in statistics where deeps can easily understand to perform and solve

specific tasks. This technique has become agile and it has been a requirement in most of the fields. 1

OUR MOTIVE Our motive for this project is to predict the liver diseases for a patient with the maximum amount of accuracy in our prediction. For this dataset was collected from 2 Indian patient liver disease dataset from Kaggle database of Indian Liver patient records and used that dataset in our three modules to predict the liver disease using various deep learning techniques



Efficient Technique Convolutional Neural Network (CNN) algorithm is considered as the best algorithm compared to other algorithms since it gives better accuracy after testing with few other algorithms.

Increased Accuracy There are many algorithms in the past for detecting the disease but the algorithms that are used in our project will increase the efficiency in predicting the liver disease. The deep learning algorithm called Convolutional Neural Network(CNN) gives us higher accuracy compared to others.

DATA MINING: Data extraction is the way to find designs in expansive informant indexes including AI crossing point strategies, measurements and

database frames. Information Mining is an interdisciplinary field of software engineering and measuring that aims to separate data from information collection (with keen strategies) and transform data into an understandable structure to be used further.

DATA PRE-PROCESSING: Data pre-processing is an important step to solve each problem of Deep Learning. In order for a deep learning algorithm to be trained, most of the data sets used with deep teaching problems must be processed. The techniques used most commonly for pre- processing are very few such as imputation of lack of value, categorical coding, scaling, etc. These techniques are easy to understand. Every dataset has its own unique challenges and is different. Total number of data points is 583, with 416 liver patient records and 167 non-liver patient records.

Dataset: The Indian Liver Patient Dataset contained 10 distinct qualities of 583 patients. The patients were portrayed as either 1 or 2 based on liver sickness. The nitty gritty portrayal of the dataset is appeared Table. The table give insights regarding the trait and characteristic sort. As plainly unmistakable from the table, every one of the highlights with the exception of sex are genuine esteemed numbers. The component Sex is changed over to numeric esteem (0 and 1) in the information pre-preparing step.

Data Collection: Collection of data is crucial for these kinds of projects. We have collected a dataset named as Indian Patient Liver Dataset from Kaggle which consists of 10 different attributes of 583 patients.

Data Cleaning: There are different columns which are called attributes. Some columns have null values and some values are fluctuated so the next step is to clean those values from the datasheet and then take that dataset for classification.

AIM OF THE PROJECT: To identify and predict the patient has diseased liver or not at early stages.

SCOPE OF THE PROJECT: The diagnosis of liver disorders is a crucial task in healthcare, as liver diseases are becoming increasingly prevalent worldwide. Identifying the early signs of liver disease can help in preventing the progression of the disease and reducing the risk of complications. Deep learning techniques can be utilized to develop predictive models for diagnosing liver disorders. These models can analyse various parameters, such as blood tests, liver function tests, and other patient data to identify potential liver disorders. In summary, the development of a deep learning-based diagnostic model for liver disorders has enormous potential to improve the accuracy and efficiency of diagnosis. With proper integration of different factors, such models can provide personalized diagnosis, improve patient outcomes and reduce healthcare costs.

To develop and implement Deep Learning approaches that learns the general patterns of the Liver Diseases, and a liver disease model is trained. To bring better efficiency to the algorithms which will be used in finding the accuracy of the liver diseases.

CHAPTER 2

2. LITERATURE SURVEY

Rakshit D B et al.in 2021 has represented a Liver disease detection system specially designed for working with the efficient prediction through KNN. This model gives on an average accuracy with BLEU index which is comparably gives better result with large dataset. KNN is a deep learning technique that is frequently employed for regression and classification tasks. This could be a helpful tool for detecting kidney stones early, which could result in better outcomes and more suitable treatment options.

B. Poonguzharselvi et al. in 2021 has proposed that identifies the significant features and then predicts whether or not a person may suffer or is suffering from Liver Disease have suggested a technique for finding key characteristics that can beused to determine whether a person has liver disease or is at risk for developing it. Knowing how likely someone is to acquire liver disease could be highly helpful for early detection and treatment because it is a serious and potentially fatal ailment. Poonguzharselvi et al. may have searched through a range of data sources, including patient medical records, lab findings, and imaging data, to find patterns and correlations that are suggestive of liver illness in order to uncover relevant features.

Shruti Suresh et al. in 2022 has have presented a paper describing the difficultyin detecting liver illnesses using computer vision. They specifically suggest that using the connection between picture attributes, keywords for new photos can be generated from a training set of images of liver illnesses. This method is predicatedon the notion that liver disease photos will have some characteristics in common, and that these characteristics can be utilized to identify the condition. A deep learning system can learn to recognize the common aspects of each disease

and provide keywords that correlate to those qualities by being trained on a database 6 of photos with known diseases.

Software based Prediction of Liver Disease with Feature Selection and Classification Techniques Jagdeep Singha, Sachin Baggab, Ranjodh Kaur: A web-based software application has been developed on basis of the software engineering life cycle model. There are four main phases: Planning and Analysis, Design and Build, Implementation, Validation, and delivery. Each phase contains several activities, and each phase is interconnected with other phases.

Preprocessing and Feature selection to normalize the missing values, preprocessing techniques have been introduced. The missing values were replaced by null values along with their instances. Feature selection was followed to classify the appropriate attribute for classification. Using both filter and wrapper approaches, feature selection was carried out. The attributes with more than 70% correlation were initially excluded by correlation analysis from the dataset. The algorithm was implemented to estimate the value of different features in a dataset on the basis of random forests.

Planning and Analysis Phase Planning phase includes the creation of ideas to support healthcare and technical team through the prediction of liver diseases. The main objective of planning phase is to plan the step involved in the development of prediction system using software engineering life cycle. In addition, challenging thing is to remove the gap between the software development members and health care specialists. In the analysis phase, the concern is to gather prediction system requirements and environmental considerations. The requirements involve the people from a different background area such as informaticists, physicians, patients etc.

Design and Build Phase In design phase, the architecture model of liver diseases prediction software is established. The architecture defines user interface,

segment, action and behaviour of the ILDP Software. The design document defines the technical plan to implement as per the requirements to build the system. The details of packages, programming language, platform, environment, and other technical/nontechnical details are established.

Implementation Phase In implementation phase, the development of ILDPS done as identified in the design phase. The main challenge in implementation phase is to implement the prediction system as per requirement, planning, and design. In the implementation phase, ILDPS is dealing with problems related to the performance, quality and debugging.

REVIEW OF LIVER DISEASE PREDICTION USING DEEP LEARNING ALGORITHM: Vijay Panwar, Naved Choudhary, Sonam Mittal, Gaurav Sahu: Data Collection: For this study, the Indian Liver Patient Dataset (ILPD) was selected from the UCI Deep Learning repository. It is a sample of the whole Indian population taken from the area of Andhra Pradesh. There were 583 instances based on ten different biological parameters in the dataset. Based on these criteria, the class value was stated as either yes (416 cases) or no (167 cases), reflecting the liver.

Randomization and splitting of dataset To build classification models, the features selected in the preceding phase were accepted. The dataset was initially randomized to produce an arbitrary sample permutation. Splitting of the dataset into training (70 percent of the dataset) and test (30 percent) sets was followed. The training set consisted of 389 cases and the evaluation set consisted of the remaining cases.

Classification algorithms is one of the greatest significant and applicable data mining techniques used to apply in disease prediction. Classification algorithm is the most common in several automatic medical health diagnoses.

1. An Effective approach for early liver disease prediction

The liver is one of the most essential organs in the human body, responsible for critical functions such as detoxification, metabolism, and nutrient storage. Despite its resilience and ability to function even when partially damaged, early detection of liver diseases remains a significant challenge. Many liver disorders remain asymptomatic in the early stages, making timely diagnosis crucial for improving patient outcomes and survival rates. This research explores the application of Artificial Intelligence (AI) techniques to enhance early detection of liver disease, focusing on the predictive capabilities of several machine learning models, including Bagged Tree, Support Vector Machine (SVM), KNearest Neighbor (KNN), and Fine Tree classifiers.

A comparative analysis of these models is conducted to identify the most effective approach for accurate prediction. Performance metrics such as the confusion matrix, True Positive Rate (TPR), False Positive Rate (FPR), Receiver Operating Characteristic (ROC) curve, and overall accuracy are used to evaluate the models. Among the models studied, the Bagged Tree classifier demonstrates superior performance, achieving an accuracy of 81.30%, outperforming other algorithms in terms of reliability and precision.

Additionally, the study performs sensitivity analysis on the dataset to assess the impact of individual attributes on model performance. Results indicate that the Alanine Aminotransferase (ALT or SGPT) attribute plays a pivotal role in predicting liver disease, suggesting that this biomarker is critical in early-stage diagnosis.

The findings of this research highlight the potential of AI-driven approaches to serve as an assistive framework for healthcare professionals, providing a robust tool for early liver disease detection and facilitating timely medical intervention. This work underscores the importance of integrating machine learning techniques in clinical practice to enhance diagnostic accuracy and improve patient care.

2. Prediction of chronic liver disease patients using integrated projection based statistical feature extraction with machine learning algorithms

The liver plays a vital role in maintaining human health, performing over 500 critical functions, including detoxification, protein synthesis, metabolism regulation, and bile production. However, any malfunction of the liver can be lifethreatening, emphasizing the urgent need for early diagnosis and intervention

in liver diseases. Early detection significantly enhances the chances of successful treatment and survival. In this context, Machine Learning (ML) emerges as a transformative tool capable of supporting healthcare professionals by improving diagnostic accuracy and efficiency in identifying liver disease, particularly in its early stages.

A standard ML diagnostic system typically involves three key phases: data preprocessing, feature extraction, and classification. The feature extraction stage is critical as it aims to reduce data redundancy while preserving the most relevant information for accurate classification. However, traditional projection-based feature extraction methods often fall short of delivering optimal results due to the limitations in their statistical approaches, which may not align with the complex characteristics of liver disease data.

This study utilizes the Indian Liver Patient Dataset (ILPD) sourced from the University of California, Irvine (UCI) repository to classify chronic liver disease. The dataset comprises 583 patient records, of which 416 are diagnosed with liver disease, while 167 are not. To enhance classification accuracy, this research proposes an integrated feature extraction approach that combines various projection methods. The pipeline begins by handling missing values and outliers through a pre-treatment imputation process. Once the data is cleaned, the integrated feature extraction technique is applied to identify the most critical features for effective classification.

A simulation study is conducted to validate the robustness of the proposed methodology. Several ML algorithms are employed, including Logistic Regression (LR), Random Forest (RF), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and the Ensemble Voting Classifier. The proposed system achieves outstanding performance metrics, with an accuracy of 88.10%, precision of 85.33%, recall of 92.30%, F1 score of 88.68%, and an AUC (Area Under the Curve) score of 88.20%.

Notably, the proposed approach outperforms existing studies, showing an improvement of 0.10% to 18.5% in various performance metrics. These results highlight the efficacy of the integrated feature extraction method and the ensemble approach in accurately predicting liver disease. The findings underscore the potential of this ML-based system to serve as a complementary diagnostic tool, providing invaluable support to physicians in diagnosing liver disease, thereby facilitating timely and effective treatment decisions.

3. An efficient classification of cirrhosis liver disease using hybrid convolutional neural network-capsule network

Liver cirrhosis represents the diffuse, irreversible, and advanced stage of liver disease, characterized by extensive scarring and functional impairment. Early and accurate diagnosis is crucial for effective management and treatment, yet existing morphological imaging methods, such as ultrasound, CT, and MRI, often suffer from bias and limited detection accuracy. These limitations highlight the need for automated and precise diagnostic systems. This study proposes an innovative automated liver cirrhosis classification system based on an optimized hybrid deep learning model to enhance diagnostic accuracy using Magnetic Resonance Imaging (MRI).

The proposed system begins by preprocessing the input MRI images using an Extended Guided Filter (EGF) to eliminate noise while preserving essential structural details. Once the noise is removed, the tumor or affected regions are segmented using a binomial thresholding technique, ensuring precise delineation of the area of interest. Following segmentation, the Feature Extraction (FE) phase is conducted using two robust texture analysis methods: Grey Level Cooccurrence Matrix (GLCM) and Grey Level Run-Length Matrix (GRLM). These methods capture essential textural features such as spatial relationships between pixels and run-length patterns, which are critical for identifying liver cirrhosis.

The classification phase integrates a hybrid deep learning model combining Convolutional Neural Networks (CNNs) and Capsule Networks (CNs), denoted as HCNN-CN. CNNs are well-known for their efficacy in feature extraction and pattern recognition, while Capsule Networks enhance the model’s ability to capture spatial hierarchies and relationships between features, improving classification accuracy. To further refine the neural network's performance, an optimization algorithm, Adaptive Emperor Penguin Optimization (AEPO), is employed. AEPO fine-tunes the parameters of the neural network, ensuring optimal performance and convergence.

The effectiveness of the proposed HCNN-CN-AEPO model is validated through extensive experiments using real-time datasets. The results demonstrate exceptional performance, with an accuracy of 99.3% and a sensitivity of 98.6%, significantly outperforming existing methods. The high sensitivity ensures

minimal false negatives, making it a reliable tool for early detection and diagnosis of liver cirrhosis.

In conclusion, the experimental results confirm that the HCNN-CN-AEPO model offers a highly accurate and automated solution for diagnosing cirrhosis from MRI images. Its superior performance compared to traditional methods underscores its potential as a powerful diagnostic aid in clinical settings, facilitating timely and accurate detection of liver tumors and advancing the field of medical imaging for liver disease diagnosis.

4. Artificial intelligence in liver diseases: Improving diagnostics, prognostics and response prediction

Hepatology, the branch of medicine focused on liver-related diseases, encompasses the diagnosis and treatment of a wide range of metabolic, infectious, autoimmune, and neoplastic conditions. Clinical routines in hepatology involve synthesizing both qualitative and quantitative information from diverse data sources, including patient history, laboratory results, imaging studies, and histopathological analyses. Clinicians rely on this integrated information to diagnose liver diseases, predict their progression, and recommend appropriate treatment strategies. However, the complexity and volume of data involved often pose challenges for timely and accurate decision-making.

