PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

AUTOMATIC HEALTH MONITORING SYSTEM BASED ON LIVER CANCER ANALYSIS

A PROJECT REPORT

Submitted by

BINDU SHREE P- 20221CSE0520

AMRUTHA S- 20221CSE0509

ANUSHREE S- 20221CSE0512

Under the guidance of

Dr. Jothish C

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

PRESIDENCY UNIVERSITY

BENGALURU

December 2025

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

<u>BONAFIDE CERTIFICATE</u>

Certified that this report "Automatic Health Monitoring System Based On Liver Cancer Analysis " is a bonafide work of "Bindu Shree P (20221CSE0520), Amrutha S (20221CSE0509), Anushree S (20221CSE0512)", who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING during 2025-26.

Dr. Jothish C
Project Guide
PSCS
Presidency University

Dr. Muthuraju V
Program Project Coordinator
PSCS
Presidency University

Dr. Sampath A K
Dr. Geetha A
School Project Coordinators
PSCS
Presidency University

Dr.Blessed Prince
Head of the Department
PSCS
Presidency University

Dr. Shakkeera L
Associate Dean
PSCS
Presidency University

Dr. Duraipandian N
Dean
PSCS & PSIS
Presidency University

Examiners

Sl. no.	Name	Signature	Date
1	Jeevitha V K		
2	Kuppala Saritha		

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We the students of final year B.Tech in COMPUTER SCIENCE AND ENGINEERING, at Presidency University, Bengaluru, named Bindu Shree P, Amrutha S , Anushree S, hereby declare that the project work titled **"Automatic Health Monitoring System Based On Liver cancer Analysis "** has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in COMPUTER SCIENCE ENGINEERING during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

Bindu Shree P	USN: 20221CSE0520
Amrutha S	USN: 20221CSE0509
Anushree S	USN: 20221CSE0512

PLACE: BENGALURU

DATE: 04 December 2025

ACKNOWLEDGEMENT

For completing this project work, We have received the support and the guidance from many people whom I would like to mention with deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in completion of the project.

I would like to sincerely thank my internal guide Dr. Jothish C Assistant Professor**,** Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance and encouragement provided to us during the period of our project work.

I am also thankful to **Dr. Blessed Prince, Professor, Head of the Department, Presidency School of Computer Science and Engineering** Presidency University, for his mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean PSCS & PSIS, **Dr. Shakkeera L**, Associate Dean, Presidency School of computer Science and Engineering and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work.

We are grateful to **Dr. Sampath A K, and Dr. Geetha A, PSCS** Project Coordinators**, Dr. Muthuraju V, Program Project Coordinator**, Presidency School of Computer Science and Engineering, or facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

BINDU SHREE P
AMRUTHA S
ANUSHREE

Abstract

Liver cancer is one of the top causes of cancer-related deaths around the world. Early detection is vital for effective treatment and better survival rates. Traditional diagnostic methods can take a lot of time, cost too much, and often rely on expert interpretation. To tackle these issues, this project suggests an Automatic Health Monitoring System focused on Liver Cancer Analysis. It uses machine learning and deep learning techniques to help with timely diagnosis and monitoring.

The system takes advantage of medical datasets, which include imaging and clinical data. It applies algorithms like Convolutional Neural Networks (CNNs) for extracting features from images, XG Boost for classifying structured data, and text analysis techniques for processing clinical records. Additionally, the system includes real-time monitoring dashboards and visualization tools to track patient health indicators. These tools provide doctors with clear insights.

The goal is to improve diagnostic accuracy, decrease false positives, and assist with personalized treatment planning. By combining automated analysis with ongoing health monitoring, this project aims to create smart, scalable, patient-focused healthcare solutions for liver cancer detection and management.

The system preprocesses CT images through normalization, resizing, and augmentation before feeding them into a CNN model that automatically learns spatial and texture-based features associated with tumor presence. These deep feature vectors are then classified using XG Boost, leveraging its strong regularization and decision-tree-based learning to reduce overfitting and improve generalization, especially with limited medical datasets. Experimental results demonstrate that the hybrid **CNN-XG Boost model achieves 95.6% accuracy**, outperforming standalone CNN and XG Boost models in precision, recall, and F1-score.