Over the past five years, rapid advancements in Artificial Intelligence (AI), particularly in deep learning, have transformed the ability to extract clinically meaningful information from complex datasets. AI systems, especially those leveraging deep learning, have demonstrated remarkable success in analyzing large, unstructured datasets, such as histopathology and radiology images. These data sources are rich in diagnostic, prognostic, and predictive information, but their complexity often limits the ability of human experts to fully harness their potential. AI-driven systems can identify subtle patterns and correlations within these images, enhancing diagnostic precision and enabling personalized prognostication.

In the context of hepatology, AI holds the promise of revolutionizing clinical practice by serving as decision support tools that assist clinicians in diagnosing liver diseases, predicting disease trajectories, and tailoring treatment plans.

Specifically, AI can enhance the interpretation of histopathological images by identifying nuanced tissue changes indicative of liver disease severity and progression. Similarly, AI can analyze radiological images with greater accuracy and consistency, detecting early signs of liver fibrosis, cirrhosis, or tumors that may be missed by human observers.

Despite these promising advancements, significant challenges remain before AI can be seamlessly integrated into clinical hepatology. Large-scale clinical validation is necessary to ensure the reliability, generalizability, and safety of these AI systems across diverse patient populations and clinical settings. Additionally, regulatory approval is essential to establish the clinical utility and compliance of AI tools with healthcare standards and guidelines.

This review summarizes the current state of AI applications in hepatology, with a particular focus on the use of AI in analyzing histopathology and radiology data. It highlights the key achievements, emerging trends, and potential benefits of AI in liver disease diagnosis and management. Furthermore, a roadmap is presented for the development of novel AI-driven biomarkers in hepatology, emphasizing the need for robust validation studies, interdisciplinary collaboration, and adherence to regulatory frameworks.

Critical obstacles are also outlined, including the need for large, high-quality datasets, the challenge of ensuring data privacy and security, and the importance of fostering clinician-AI collaboration to enhance trust and adoption. By addressing these challenges, the integration of AI into hepatology has the potential to significantly improve patient outcomes, optimize clinical workflows, and advance the field of liver disease research and treatment.

5. Liver Disease Prediction and Classification using Machine Learning Techniques

Liver diseases are emerging as one of the most lethal health disorders across numerous countries, contributing significantly to global morbidity and mortality rates. The rising prevalence of liver disorders is attributed to various factors, including excessive alcohol consumption, inhalation of harmful gases, ingestion of contaminated food, and the use of hepatotoxic drugs. This alarming trend underscores the urgent need for early diagnosis and efficient management of liver diseases. To alleviate the burden on healthcare professionals, particularly in

diagnosing chronic liver disorders, the development of automated classification models using patient datasets has become a critical focus of research.

In this study, we explore the application of machine learning (ML) algorithms to predict liver disorders by analyzing liver patient datasets. Chronic liver disorder is defined as a condition that persists for at least six months, causing proressive liver damage over time. The objective of this work is to build robust classification models that can process and analyze patient data to identify individuals at risk of developing liver disease. By leveraging the percentage of patients diagnosed with liver disorders as both positive and negative information, the ML models are trained to differentiate between healthy and diseased states.

The proposed methodology involves the use of multiple machine learning classifiers to process liver disease data. The classifiers are evaluated through the construction of confusion matrices, which provide a comprehensive assessment of model performance by highlighting true positives, false positives, true negatives, and false negatives. This analysis enables the identification of the most effective classification schemes capable of improving diagnostic accuracy when a suitable training dataset is available.

The classification process begins with the pre-processing of patient data to ensure data quality and consistency. Several ML algorithms, including decision trees, support vector machines (SVM), random forests, and logistic regression, are employed to classify liver health status. The classifiers are optimized to distinguish between good (healthy) and bad (diseased) liver function based on input features such as biochemical markers, demographic information, and clinical history.

The results of the proposed classification models demonstrate high accuracy in predicting liver disease outcomes. By automating the diagnosis process, these models significantly reduce the workload on healthcare professionals, allowing them to focus on critical cases and improve patient management. The findings suggest that the integration of ML-based classification models into clinical practice could serve as a valuable decision support tool, enhancing early detection, facilitating timely intervention, and ultimately improving patient outcomes in the management of liver diseases.

6. A randomized controlled trial for response of microbiome network to exercise and diet intervention in patients with nonalcoholic fatty liver disease

Non alcoholic fatty liver disease (NAFLD) and prediabetes are closely linked metabolic disorders, with lifestyle modifications such as exercise and dietary changes being the cornerstone treatments. However, the mechanisms by which these interventions impact the gut microbiota in affected patients remain poorly understood. The gut microbiota plays a crucial role in host metabolism, immune function, and overall health, suggesting that alterations in microbial composition may influence the effectiveness of NAFLD and prediabetes treatments. Understanding these interactions is vital for optimizing therapeutic strategies.

In a previously reported 8.6-month randomized, controlled trial, we investigated the effects of aerobic exercise, dietary intervention, and their combination on primary outcomes such as liver fat content and glucose metabolism in patients with NAFLD and prediabetes. This study employed a four-arm design involving 115 participants: Aerobic Exercise (n = 29), Diet (n = 28), Aerobic Exercise + Diet (n = 29), and No Intervention (n = 29). The trial was single-blinded for researchers to minimize bias. Here, we present findings related to the third primary outcome—gut microbiota composition—in 85 participants who completed the intervention (22 in Aerobic Exercise, 22 in Diet, 23 in Aerobic Exercise + Diet, and 18 in No Intervention).

Our results indicate that the combined Aerobic Exercise + Diet intervention produced significant diversification and stabilization of keystone microbial taxa, suggesting a synergistic effect on gut microbiota. In contrast, aerobic exercise or diet alone led to increased network connectivity and enhanced robustness among microbial taxa, emphasizing their individual contributions to gut ecosystem dynamics. Importantly, no adverse effects were observed across any of the intervention groups, underscoring the safety of these lifestyle modifications.

Exploratory ad-hoc analyses revealed variability in individual responses to the interventions. By examining differentially altered gut microbial amplicon sequence variants, we classified participants as responders or low/nonresponders. Notably, a personalized gut microbial network profile at baseline emerged as a predictor of individual responses to exercise, specifically in terms of liver fat reduction. This finding suggests that the baseline gut microbiome composition

may play a pivotal role in determining the efficacy of lifestyle interventions in NAFLD patients.

Our findings highlight the potential for developing personalized intervention strategies for NAFLD treatment, leveraging host-gut microbiome interactions to tailor therapies. These results pave the way for precision medicine approaches in metabolic disease management. However, further studies with larger sample sizes are necessary to validate these preliminary discoveries and establish robust, clinically applicable predictive models for personalized treatment in NAFLD and prediabetes.

7. Performance Analysis of Liver Disease Prediction Using Machine Learning Algorithms

Data mining has emerged as a cornerstone in the field of automated disease diagnosis and prediction, particularly in the analysis of complex medical datasets. It involves employing sophisticated algorithms and techniques to uncover hidden patterns, trends, and relationships within data, thereby aiding in early disease detection and clinical decision-making. In recent years, liver disorders have become a growing global health concern, with the incidence of liver disease increasing significantly across many countries. Liver diseases are now recognized as some of the most fatal conditions, necessitating the development of accurate and efficient predictive models for timely diagnosis.

This thesis focuses on the use of data mining techniques to build classification models for predicting liver disease by analyzing liver patient datasets. Specifically, the study aims to enhance the prediction accuracy of liver disease diagnosis among Indian patients through a systematic, multi-phase approach. The dataset utilized in this research is sourced from the University of California, Irvine (UCI) Machine Learning Repository, a widely recognized and trusted source for medical datasets.

In the first phase, the original liver patient dataset is preprocessed using the MinMax normalization algorithm. This step ensures that all attributes in the dataset are scaled within a specific range, typically between 0 and 1. Normalization is crucial as it eliminates biases due to varying data scales, improves the efficiency of classification algorithms, and enhances the convergence rate of the model training process.

The second phase involves feature selection using the Particle Swarm Optimization (PSO) algorithm. PSO is a population-based optimization technique inspired by the social behavior of birds flocking or fish schooling. It is used to identify a subset of significant features from the normalized dataset that are most relevant to liver disease prediction. By selecting only the most impactful attributes, this step reduces data dimensionality, minimizes computational complexity, and enhances the overall performance of the classification models.

In the third phase, various classification algorithms are applied to the dataset with the selected features. Among these algorithms, the J48 decision tree algorithm—a Java implementation of the C4.5 algorithm—is prominently used. J48 constructs decision trees by recursively partitioning the dataset based on attribute values, resulting in a tree-like structure that can be used for both classification and prediction.

The final phase involves evaluating the performance of the classification models using key metrics such as Root Mean Square Error (RMSE) and Root Mean Error (RME) values. The accuracy of each model is calculated to determine its effectiveness in predicting liver disease. The results demonstrate that the J48 algorithm, when combined with PSO-based feature selection, outperforms other classification algorithms, achieving an impressive accuracy of 95.04%.

8. Efficient Diagnosis of Liver Disease using Deep Learning Technique

The diagnosis a patient receives plays a critical role in ensuring patient safety, guiding clinical investigations, and informing healthcare policymaking. Accurate and timely diagnosis can significantly impact patient outcomes, reduce medical errors, and optimize treatment strategies. Traditionally, medical practitioners have relied on a variety of pathological and diagnostic techniques to evaluate patients' clinical information. However, recent advancements in Artificial Intelligence (AI) and Deep Learning (DL) have transformed the field of medical diagnosis, offering enhanced accuracy, efficiency, and reliability.

AI and DL, when integrated with clinical data, provide powerful tools to improve diagnostic precision. The widespread use of computers and internet connectivity has enabled the acquisition, analysis, and visualization of complex clinical datasets. Moreover, AI techniques can address challenges such as missing values in clinical research, which often compromise data integrity. Deep Learning algorithms, in particular, offer problem-specific solutions by automating complex pattern recognition, making them ideal for disease prediction and classification.

In this study, we focus on leveraging Deep Learning to distinguish liver patients from healthy individuals. Liver disease poses a significant global health burden, and early, accurate diagnosis is critical for effective treatment and management. To achieve this, we implemented a Deep Learning model based on the Bidirectional Long Short-Term Memory (BiLSTM) architecture. BiLSTM is a variant of the LSTM neural network that is capable of capturing long-term dependencies in sequential data by processing input sequences in both forward and backward directions. This bidirectional approach allows the model to retain contextual information from both past and future data points, enhancing its predictive capabilities.

The research involved multiple stages, beginning with data preprocessing to handle missing values and ensure the dataset's quality. The BiLSTM model was then trained on the preprocessed dataset, learning intricate patterns and relationships between various clinical features. The model's bidirectional nature allowed it to effectively capture temporal dependencies, making it well-suited for analyzing sequential clinical data related to liver function.

The BiLSTM model demonstrated remarkable predictive accuracy, achieving an overall efficiency of 93.00% in distinguishing liver patients from healthy individuals. This high level of accuracy indicates that the model effectively learned the underlying patterns associated with liver disease. Additionally, the implementation of this hybrid BiLSTM-based approach improved the overall predictive accuracy compared to traditional models, showcasing its potential as a powerful diagnostic tool.

The findings of this research highlight the potential of Deep Learning models, particularly BiLSTM, in enhancing the accuracy of disease diagnosis, specifically for liver disease. The ability to capture long-term dependencies in both temporal directions makes BiLSTM an effective tool for analyzing complex clinical datasets. The implementation of this hybrid model represents a significant

advancement in predictive healthcare, offering clinicians a robust, automated method for early and accurate detection of liver disease. Future work could focus on further refining the model, incorporating larger datasets, and validating the approach in real-world clinical settings to ensure broader applicability and reliability. This research underscores the transformative role of AI and DL in modern medical diagnostics, paving the way for more personalized and precise healthcare solutions.

9. When liver disease diagnosis encounters deep learning: Analysis, challenges, and prospects

The liver, the second-largest organ in the human body, plays a critical role in digesting food, metabolizing nutrients, and removing toxins. Despite its resilience, it remains vulnerable to damage from factors such as viral infections, excessive alcohol consumption, obesity, and exposure to harmful substances. If not addressed, liver disease can progress to serious conditions like cirrhosis or liver cancer. Traditionally, diagnosing liver disease has relied heavily on the expertise and judgment of healthcare professionals, making the process subjective, time-consuming, and prone to variability. However, with advancements in artificial intelligence (AI), particularly in deep learning (DL), there is growing interest in utilizing these technologies to improve diagnostic accuracy, objectivity, and efficiency in liver research. This paper provides a comprehensive review of deep learning applications in liver research, analyzing 139 studies from the last five years. By exploring how DL methodologies address challenges in diagnosing and treating liver diseases, the paper bridges the gap between data modalities, liver-specific topics, and clinical applications. To illustrate these connections, the study employs Sankey diagrams to visually map the relationships between data sources such as imaging and histopathology, liver disease topics, and the DL methods applied, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs). The review reveals that CNNs dominate tasks involving liver imaging due to their strength in visual data interpretation, while emerging models like transformers and GANs show promise in generating synthetic data to enhance small datasets. Despite notable successes, such as automating the detection and classification of liver lesions and predicting disease progression, challenges persist, including limited access to large, high-quality datasets, the need for standardized data acquisition protocols, and ethical concerns surrounding patient privacy and model interpretability. Nonetheless, the potential for DL to

revolutionize liver research is immense. As advancements continue, the development of explainable AI (XAI) is expected to address transparency issues, enabling clinicians to better understand and trust DL-driven insights. Collaborative efforts to create diverse datasets will further accelerate the integration of DL into clinical practice. This review underscores the transformative potential of deep learning in hepatology, highlighting its ability to enhance diagnostic accuracy, streamline clinical workflows, and support datadriven decision-making, ultimately paving the way for more personalized and effective liver disease management.

10. Detection of Disease in Liver Image Using Deep Learning Technique

Liver disease is among the leading causes of death globally, with liver cancer contributing significantly to this alarming statistic. Diagnosing liver cancer, particularly through manual identification of cancerous tissue, is a complex, time-intensive, and resource-demanding task that requires expert knowledge and precise evaluation. Early and accurate detection is essential for effective treatment planning, prognosis, and monitoring of clinical responses. One effective approach to addressing this challenge is the segmentation of liver lesions in computed tomography (CT) scans, which provides critical insights into the severity of the disease and aids in developing personalized treatment plans. This study proposes the application of a Fully Convolutional Neural Network (FCNN) for the automatic detection and segmentation of liver cancer in CT scans, utilizing a mathematical model to optimize the segmentation process.

FCNNs are particularly well-suited for semantic segmentation tasks, enabling pixel-level classification that is essential for differentiating between cancerous and non-cancerous liver lesions. Accurately distinguishing between malignant lesions such as colorectal cancer liver metastases and benign conditions like cysts is crucial, as the diagnosis directly influences the course of treatment. In traditional clinical settings, this differentiation relies heavily on the expertise of radiologists and pathologists, requiring extensive resources and specialized skills. However, the introduction of deep learning techniques, including FCNNs, offers a scalable and efficient alternative. By leveraging large datasets of annotated CT scans, FCNNs can learn complex patterns and improve the accuracy of lesion segmentation, thus reducing the burden on medical professionals.

In addition to FCNNs, this research also explores the implementation of convolutional neural networks (CNNs) and AlexNet models to develop a binary

classifier capable of distinguishing between healthy livers and those affected by hemochromatosis—a condition characterized by excessive iron accumulation in the liver. Both models were trained on a dataset of liver images to evaluate their performance in classifying liver health. The results demonstrate that AlexNet achieved a classification accuracy of 95%, outperforming the CNN model, which achieved an accuracy of 90%. AlexNet's superior performance can be attributed to its deeper architecture and more sophisticated feature extraction capabilities, enabling it to capture finer details in the liver tissue images.

This research highlights the potential of deep learning models in revolutionizing liver cancer diagnosis by automating the identification and classification of liver lesions with high accuracy. By integrating FCNNs for segmentation and AlexNet for classification, the study provides a robust framework for enhancing the diagnostic process. The findings suggest that deep learning models, particularly AlexNet, can significantly improve diagnostic accuracy, reduce the workload on healthcare professionals, and potentially lead to earlier detection and better clinical outcomes for patients with liver disease. As the field of AI in medical imaging continues to evolve, future studies could focus on refining these models through larger datasets, incorporating multimodal data, and addressing challenges related to model interpretability and clinical validation to ensure widespread adoption in clinical practice.

CHAPTER 3

SYSTEM DESIGN

3.1 EXISTING SYSTEM:

The traditional liver disease diagnosis system involves doctors or medical professionals using various medical tests, such as blood tests, biopsy, and imaging techniques like ultrasound, MRI, or CT scans to identify liver disease in patients. The interpretation of the test results and diagnosis is done by medical professionals based on their experience and knowledge. This approach can be 9 time-consuming and costly, and the accuracy of diagnosis may depend on the skills and experience of the medical professionals.

ISSUES IN EXISTING METHODOLOGY: Liver disease prediction systems have played a vital part in people's lives, and many scholars believe it to be an important topic. Although the results of the forecast are promising, these old methods are still far from being highly precise and efficient. Existing systems are straightforward and effective, but they are extremely sensitive to disruption. Furthermore, state-of-the-art methods only use one algorithm, resulting in erroneous findings. This could lead to erroneous assumptions and incorrect diagnoses and treatments for patients. The existing methodology had an algorithm called SVM algorithm to calculate that disease details. SVM algorithm is slow algorithm for classifying and it also gives less accuracy. In that main disadvantage is time efficiency. Details of patient liver diseases start from large scale, diverse, fully independent and distributed and seek to explore complex and evolving patterns between data. Existing system shows an algorithm called SVM [Support Vector Deep] theorem that characterizes the options of the massive information revolt, and implement an enormous processing model.

Liver disease is the leading reason of death worldwide , The liver is responsible for metabolic, strength-storing, and waste filtering functioning in your body. The aim of this study is to developing a deep learning based technique for liver disease prediction in people. This study on liver disease detection models meant to determine the best techniques for selecting and synthesising the many studies of high quality. The majority of health data is nonlinear, correlation-structured, and

complex, make it complex to evaluate. The use of ML based techniques in healthcare has been ruled out. In this work use various deep learning algorithim like decision tree, Naïve Bayes, SVM , Random Forest , CatBoost ,Soft Voting Classifier on Indian Liver patient dataset to predict liver disease. The research work gives the correct or maximum accuracy model show that the model is able of predicting liver diseases effectively. Our end result shows that voting classifier attain the higher accuracy as compared to other deep learning models

3.2 PROPOSED SYSTEM:

Our proposed system represents a ground breaking advancement in medical diagnostics, offering a comprehensive solution for the early detection of liver infections through image analysis. By integrating cutting-edge object detection techniques using YOLO (You Only Look Once) in Python, our system exhibits unparalleled accuracy and efficiency in assessing damaged liver images. Through intricate algorithms and deep learning capabilities, it not only identifies the presence of infection but also provides a nuanced analysis, assigning a confidence percentage to each diagnosis.

The proposed liver tumour detection system utilizes state-of-the-art deep learning algorithms, specifically YOLO, for efficient and accurate identification of tumour regions within scanned liver images. The system workflow involves the following steps:

1. **Image Acquisition:** Scanned liver images are obtained from medical imaging devices or archival databases.
2. **Object Detection:** The YOLO model is applied to the input images to detect and localize liver tumours. YOLO's real-time detection capabilities enable rapid processing of images, allowing for swift identification of tumour regions.

3. **Tumour Classification:** Detected tumour regions are classified based on their characteristics, such as size, shape, and texture, to determine the stage or severity of the tumour. Classification may involve distinguishing between different tumour stages, such as early-stage, intermediate-stage, and advanced-stage tumours.

4. **Outcome Prediction:** Predictive analytics techniques may be employed to assess the prognosis of detected tumours and predict patient outcomes based on tumour characteristics and other clinical factors.

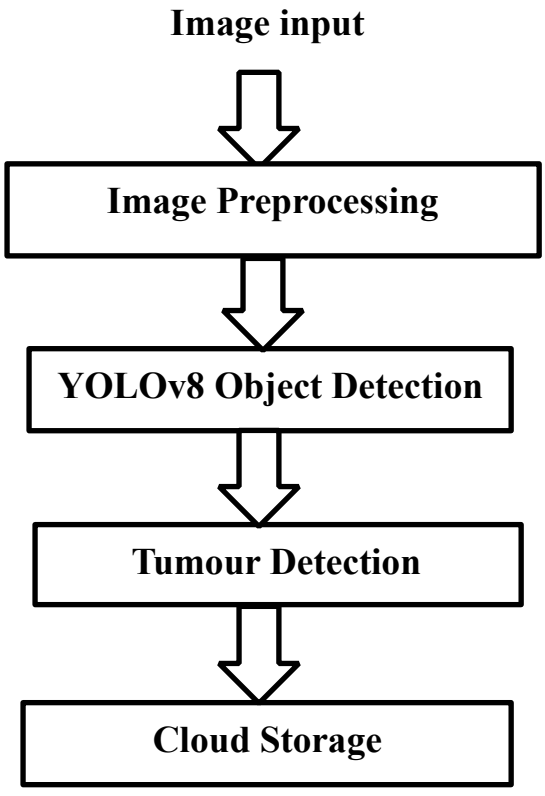
5. **Reporting and Integration:** The system generates comprehensive reports detailing the location, size, and characteristics of detected tumours, providing valuable insights for healthcare professionals. Additionally, the system may integrate with existing healthcare infrastructure for seamless data exchange and collaboration.

6. **Patient Management:** The detected tumour information is used to guide treatment decisions and patient management strategies, including surgery, chemotherapy, or radiotherapy. Continuous monitoring and follow-up assessments are conducted to track tumour progression and treatment response.

7. **Cloud Server Implementation:** The proposed system enhances the existing liver tumour detection system by integrating a cloud-based storage solution. This integration involves establishing a connection with a cloud storage service provider. Input images, output images with annotated tumour detection results, and corresponding metadata are uploaded to the cloud storage, ensuring secure and efficient data management. The system utilizes cloud links to retrieve input images for processing, store output images with detection results, and provide access to the results database for analysis and visualization. By integrating advanced deep learning

algorithms with medical imaging technology, the proposed system offers a robust solution for liver tumour detection and management. The system's ability to accurately identify tumour regions and provide actionable insights contributes to improved patient outcomes and enhanced healthcare delivery in the field of liver disease management. Future enhancements may include the integration of additional diagnostic modalities and the development of personalized treatment plans tailored to individual patient profiles.

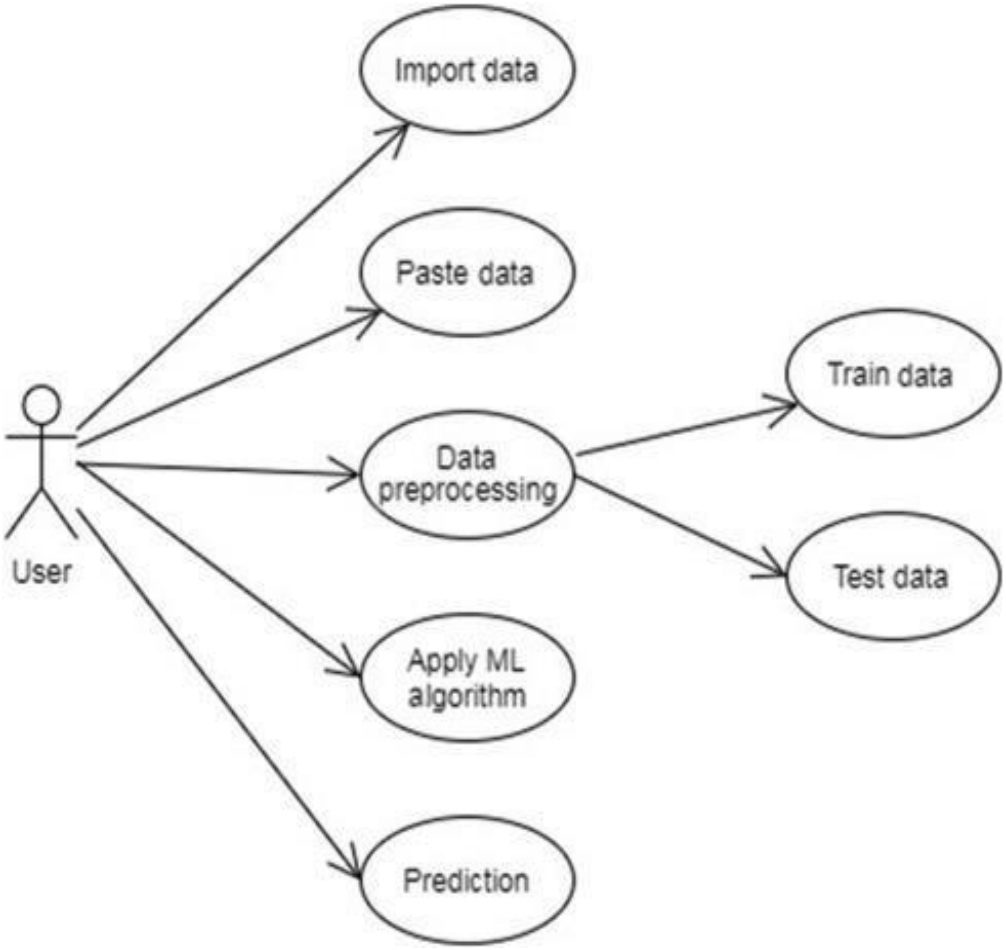
3.3 FLOW DIAGRAM:



3.4 USE CASE DIAGRAM:

Use case diagrams are usually reffered to as behaviour diagrams used to describe a set of actions(use cases) that some system or systems(subject)should

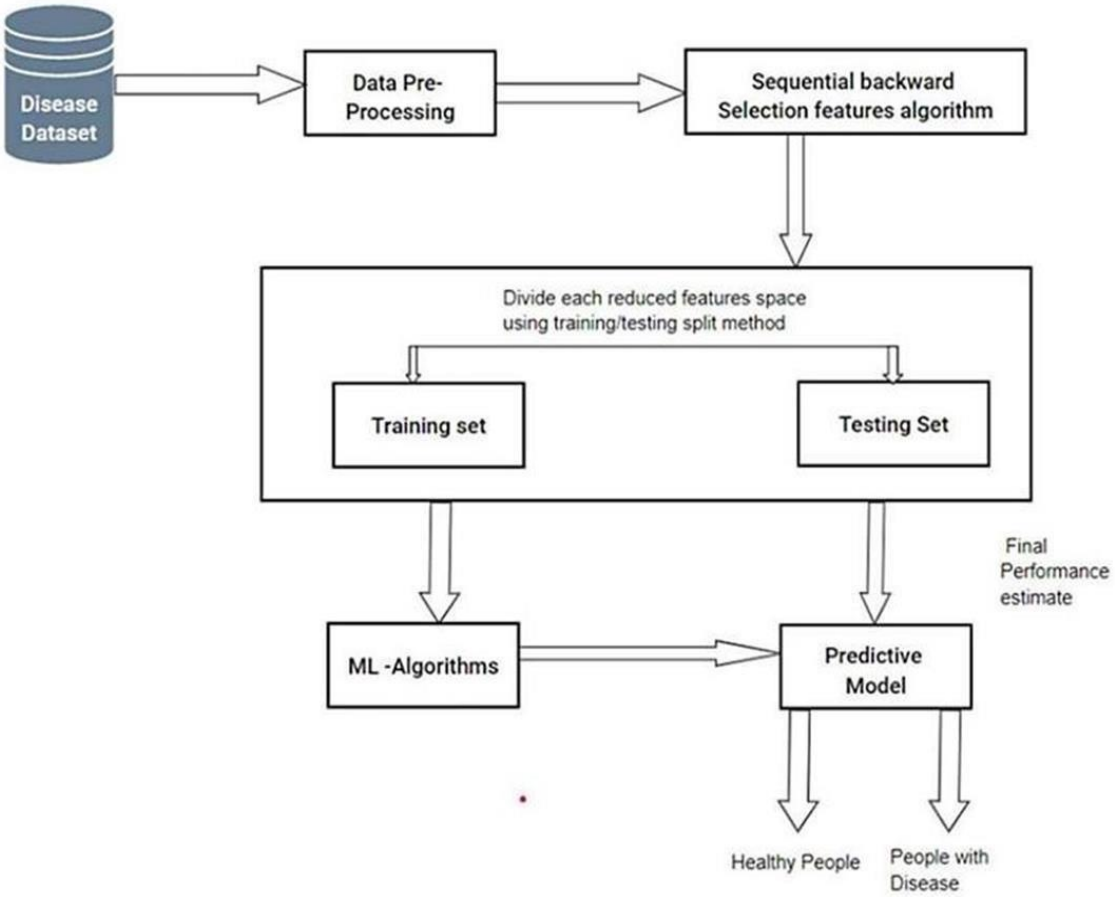
or can perform in collaboration with one or more external users of the system(actors). Each use case should provide some observiable and valuable result to the actors or other stakeholders of the system.



3.5 SEQUENCE DIAGRAM:

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence

chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realization in the Logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.



CHAPTER 4

SYSTEM REQUIREMENT SPECIFICATION

This chapter describes about the requirements. It specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes overview of dissertation as well as the functional and nonfunctional requirement of this dissertation.

A SRS document describes all data, functional and behavioural requirements of the software under production or development. SRS is a fundamental document, which forms the foundation of the software development process. It is the complete description of the behaviour of a system to be developed. Requirement Analysis discusses the conditions to be met for a new or altered product. Requirement Analysis is critical to the success to a development project. Requirement must be documented, measurable, testable, related to in identified business needs or opportunities, and defined to a level of detail sufficient for system design. The SRS functions as a blueprint for completing a project. The goal of preparing the SRS document is to:

- Facilitate communication between the customer, analyst, system developers, maintainers.
- To form a foundation for the design phase.
- Support system testing facilities.
- Controlling the evolution of the system.

- **Python**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming 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.

- **GUI**

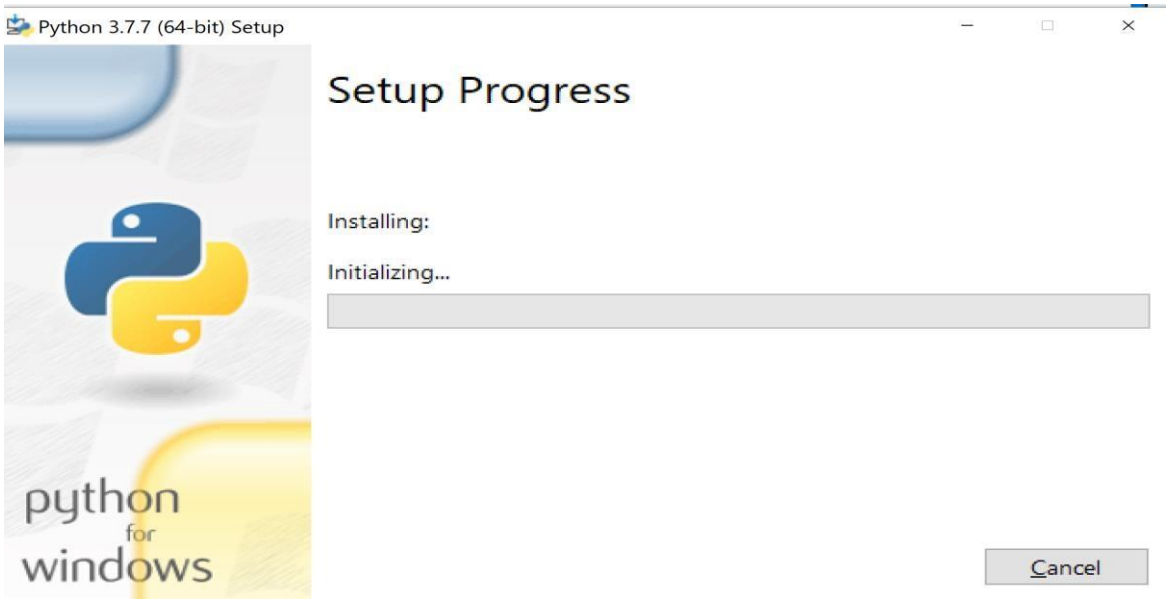
Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.

4.1 HARDWARE REQUIREMENT:

- Processor: Intel i3/i5/i7
- Ram: 4 GB
- Hard disk: 160 GB
- Monitor: 18inch Lcd/Led
- Webcam

4.2 SOFTWARE REQUIREMENT:

- OS: Windows 8/10/11
- Editor: VS Code
- Python 3.7
- Python with its neural network libraries
- Keras – module for deep learning



4.3 NON-FUNCTIONAL REQUIREMENTS:

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours. They may relate to emergent system properties. Non-functional requirements are requirements that are not specifically concerned with the functionality of a system. They normally place restrictions on the product being developed and the development process. Non-functional requirements may be regarded as parameters of functionality in that they determine how quickly, how accurately, how reliably, how securely, etc., functions must operate. Some of the ALPR non-functional requirements are as follows:

- The system may issue a receipt to remove any papers printed and make it a green initiative
- The system must be working at 100% peak efficiency
- When checking the database for errors, a 100% scan of the data is required, rather than selecting a sample set

- A process must be devised to support normal precinct business hours
- The system should provide documentation to inform users of system functionality and any change to the system
- The system should provide friendly graphical Interface to ensure ease of use when end users utilize system functionality.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES

IMAGE UPLOAD MODULE:

Allows users to upload images for tumor detection with validation for format and size.

MODEL INTEGRATION MODULE:

Handles the integration of the YOLOv8 model to process uploaded images and detect tumors.

DETECTION RESULT MODULE:

Displays detection results,including bounding boxes,confidence scores,and tumor classifications on the upload image.

USER INTERFACE MODULE:

Provides a user-friendly interface for uploading images and viewing results.

BACK END MODULE:

Manages backend operations such as handling image uploads, invoking the Yolov8 model, and returning results.

5.2 MODULES DESCRIPTIONS

5.2.1 OpenCV:

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Deep Learning and is expanding day by day. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X,

Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCV are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language. OpenCV-Python OpenCV-Python is a library of Python bindings designed to solve computer vision problems. Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib. Applications of OpenCV: There are lots of applications which are solved using OpenCV; some of them are listed below

- Face recognition
- Automated inspection and surveillance
- Number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations

- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

OpenCV Functionality

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Deep learning & clustering (ml, flann)
- CUDA acceleration (gpu)

Image-Processing:

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it. If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”.

Digital-Image:

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point. In another word An image is nothing more than a two-dimensional matrix (3-D in case of colored images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what color it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image. Image processing basically includes the following three steps:

1. Importing the image
2. Analysing and manipulating the image
3. Output in which result can be altered image or report that is based on image analysis

Haar Cascade Algorithm:

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001. The algorithm is given a lot of positive images consisting of faces, and a lot of negative 22 images not consisting

of any face to train on them. The model created from this training is available at the OpenCV GitHub repository.

Feature:

The first contribution to the research was the introduction of the haar features. These features on the image make it easy to find out the edges or the lines in the image, or to pick areas where there is a sudden change in the intensities of the pixels.

The objective here is to find out the sum of all the image pixels lying in the darker area of the haar feature and the sum of all the image pixels lying in the lighter area of the haar feature. And then find out their difference. Now if the image has an edge separating dark pixels on the right and light pixels on the left, then the haar value will be closer to 1. That means, we say that there is an edge detected if the haar value is closer to 1. In the example above, there is no edge as the haar value is far from 1.

This is just one representation of a particular haar feature separating a vertical edge. Now there are other haar features as well, which will detect edges in other directions and any other image 24 structures. To detect an edge anywhere in the image, the haar feature needs to traverse the whole image.

The haar feature continuously traverses from the top left of the image to the bottom right to search for the particular feature. This is just a representation of the whole concept of the haar feature traversal. In its actual work, the haar feature would traverse pixel by pixel in the image. Also all possible sizes of the haar features will be applied.

Depending on the feature each one is looking for, these are broadly classified into three categories. The first set of two rectangle features is responsible for finding out the edges in a horizontal or in a vertical direction (as shown above). The

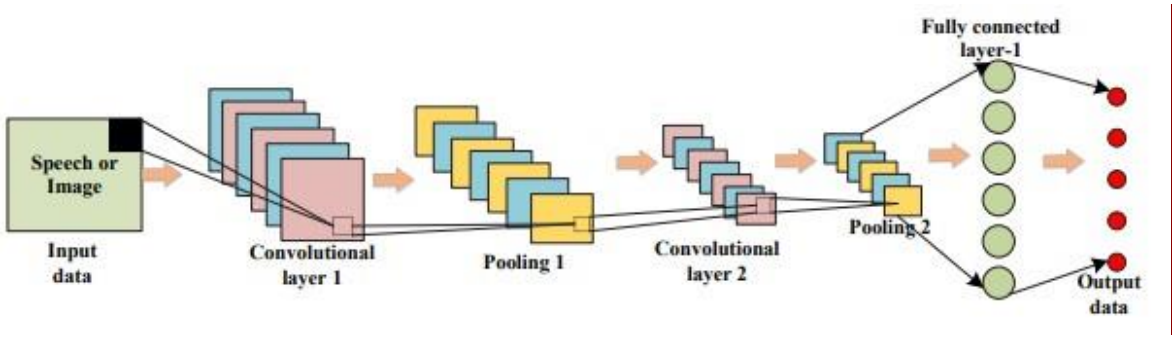
second set of three rectangle features are responsible for finding out if there is a lighter region surrounded by darker regions on either side or vice-versa. The third set of four rectangle features is responsible for finding out change of pixel intensities across diagonals.

Now, the haar features traversal on an image would involve a lot of mathematical calculations. As we can see for a single rectangle on either side, it involves 18 pixel value additions (for a rectangle enclosing 18 pixels). Imagine doing this for the whole image with all sizes of the haar features. This would be a hectic operation even for a high performance deep.

To tackle this, they introduced another concept known as The Integral Image to perform the same operation. An Integral Image is calculated from the Original Image in such a way that each pixel in this is the sum of all the pixels lying in its left and above in the Original Image. The calculation of a pixel in the Integral Image can be seen in the above GIF. The last pixel at the bottom right corner of the Integral Image will be the sum of all the pixels in the Original Image.

Training Set Selection: Each weak classifier should be trained on a random subset of the total training set. The subsets can overlap–it's not the same as, for example, dividing the training set into ten portions. AdaBoost assigns a "weight" to each training example, which determines the probability that each example should appear in the training set. Examples with higher weights are more likely to be included in the training set, and vice versa. After training a classifier, AdaBoost increases the weight on the misclassified examples so that these examples will make up a larger part of the next classifiers training set, and hopefully the next classifier trained will perform better on them. The equation for this weight update step is detailed later on.

5.2.2 CONVOLUTED NEURAL NETWORK (CNN):



A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well suited for image processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image. CNNs are trained using a large dataset of labelled images, where the network learns to recognize patterns and features that are associated with specific objects or classes. Once trained, a CNN can be used to classify new images, or extract features for use in other applications such as object detection or image segmentation.CNNs have achieved state-of-the-art performance on a wide range of image recognition tasks, including object classification, object detection, and image segmentation. They are widely used in computer vision, image processing, and other related fields, and have been applied to a wide range of applications, including selfdriving cars, medical imaging, and security systems.

Convolutional Neural Network Design:

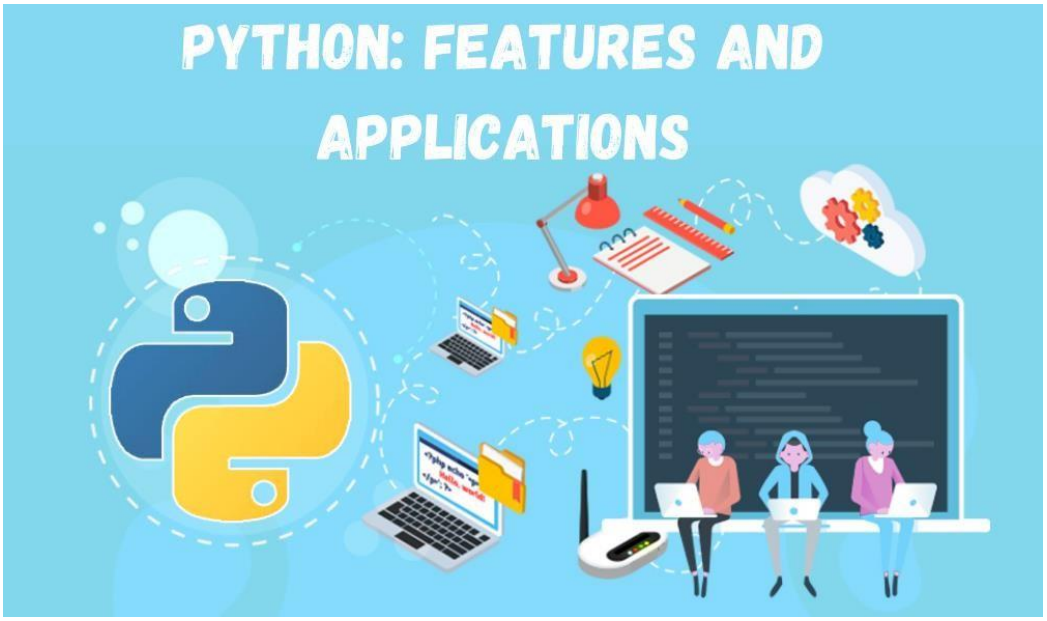
- The construction of a convolutional neural network is a multi-layered feedforward neural network, made by assembling many unseen layers on top of each other in a particular order.
- It is the sequential design that give permission to CNN to learn hierarchical attributes.
- In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.
- The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

Different Types of CNN Models:

1. LeNet
2. AlexNet
3. ResNet
4. GoogleNet
5. MobileNet
6. VGG

Classifier Output Weights:

After each classifier is trained, the classifier's weight is calculated based on its accuracy. More accurate classifiers are given more weight. A classifier with 50% accuracy is given a weight of zero, and a classifier with less than 50% accuracy (kind of a funny concept) is given negative weight.



WHY PYTHON:

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems.

Python provides many useful features to the programmer. These features make it the most popular and widely used language. We have listed below few-essential features of Python.

- **Easy to use and Learn:** Python has a simple and easy-to-understand syntax, unlike traditional languages like C, C++, Java, etc., making it easy for beginners to learn.
- **Expressive Language:** It allows programmers to express complex concepts in just a few lines of code or reduces Developer's Time.
- **Interpreted Language:** Python does not require compilation, allowing rapid development and testing. It uses Interpreter instead of Compiler.
- **Object-Oriented Language:** It supports object-oriented programming, making writing reusable and modular code easy.
- **Open Source Language:** Python is open source and free to use, distribute and modify.
- **Extensible:** Python can be extended with modules written in C, C++, or other languages.
- **Learn Standard Library:** Python's standard library contains many modules and functions that can be used for various tasks, such as string manipulation, web programming, and more.
- **GUI Programming Support:** Python provides several GUI frameworks, such as Tkinter and PyQt, allowing developers to create desktop applications easily.
- **Integrated:** Python can easily integrate with other languages and technologies, such as C/C++, Java, and . NET.
- **Embeddable:** Python code can be embedded into other applications as a scripting language.
- **Dynamic Memory Allocation:** Python automatically manages memory allocation, making it easier for developers to write complex programs without worrying about memory management.

- **Wide Range of Libraries and Frameworks:** Python has a vast collection of libraries and frameworks, such as NumPy, Pandas, Django, and Flask, that can be used to solve a wide range of problems.
- **Versatility:** Python is a universal language in various domains such as web development, deep learning, data analysis, scientific computing, and more.
- **Large Community:** Python has a vast and active community of developers contributing to its development and offering support. This makes it easy for beginners to get help and learn from experienced developers.
- **Career Opportunities:** Python is a highly popular language in the job market. Learning Python can open up several career opportunities in data science, artificial intelligence, web development, and more.
- **High Demand:** With the growing demand for automation and digital transformation, the need for Python developers is rising. Many industries seek skilled Python developers to help build their digital infrastructure.
- **Increased Productivity:** Python has a simple syntax and powerful libraries that can help developers write code faster and more efficiently. This can increase productivity and save time for developers and organizations.
- **Big Data and Deep Learning:** Python has become the go-to language for big data and deep learning. Python has become popular among data scientists and deep learning engineers with libraries like NumPy, Pandas, Scikit-learn, TensorFlow, and more.

5.2.3 Python Popular Frameworks and Libraries:

Python has wide range of libraries and frameworks widely used in various fields such as deep learning, artificial intelligence, web applications, etc. We define some popular frameworks and libraries of Python as follows.

- **Web development (Server-side) -** Django Flask, Pyramid, CherryPy ○ **GUIs based applications -** Tk, PyGTK, PyQt, PyJs, etc.
- **Machine Learning -** TensorFlow, PyTorch, **Scikit-learn**, Matplotlib, Scipy, etc.
- **Mathematics -** Numpy, Pandas, etc.
- **BeautifulSoup:** a library for web scraping and parsing HTML and XML ○ **Requests:** a library for making HTTP requests ○ **SQLAlchemy:** a library for working with SQL databases ○ **Kivy:** a framework for building multi-touch applications ○ **Pygame:** a library for game development ○ **Pytest:** a testing framework for Python
- **Django REST framework:** a toolkit for building RESTful APIs ○ **FastAPI:** a modern, fast web framework for building APIs
- **Streamlit:** a library for building interactive web apps for machine learning and data science
- **NLTK:** a library for natural language processing.

Scripting Language:

A scripting or script language is a programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator.

Scripting languages are often interpreted (rather than compiled). Primitives are usually the elementary tasks or API calls, and the language allows them to be combined into more complex programs. Environments that can be automated through scripting include software applications, web pages within a web browser, the shells of operating systems (OS), embedded systems, as well as numerous games. A scripting language can be viewed as a domain-specific language for a particular environment; in the case of scripting an application, this is also known as an extension language. Scripting languages are also sometimes referred to as very high-level programming languages, as they operate at a high level of abstraction, or as control languages.

Object Oriented Programming Language:

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OO programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type.

Data Pre-Processing:

For the CBIS-DDSM and INbreast datasets, the coordinates of the ROIs bounding box required for Yolo training were calculated considering the coordinates of the smallest rectangle containing the segmented lesion. Instead, the ROI coordinates for the proprietary dataset were computed from the square region that inscribes

the circle containing the ROI. The CBIS-DDSM dataset has an acceptable size for deep learning architecture training.

However, it is com-posed of scanned film mammograms, much noisier and less detailed than FFDM. For this reason, only for the CBIS-DDSM dataset, the contrast limited adaptive histogram equalization (CLAHE) was applied for image enhancement, with the following setting: 1 as contrast limit, 2×2 as grid size, followed by a 3×3 Gaussian filter. For all data-sets, the gray levels were scaled in the range 0–255, and the images were resized to 640 ×640 using the Lanczos filter. The CBIS-DDSM dataset was spitted randomly considering 70% training, 15% validation, and 15% test set. Conversely, the INbreast and the proprietary datasets were split into training (80%) and test set (20%), respectively. Considering the small size of the two datasets and the unbalanced issue, the next "Data Augmentation" discusses data augmentation for class balancing and generation of the validation set "Techniques for Class Balancing Before the Training Phase", as well as the procedure to improve the training "Techniques Used During the Training Phase".

Data Augmentation Techniques for Class Balancing Before the Training Phase. Due to the excessive imbalance classes for the INbreast and proprietary dataset, the minority class images (benign) of the training set were augmented. Although the main purpose of the work is to evaluate the detection performance on the proprietary dataset (regardless of lesion class), the following data augmentation procedure was applied to the proprietary dataset before the training phase.

Figure 2 sum-marizes the transformation considered. In particular, 180◦ rotation and 180◦ rotation + flip upper-down (UD) were applied for benign images. The other transformations were applied during the training of Yolo, as discussed in the next subsection "Techniques Used during the Training Phase".

In addition, according to, the remaining test dataset was augmented to obtain the validation set. In fact, flip UD, 180◦ rotation + flip UD, flip left-right (LR) and 180◦ rotation were applied on benign images, and Flip LR for malignant images. Considering the smaller difference between the classes, on INbreast, also, 180◦ rotations for malignant masses was considered [9].This procedure resulted in the generation of a balanced validation set. In addition, the discussed procedure for INbreast and the proprietary datasets was repeated considering 5 different splitting of training and test sets (5-fold cross-validation).

Yolo Architectures Training:

Like other single-stage object detectors, Yolo consists of three parts: backbone, neck, and head. The backbone part is a CNN that extracts and aggregates image features. The neck part allows for features extraction optimized for small, medium, and large object detection. In the end, the three feature maps for small, medium, and large object detection are given as input to the head part, thus composed of convolutional layers for the final prediction. Yolo requires that the image is divided into a grid, and then makes a prediction for each grid cell. The prediction consists of a 6-tuple y=(pc, bx, by, bh, bw, c) , where (bx,by,bh,bw) identify coordinates (x, y) and sizes (height, width) of the predicted boundingbox, pc represent the probability that there is an object in the cell, and c represent the predicted class. The mechanism of anchors is also used, to allow multiple object detection in the same grid cell. For this reason, the prediction is the 6-tuple discussed for each specified anchor. Each version of Yolo has its own peculiarities, which mainly concern the structure of the feature extractor, that is, the backbone.

5.2.4 YoloV3 Model:

YoloV3 is much deeper than the previous two versions and is more accurate but requires more time and data for training. In YoloV3, the Darknet53 was used as backbone. Darknet53 is a hybrid approach between Darknet19 (used in YoloV2) and residual network elements (e.g., BottleNeck), proposed to improve the Darknet19 and the efficiency of ResNet-101/152. The short-cut connections allow getting more fine-grained information, leading to better performance for small objects. The feature pyramids network (FPN) is used as neck, allowing learning objects of different sizes: it specializes in detecting large and small objects. In addition, the non-maximum suppression to select one bounding box out of many overlap-ping bounding boxes is used.YoloV5 ModelYoloV5 uses CSPDarknet53 as its backbone: it exploits the architecture Darknet53 proposed in and employs a CSPNet strategy to partition the feature map of the base layer into two parts and then merges them through a cross-stage hierarchy. In the neck part, PAnet is used to generate the feature pyramids network (FPN) and allow the extraction of multi-scale feature maps. This structure allows the extraction of features optimized for small, medium, and large object detection. YoloV5 was released in nano, small, medium, large, and extra-large versions. The versions differ in the number of convolutional kernels used and thus the number of parameters. In this paper, a comparison between nano, small, medium, and large versions was performed.

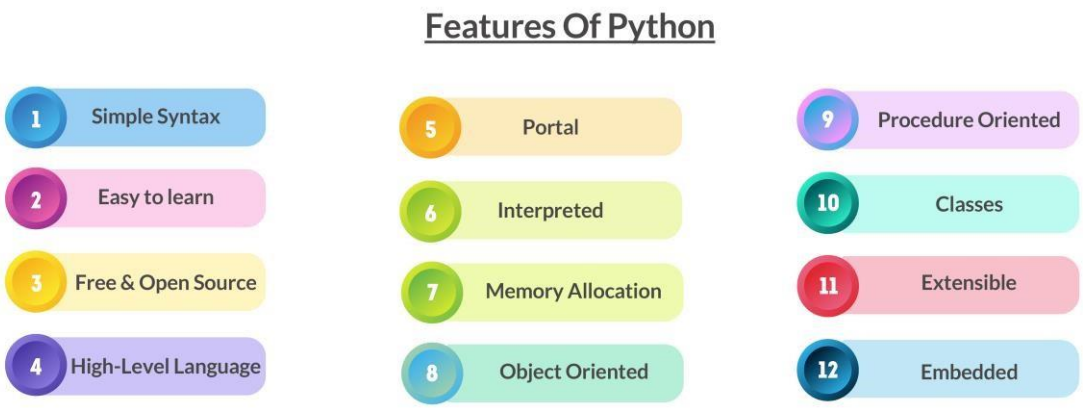
5.2.5 YoloV5-Transformer:

In contrast to convolutional networks, Transformer are able to model the relationships among various small patches in the image. The Transformer block assumes the image is split into a sequence of patches, where each patch is flattened to a vector. These flattened image patches are used to create lowerdimensional linear embeddings and fed into a Transformer encoder,

composed by a multi-head attention to find local and global dependencies in the image. It has been shown that the introduction of a Transformer block to convolutional networks can improve efficiency and overall accuracy. In YoloV5, the Transformer block was embedded in the penultimate layer of the backbone, that is, among the three convolutional layers preceding the spatial pyramid pooling layer.

Python Features:

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.



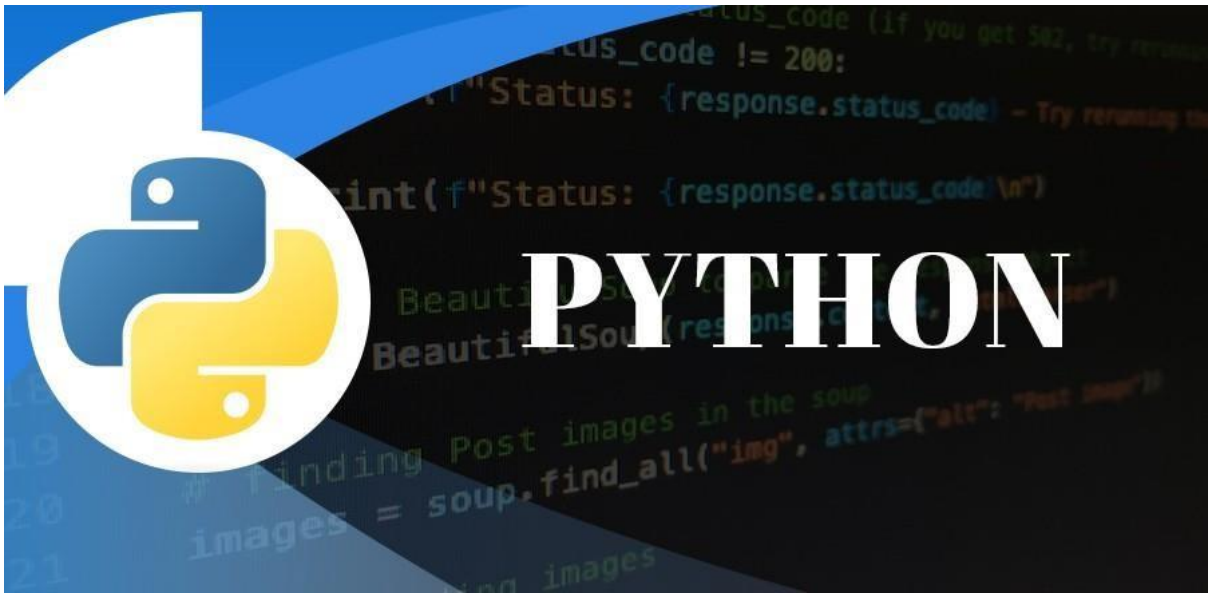
1) Easy to Learn and Use

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

2) Expressive Language

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type print("Hello World"). It will take only one line to execute, while Java or C takes multiple lines.

3) Interpreted Language



Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

5) Free and Open Source

Python is freely available for everyone. It is freely available on its official website www.python.org. It has a large community across the world that is dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

6) Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code.

8) Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

9) GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

10) Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C,C++ Java. It makes easy to debug the code.

11. Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

12. Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at run time. Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

Python Applications:

Python is known for its general-purpose nature that makes it applicable in almost every domain of software development. Python makes its presence in every emerging field. It is the fastest-growing programming language and can develop any application.

Python is a general-purpose, popular programming language, and it is used in almost every technical field. The various areas of Python use are given below.

- **Data Science:** Data Science is a vast field, and Python is an important language for this field because of its simplicity, ease of use, and availability of powerful data analysis and visualization libraries like NumPy, Pandas, and Matplotlib.
- **Desktop Applications:** PyQt and Tkinter are useful libraries that can be used in GUI - Graphical User Interface-based Desktop Applications. There are better languages for this field, but it can be used with other languages for making Applications.
- **Console-based Applications:** Python is also commonly used to create command-line or console-based applications because of its ease of use and support for advanced features such as input/output redirection and piping.
- **Mobile Applications:** While Python is not commonly used for creating mobile applications, it can still be combined with frameworks like Kivy or BeeWare to create cross-platform mobile applications.
- **Software Development:** Python is considered one of the best softwaremaking languages. Python is easily compatible with both from Small Scale to Large Scale software.
- **Artificial Intelligence:** AI is an emerging Technology, and Python is a perfect language for artificial intelligence and machine learning because of the availability of powerful libraries such as TensorFlow, Keras, and PyTorch.
- **Web Applications:** Python is commonly used in web development on the backend with frameworks like Django and Flask and on the front end with tools like JavaScript and HTML.

- **Enterprise Applications:** Python can be used to develop large-scale enterprise applications with features such as distributed computing, networking, and parallel processing.

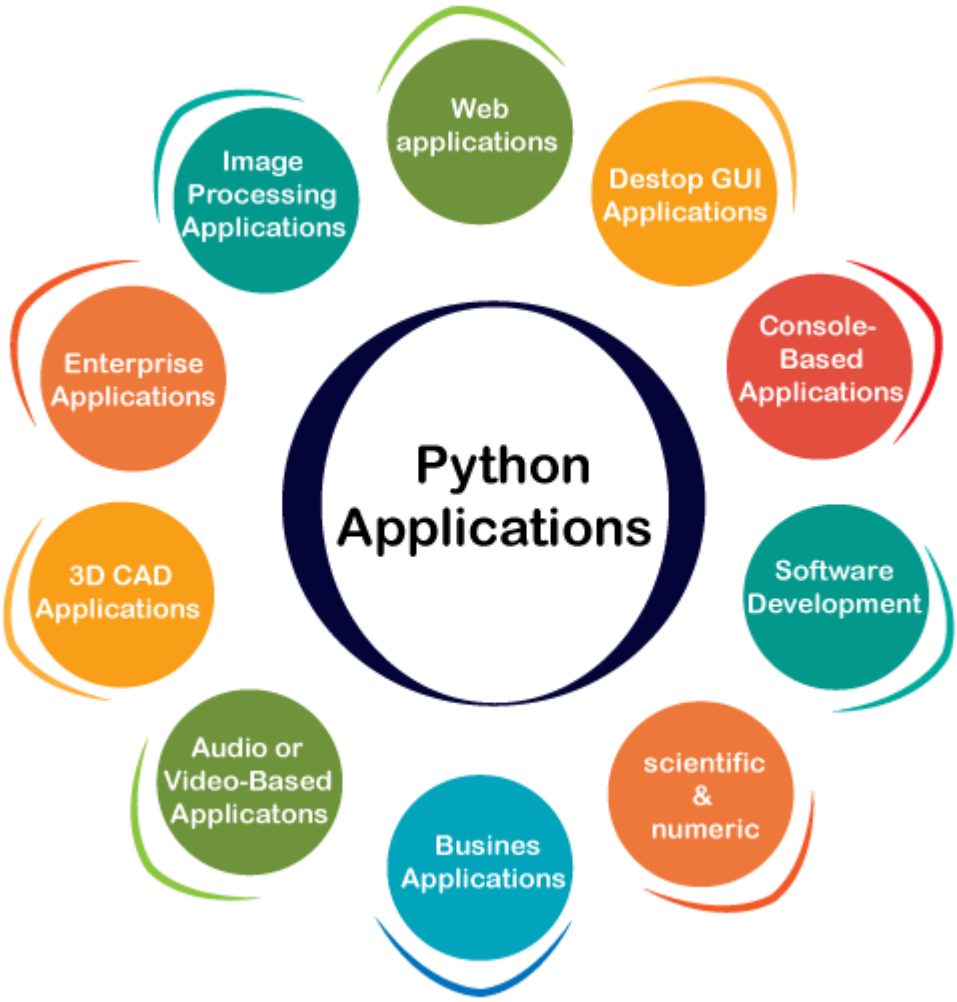
  3D CAD Applications: Python can be used for 3D computer-aided design (CAD) applications through libraries such as Blender.
- **Machine Learning:** Python is widely used for machine learning due to its simplicity, ease of use, and availability of powerful machine learning libraries.
- **Computer Vision or Image Processing Applications:** Python can be used for computer vision and image processing applications through powerful libraries such as OpenCV and Scikit-image.
- **Speech Recognition:** Python can be used for speech recognition applications through libraries such as SpeechRecognition and PyAudio.
- **Scientific computing:** Libraries like NumPy, SciPy, and Pandas provide advanced numerical computing capabilities for tasks like data analysis, machine learning, and more.
- **Education:** Python's easy-to-learn syntax and availability of many resources make it an ideal language for teaching programming to beginners.
- **Testing:** Python is used for writing automated tests, providing frameworks like unit tests and pytest that help write test cases and generate reports.
- **Gaming:** Python has libraries like Pygame, which provide a platform for developing games using Python.
- **IoT:** Python is used in IoT for developing scripts and applications for devices like Raspberry Pi, Arduino, and others.

- **Networking:** Python is used in networking for developing scripts and applications for network automation, monitoring, and management.
- **DevOps:** Python is widely used in DevOps for automation and scripting of infrastructure management, configuration management, and deployment processes.

  Finance: Python has libraries like Pandas, Scikit-learn, and Statsmodels for financial modeling and analysis.
- **Audio and Music:** Python has libraries like Pyaudio, which is used for audio processing, synthesis, and analysis, and Music21, which is used for music analysis and generation.
- **Writing scripts:** Python is used for writing utility scripts to automate tasks like file operations, web scraping, and data processing.

Here, we are specifying application areas where Python can be applied.

○


Python Applications
Web applications
Destop GUI Applications
Console-Based Applications
Software Development
scientific & numeric
Busines Applications
Audio or Video-Based Applicatons
3D CAD Applications
Enterprise Applications
Image Processing Applications

1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser, etc. One of Python web-framework named Django is used on Instagram. Python provides many useful frameworks, and these are given below:

- Django and Pyramid framework(Use for heavy applications) ○ Flask and Bottle (Micro-framework)
- Plone and Django CMS (Advance Content management)

2) Desktop GUI Applications

The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a **Tk GUI library** to develop a user interface. Some popular GUI libraries are given below

- Tkinter or Tk ○ wxWidgetM
- Kivy (used for writing multitouch applications) ○ PyQt or Pyside

3) Console-based Application

Console-based applications run from the command-line or shell. These applications are computer program which are used commands to execute. This kind of application was more popular in the old generation of computers. Python can develop this kind of application very effectively. It is famous for having REPL, which means the Read-Eval-Print Loop that makes it the most suitable language for the command-line applications.

Python provides many free library or module which helps to build the commandline apps. The necessary IO libraries are used to read and write. It helps to parse argument and create console help text out-of-the-box. There are also advance libraries that can develop independent console apps.

4) Software Development

Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

- SCons is used to build control.
- Buildbot and Apache Gumps are used for automated continuous compilation and testing.
- Round or Trac for bug tracking and project management.

5) Scientific and Numeric

This is the era of Artificial intelligence where the machine can perform the task the same as the human. Python language is the most suitable language for Artificial intelligence or machine learning. It consists of many scientific and mathematical libraries, which makes easy to solve complex calculations.

Implementing machine learning algorithms require complex mathematical calculation. Python has many libraries for scientific and numeric such as Numpy, Pandas, Scipy, Scikit-learn, etc. If you have some basic knowledge of Python, you need to import libraries on the top of the code. Few popular frameworks of machine libraries are given below.

- SciPy
- Scikit-learn

- NumPy
- Pandas
- Matplotlib

6) Business Applications

Business Applications differ from standard applications. E-commerce and ERP are an example of a business application. This kind of application requires extensively, scalability and readability, and Python provides all these features.

Oddo is an example of the all-in-one Python-based application which offers a range of business applications. Python provides a Tryton platform which is used to develop the business application.

7) Audio or Video-based Applications

Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are TimPlayer, cplay, etc.These are powerful yet easy-to-use Python library for developing games and other visually rich applications on Windows, macOS, and Linux. The few multimedia libraries are given below.

- Gstreamer
- Pyglet
- QT
- Phonon
- imutils

8) 3D CAD Applications

The CAD (Computer-aided design) is used to design engineering related architecture. It is used to develop the 3D representation of a part of a system. Python can create a 3D CAD application by using the following functionalities.

- Fandango (Popular)
- CAMVOX ○ HeeksCNC ○ AnyCAD ○ RCAM

9) Enterprise Applications

Python can be used to create applications that can be used within an Enterprise or an Organization. Some real-time applications are OpenERP, Tryton, Picalo, etc.

10) Image Processing Application

Python contains many libraries that are used to work with the image. The image can be manipulated according to our requirements. Some libraries of image processing are given below.

- ○ OpenCV ○ Pillow ○ SimpleITK

In this topic, we have described all types of applications where Python plays an essential role in the development of these applications. In the next tutorial, we will learn more concepts about Python.

What is machine learning?

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

IBM has a rich history with machine learning. One of its own, Arthur Samuel, is credited for coining the term, “machine learning” with his research (link resides outside ibm.com) around the game of checkers. Robert Nealey, the selfproclaimed checkers master, played the game on an IBM 7094 computer in 1962, and he lost to the computer. Compared to what can be done today, this feat seems trivial, but it’s considered a major milestone in the field of artificial intelligence.

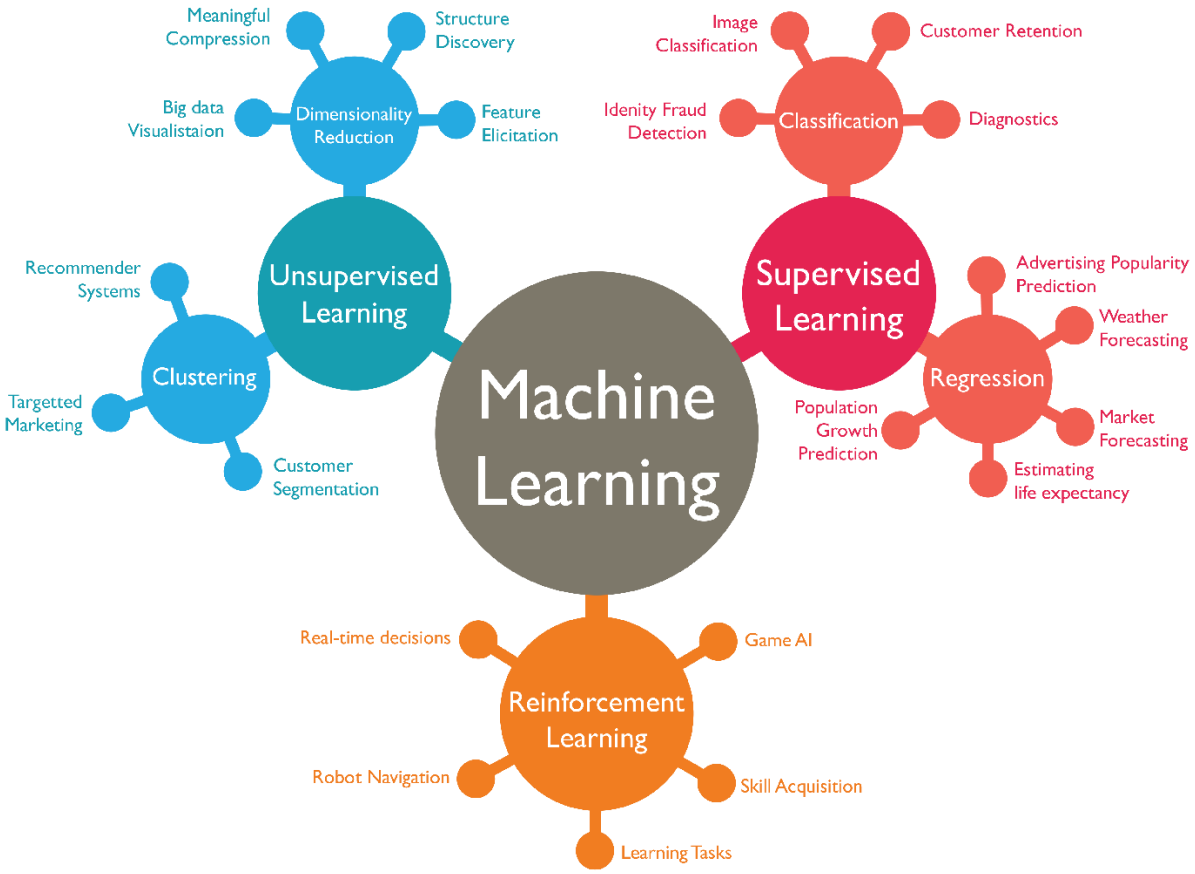
Over the last couple of decades, the technological advances in storage and processing power have enabled some innovative products based on machine learning, such as Netflix’s recommendation engine and self-driving cars.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase. They will be required to help identify the most relevant business questions and the data to answer them.

Machine learning algorithms are typically created using frameworks that accelerate solution development, such as TensorFlow and PyTorch.

How machine learning works:

1. A Decision Process: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labeled or unlabelled, your algorithm will produce an estimate about a pattern in the data.
2. An Error Function: An error function evaluates the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
3. A Model Optimization Process: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this "evaluate and optimize" process, updating weights autonomously until a threshold of accuracy has been met.



Machine learning methods

Machine learning models fall into three primary categories.

Supervised machine learning

Supervised learning, also known as supervised machine learning, is defined by its use of labelled datasets to train algorithms to classify data or predict outcomes accurately. As input data is fed into the model, the model adjusts its weights until it has been fitted appropriately. This occurs as part of the cross validation process to ensure that the model avoids overfitting or underfitting. Supervised learning helps organizations solve a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised learning include neural networks, linear regression, logistic regression, random forest, and support vector machine (SVM).

Unsupervised machine learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyse and cluster unlabelled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. This method's ability to discover similarities and differences in information make it ideal for exploratory data analysis, cross-selling strategies, customer segmentation, and image and pattern recognition. It's also used to reduce the number of features in a model through the process of dimensionality reduction. Principal component analysis (PCA) and singular value decomposition (SVD) are two common approaches for this. Other algorithms used in unsupervised learning include neural networks, k-means clustering, and probabilistic clustering methods.

Semi-supervised learning

Semi-supervised learning offers a happy medium between supervised and unsupervised learning. During training, it uses a smaller labelled data set to guide

classification and feature extraction from a larger, unlabelled data set. Semisupervised learning can solve the problem of not having enough labelled data for a supervised learning algorithm. It also helps if it's too costly to label enough data.

Common machine learning algorithms

A number of machine learning algorithms are commonly used. These include:

- **Neural networks:** Neural networks simulate the way the human brain works, with a huge number of linked processing nodes. Neural networks are good at recognizing patterns and play an important role in applications including natural language translation, image recognition, speech recognition, and image creation.
- **Linear regression:** This algorithm is used to predict numerical values, based on a linear relationship between different values. For example, the technique could be used to predict house prices based on historical data for the area.
- **Logistic regression:** This supervised learning algorithm makes predictions for categorical response variables, such as"yes/no" answers to questions. It can be used for applications such as classifying spam and quality control on a production line.
- **Clustering:** Using unsupervised learning, clustering algorithms can identify patterns in data so that it can be grouped. Computers can help data scientists by identifying differences between data items that humans have overlooked.
- **Decision trees:** Decision trees can be used for both predicting numerical values (regression) and classifying data into categories. Decision trees use a branching sequence of linked decisions that can be represented with a tree

diagram. One of the advantages of decision trees is that they are easy to validate and audit, unlike the black box of the neural network.

- **Random forests:** In a random forest, the machine learning algorithm predicts a value or category by combining the results from a number of decision trees.

What is Deep Learning?

Deep learning is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions. You can use deep learning methods to automate tasks that typically require human intelligence, such as describing images or transcribing a sound file into text.

Why is deep learning important?

Artificial intelligence (AI) attempts to train computers to think and learn as humans do. Deep learning technology drives many AI applications used in everyday products, such as the following:

- Digital assistants
- Voice-activated television remotes
- Fraud detection
- Automatic facial recognition

It is also a critical component of emerging technologies such as self-driving cars, virtual reality, and more.

Deep learning models are computer files that data scientists have trained to perform tasks using an algorithm or a predefined set of steps. Businesses use deep learning models to analyse data and make predictions in various applications.

What are the uses of deep learning?

Deep learning has several use cases in automotive, aerospace, manufacturing, electronics, medical research, and other fields. These are some examples of deep learning:

- Self-driving cars use deep learning models to automatically detect road signs and pedestrians.
- Defence systems use deep learning to automatically flag areas of interest in satellite images.
- Medical image analysis uses deep learning to automatically detect cancer cells for medical diagnosis.
- Factories use deep learning applications to automatically detect when people or objects are within an unsafe distance of machines.

You can group these various use cases of deep learning into four broad categories—computer vision, speech recognition, natural language processing (NLP), and recommendation engines.

Computer vision

Computer vision is the computer's ability to extract information and insights from images and videos. Computers can use deep learning techniques to comprehend images in the same way that humans do. Computer vision has several applications, such as the following:

- Content moderation to automatically remove unsafe or inappropriate content from image and video archives
- Facial recognition to identify faces and recognize attributes like open eyes, glasses, and facial hair

- Image classification to identify brand logos, clothing, safety gear, and other image details

Speech recognition

Deep learning models can analyze human speech despite varying speech patterns, pitch, tone, language, and accent. Virtual assistants such as Amazon Alexa and automatic transcription software use speech recognition to do the following tasks:

- Assist call centre agents and automatically classify calls.
- Convert clinical conversations into documentation in real time.
- Accurately subtitle videos and meeting recordings for a wider content reach.

Natural language processing

Computers use deep learning algorithms to gather insights and meaning from text data and documents. This ability to process natural, human-created text has several use cases, including in these functions:

- Automated virtual agents and chatbots
- Automatic summarization of documents or news articles
- Business intelligence analysis of long-form documents, such as emails and forms
- Indexing of key phrases that indicate sentiment, such as positive and negative comments on social media

Recommendation engines

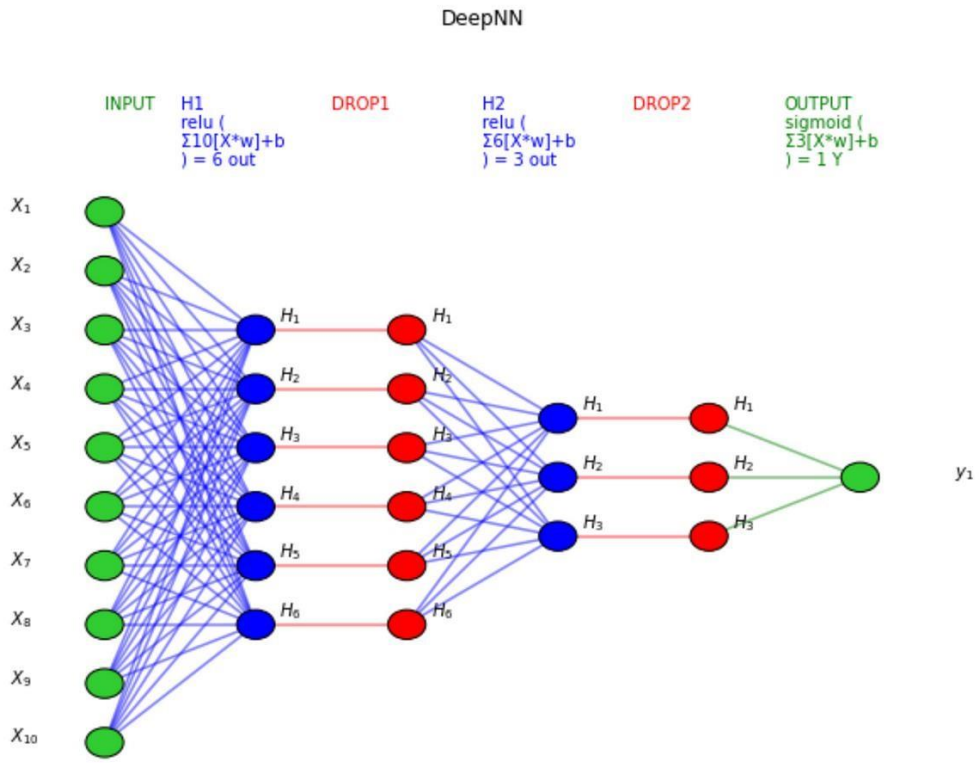
Applications can use deep learning methods to track user activity and develop personalized recommendations. They can analyse the behaviour of various users and help them discover new products or services. For example, many media and

entertainment companies, such as Netflix, Fox, and Peacock, use deep learning to give personalized video recommendations.

How does deep learning work?

Deep learning algorithms are neural networks that are modelled after the human brain. For example, a human brain contains millions of interconnected neurons that work together to learn and process information. Similarly, deep learning neural networks, or artificial neural networks, are made of many layers of artificial neurons that work together inside the computer.

Artificial neurons are software modules called nodes, which use mathematical calculations to process data. Artificial neural networks are deep learning algorithms that use these nodes to solve complex problems.



What are the components of a deep learning network?

The components of a deep neural network are the following.

Input layer

An artificial neural network has several nodes that input data into it. These nodes make up the input layer of the system.

Hidden layer

The input layer processes and passes the data to layers further in the neural network. These hidden layers process information at different levels, adapting their behavior as they receive new information. Deep learning networks have hundreds of hidden layers that they can use to analyze a problem from several different angles.

For example, if you were given an image of an unknown animal that you had to classify, you would compare it with animals you already know. For example, you would look at the shape of its eyes and ears, its size, the number of legs, and its fur pattern. You would try to identify patterns, such as the following:

- The animal has hooves, so it could be a cow or deer.
- The animal has cat eyes, so it could be some type of wild cat.

The hidden layers in deep neural networks work in the same way. If a deep learning algorithm is trying to classify an animal image, each of its hidden layers processes a different feature of the animal and tries to accurately categorize it.

Output layer

The output layer consists of the nodes that output the data. Deep learning models that output "yes" or "no" answers have only two nodes in the output layer. On the other hand, those that output a wider range of answers have more nodes.

What is deep learning in the context of machine learning?

Deep learning is a subset of machine learning. Deep learning algorithms emerged in an attempt to make traditional machine learning techniques more efficient. Traditional machine learning methods require significant human effort to train the software. For example, in animal image recognition, you need to do the following:

- Manually label hundreds of thousands of animal images.
- Make the machine learning algorithms process those images.
- Test those algorithms on a set of unknown images.
- Identify why some results are inaccurate.
- Improve the dataset by labeling new images to improve result accuracy.

This process is called supervised learning. In supervised learning, result accuracy improves only when you have a broad and sufficiently varied dataset. For instance, the algorithm might accurately identify black cats but not white cats because the training dataset had more images of black cats. In that case, you would need to label more white cat images and train the machine learning models once again.

What are the benefits of deep learning over machine learning?

A deep learning network has the following benefits over traditional machine learning.

Efficient processing of unstructured data

Machine learning methods find unstructured data, such as text documents, challenging to process because the training dataset can have infinite variations. On the other hand, deep learning models can comprehend unstructured data and make general observations without manual feature extraction. For instance, a

neural network can recognize that these two different input sentences have the same meaning:

- Can you tell me how to make the payment?
- How do I transfer money?

Hidden relationships and pattern discovery

A deep learning application can analyse large amounts of data more deeply and reveal new insights for which it might not have been trained. For example, consider a deep learning model that is trained to analyse consumer purchases. The model has data only for the items you have already purchased. However, the artificial neural network can suggest new items that you haven't bought by comparing your buying patterns to those of other similar customers.

Unsupervised learning

Deep learning models can learn and improve over time based on user behaviour. They do not require large variations of labelled datasets. For example, consider a neural network that automatically corrects or suggests words by analysing your typing behaviour. Let's assume it was trained in the English language and can spell-check English words. However, if you frequently type non-English words, such as *danke*, the neural network automatically learns and autocorrects these words too.

Volatile data processing

Volatile datasets have large variations. One example is loan repayment amounts in a bank. A deep learning neural network can categorize and sort that data as well, such as by analysing financial transactions and flagging some of them for fraud detection.

What are the challenges of deep learning?

As deep learning is a relatively new technology, certain challenges come with its practical implementation.

Large quantities of high-quality data

Deep learning algorithms give better results when you train them on large amounts of high-quality data. Outliers or mistakes in your input dataset can significantly affect the deep learning process. For instance, in our animal image example, the deep learning model might classify an airplane as a turtle if nonanimal images were accidentally introduced in the dataset.

To avoid such inaccuracies, you must clean and process large amounts of data before you can train deep learning models. The input data pre-processing requires large amounts of data storage capacity.

Large processing power

Deep learning algorithms are compute-intensive and require infrastructure with sufficient compute capacity to properly function. Otherwise, they take a long time to process results.

CHAPTER 6

CONCLUSION

In conclusion, the integration of a cloud-based storage solution represents a significant advancement in the liver tumour detection system. By incorporating cloud technology, the system enhances accessibility to input and output images, improves data management efficiency, and facilitates scalability for handling large volumes of data. Furthermore, the cloud-based architecture enhances system reliability and ensures seamless access to tumour detection results from anywhere at any time. Overall, this enhancement contributes to the effectiveness and usability of the liver tumour detection system, advancing its potential for early diagnosis and treatment of liver diseases.

This innovative prototype empowers healthcare professionals with timely and precise information, enabling proactive intervention and personalized treatment plans. With its potential to revolutionize medical imaging, our system stands at the forefront of the fight against liver infections, promising improved patient outcomes and enhanced healthcare delivery. It also assists medical faculties to take the necessary steps accordingly in prior.

CHAPTER 7

FUTURE ENHANCEMENT

A key future enhancement involves upgrading the system to utilize YOLOv11, the next-generation object detection model, which promises improved accuracy, faster processing speeds, and enhanced capability to detect smaller and more complex liver tumors.

YOLOv11's advanced architecture will enable more precise localization of tumors, even in challenging cases, thereby reducing false positives and false negatives. This upgrade will further enhance the system's diagnostic reliability, ensuring quicker, more accurate results, ultimately empowering healthcare professionals with superior tools for early detection, personalized treatement, and improved patient outcomes.

CHAPTER 8

REFERENCE

1. Khan, Md Ashikur Rahman, Faria Afrin, Farida Siddiqi Prity, Ishtiaq Ahammad, Sharmin Fatema, Ratul Prosad, Mohammad Kamrul Hasan, and Main Uddin. "An effective approach for early liver disease prediction and sensitivity analysis." Iran Journal of Computer Science (2023): 1-19.
2. Amin, Ruhul, Rubia Yasmin, Sabba Ruhi, Md Habibur Rahman, and Md Shamim Reza. "Prediction of chronic liver disease patients using integrated projection based statistical feature extraction with deep learning algorithms." Informatics in Medicine Unlocked 36 (2023): 101155.
3. Shaheen, H., K. Ravikumar, N. Lakshmipathi Anantha, A. Uma Shankar Kumar, N. Jayapandian, and S. Kirubakaran. "An efficient classification of cirrhosis liver disease using hybrid convolutional neural network-capsule network." Biomedical Signal Processing and Control 80 (2023): 104152.
4. Nam, David, Julius Chapiro, Valerie Paradis, Tobias Paul Seraphin, and Jakob Nikolas Kather. "Artificial intelligence in liver diseases: Improving diagnostics, prognostics and response prediction." JHEP Reports 4, no. 4 (2022): 100443.
5. Srivastava, Aviral, V. Vineeth Kumar, T. R. Mahesh, and V. Vivek. "Automated Prediction of Liver Disease using Deep Learning (ML) Algorithms." In 2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), pp. 1-4. IEEE, 2022.
6. Cheng, Runtan, Lu Wang, Shenglong Le, Yifan Yang, Can Zhao, Xiangqi Zhang, Xin Yang et al. "A randomized controlled trial for response of microbiome network to exercise and diet intervention in patients with nonalcoholic fatty liver disease." Nature Communications 13, no. 1 (2022): 2555.

7. Paul R. Harper, A review and comparison of classification algorithms for medical decision making.
8. BUPA Liver Disorder Dataset. UCI repository deep learning databases.
9. Banu Priya, M., Laura Juliet, P.: Performance analysis of liver disease prediction using deep learning algorithms. IRJET. 5(1) (2018)
10. Jacob, J., Chakkalakal Mathew, J.: Diagnosis of liver disease using deep learning techniques. IRJET.(2018)
11. Hassoon, M.: Rule optimization of boosted C5.0 classification using genetic algorithm for liver disease prediction. IEEE (2017)

CHAPTER 9

CODING

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Liver Tumor Detection Using Machine Learning</title>
    <!-- Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css
" rel="stylesheet">    <style>        body {            background-color: #f0f9e8;
font-family: 'Arial', sans-serif;
        }
        .container                {
padding-top: 40px;
        }
        .form-group {
            background-color: #fff;            padding:
20px;            border-radius: 8px;            box-
shadow: 0 2px 5px rgba(0,0,0,0.15);
```

```
    }
    .image-preview {            height: 100%; /* Responsive width */
object-fit: cover; /* Cover to maintain aspect ratio */            border:
1px solid #ccc; /* Adds a light border */            border-radius: 4px; /*
Slightly rounded corners for aesthetics */
    }
    .btn-success            {
width: 100%;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-md-8">
        <form id="imageForm" class="form-group" action="" method="post"
enctype="multipart/form-data">
          {%csrf_token%}
          <h2 class="text-center mb-4">Liver Tumor Detection</h2>
          <input      type="file"   name="myimg"      id="imageInput"
accept="image/*" class="form-control mb-3">
```

```
            <button type="submit" class="btn btn-success mb-3">Upload Image</button>
            <p>Detected Disease: {{disease}}</p>
            <img id="displayImage" src="{{img_out}}" class="image-preview img-fluid" >
        </form>
      </div>
    </div>
  </div>

  <script>
document.getElementById('imageInput').addEventListener('change', function(event) {
        const file = event.target.files[0];
        const reader = new FileReader();
        reader.onload = function(e) {
document.getElementById('displayImage').src = e.target.result;
        };
        reader.readAsDataURL(file);
    });
    function submitForm() {
        alert('Image upload functionality is not implemented yet.');
        // Normally, here you would handle the form submission to the server.
```

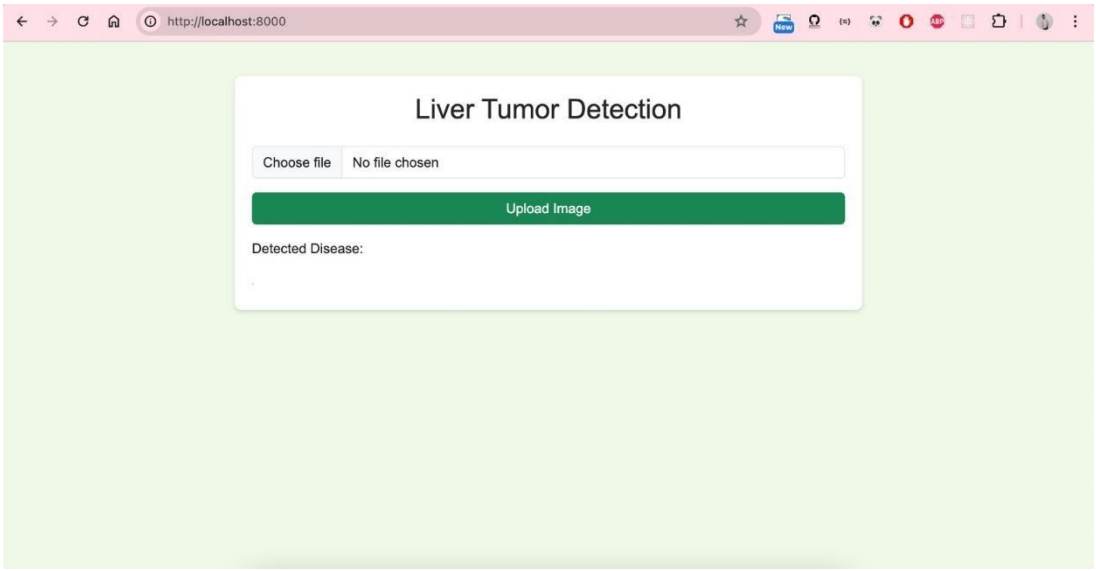
```
        }
    </script>
</body>
</html> Python
code:
from django.shortcuts import
render from ultralytics import
YOLO from PIL import Image
import json import cv2 import
requests import tkinter as tk from
tkinter import filedialog
from time import sleep from myapp.models
import Uploads from django.core.files.base
import ContentFile model =
YOLO("static/best1.pt") # Create your views
here. def detect_objects_on_image(buf):
results = model.predict(buf)     result =
results[0]     output = []     for box in
result.boxes:         x1, y1, x2, y2 = [
round(x) for x in box.xyxy[0].tolist()
        ]
```

```
        class_id = box.cls[0].item()         prob =
round(box.conf[0].item(), 2)         output.append([
x1, y1, x2, y2, result.names[class_id], prob
        ])
    return output

def    index(request):                  if
(request.method == "GET"):
        return   render(request,   'index.html')
if (request.method == "POST"):
        try:
            img = request.FILES.get('myimg')             uploads
= Uploads()             uploads.in_image = img
uploads.save()             print(str(uploads.in_image))
img = cv2.imread("media/"+str(uploads.in_image))
rr = detect_objects_on_image(img)             name = ""
for bb in rr:                x1 = bb[0]                y1 = bb[1]
x2 = bb[2]                y2 = bb[3]                cls = bb[4]
conf = bb[5]
                if (cls == '1'):
```

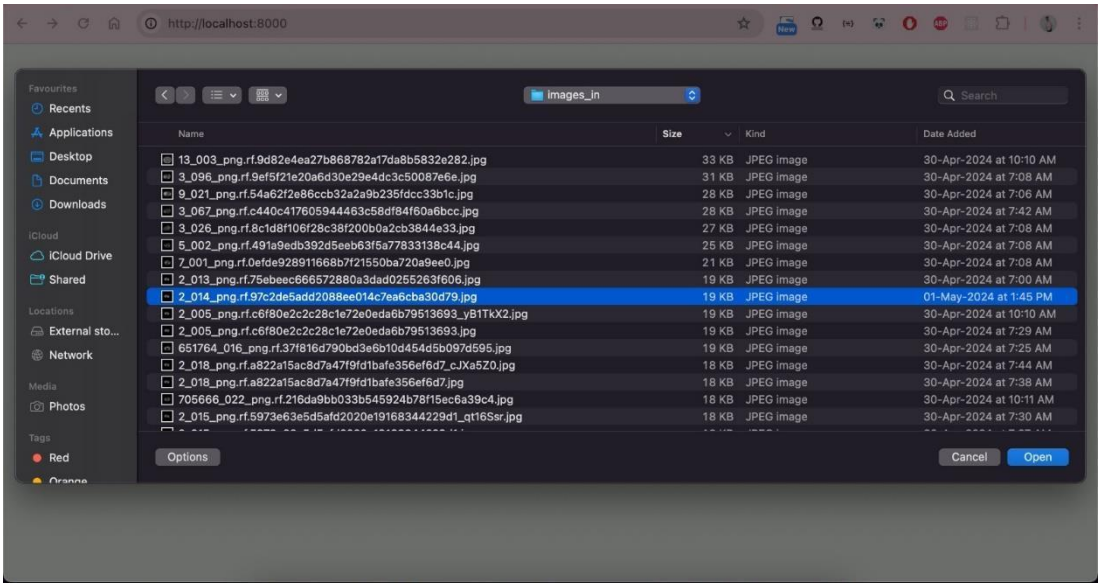
```
                name  =  "Stage  1"
if (cls == '2'):
                name  =  "Stage  2"
if (cls == '3'):
                name = "Stage 2"              cv2.rectangle(img, (x1,
y1), (x2, y2), (255, 0, 255), 2)              cv2.putText(img,
f'{name.upper()} {int(conf*100)}%',
                        (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0,
255), 2)
          print(rr)          data = {}          data['img_out']
= 'media/'+str(uploads.out_image)          data['disease']
= name


      except Exception as e:
          data = {}          print(e)        return
render(request, 'index.html', data)
```

CHAPTER 10

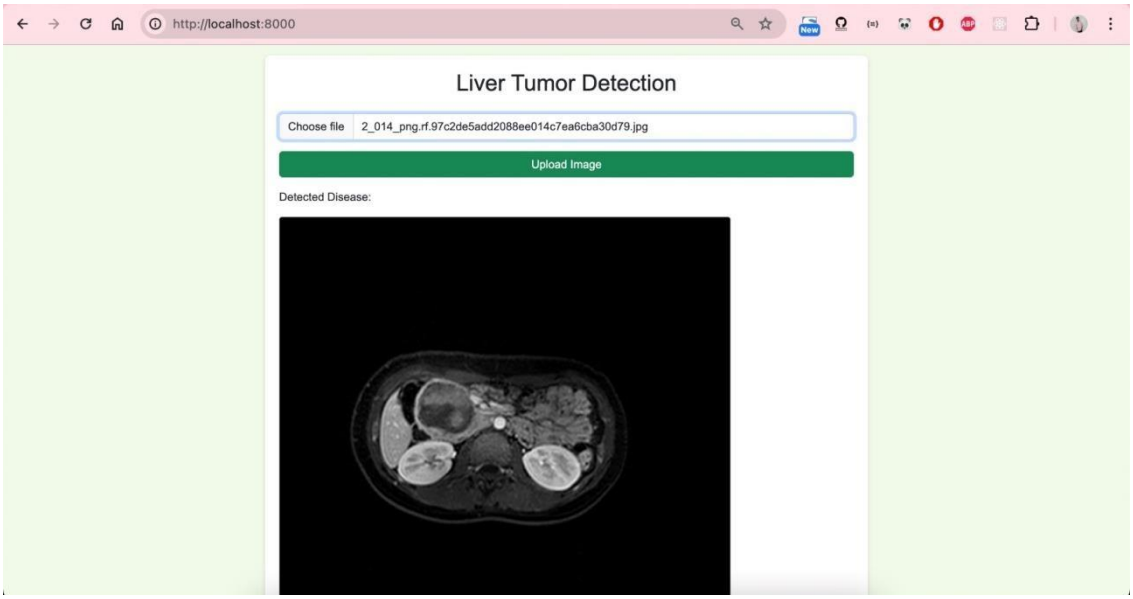
SCREENSHOT



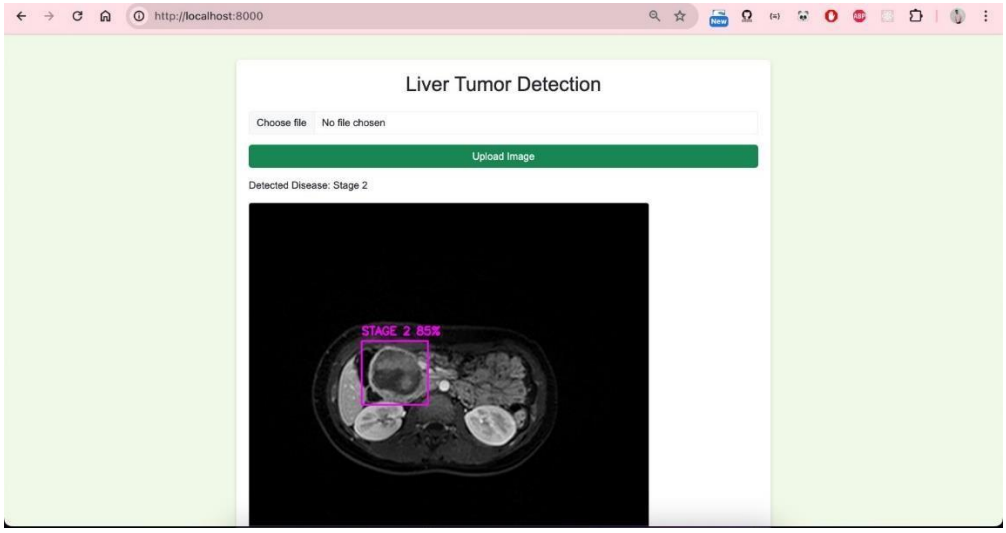
This the main web view to select the image to detect the liver tumor in the given CT image.



Once you select the choose image, it will give you the pormpt to select the image.



After you select the image, the image will show in the webview.



After upload the image, the image will be processed using the yolo, and the output image will return back to the web view with the stage of cancer and annotated positon.