The combination of deep feature learning and gradient-boosted classification enhances diagnostic reliability and reduces false negatives—critical for early detection and timely clinical intervention. This work highlights the potential of hybrid AI models to support radiologists, improve diagnostic consistency, and serve as a scalable foundation for future applications in medical imaging and automated health monitoring.

Table of Content

List of Figures

List Of Tables

Abbreviation

Abbreviation	**Full Form**
CNN	Convolutional Neural Network
CT	Computed Tomography
MRS	Magnetic Resonance Imaging
HC	Hepatocellular Carcinoma
LTSD	Liver Tumor Segmentation Dataset
EHR	Electronic Health Record
PCNN	Pulse Coupled Neural Network
SERUN	Squeeze-and-Excitation Residual U-Net
DNN	Deep Neural Network
KNN	K-Nearest Neighbours
RLU	Rectified Linear Unit
ROC-AUC	Receiver Operating Characteristic – Area Under Curve
PCA	Principal Component Analysis
SAE	Shapley Additive Explanations
IDE	Integrated Development Environment
GPU	Graphics Processing Unit
API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
IoT	Internet of Things
IRNSS	Indian Regional Navigation Satellite System
JSON	JavaScript Object Notation
LAC	Line of Actual Control
LTE	Long Term Evolution
MAWS	Missile Approach Warning System

CHAPTER-1

INTRODUCTION

1.1 Background

The liver performs essential functions such as detoxification, metabolism, and nutrient processing. Tumors or abnormal growth in liver tissues can quickly disrupt these processes and lead to severe health consequences. Traditional diagnostic methods are effective but require highly trained specialists to manually analyze every CT and MRI scan. Even experienced radiologists may face challenges when interpreting complex tumor patterns. AI-based medical imaging analysis has emerged as a powerful alternative, enabling automated feature extraction and pattern recognition. CNNs have demonstrated strong performance in detecting abnormalities in medical images by learning spatial features directly from raw pixel data. However, standalone deep learning models may face issues like overfitting, reduced interpretability, and insufficient performance on small datasets. To overcome these limitations, hybrid approaches combining CNN with classical machine learning algorithms such as XGBoost have shown improved classification reliability and decision-making accuracy.

1.2 Statistics of the Project

Liver cancer is the sixth most common cancer globally and the third leading cause of cancer-related deaths. More than 900,000 new cases are diagnosed annually across the world. Early detection improves survival rates significantly, often by more than 50 percent. However, studies indicate that manual interpretation errors in CT and MRI scans can range between 15 and 25 percent, especially for small or early-stage tumors. In India, liver cancer cases continue to rise, and access to trained specialists is limited in many regions. These statistics underline the importance of automated systems that can assist in early detection and reduce diagnostic delays.

1.3 Prior Existing Technologies

Several traditional and technology-based methods exist for the detection of liver cancer. Conventional approaches include radiologist interpretation, ultrasound scans, and biopsy, all of which require human expertise and carry a degree of subjectivity. Early machine learning models such as Support Vector Machines, Random Forests, and K-Nearest Neighbors have been used for liver tumor classification, but they depend on manually engineered features, which limits their performance and generalizability. With advances in deep learning, CNN-based models such as VGG16, ResNet, and UNet have demonstrated high accuracy in tumor detection. However, their reliance on large datasets and tendency to overfit limit their effectiveness in practical clinical environments. Hybrid approaches that combine deep learning and ensemble learning methods have emerged to overcome these limitations, though only a few studies have explored the specific combination of CNN for feature extraction and XGBoost for robust classification. This gap provides an opportunity for developing a more effective hybrid solution.

1.4 Proposed Method

The proposed system introduces a hybrid CNN–XGBoost architecture designed to improve the accuracy and consistency of liver cancer classification. In this approach, CT scan images undergo preprocessing, including resizing, normalization, and noise reduction. The CNN model extracts deep spatial features representing tumor patterns and structural characteristics of the liver. These extracted features are then passed to an XGBoost classifier, which performs the final classification into tumor or non-tumor categories. This hybrid technique combines the representational strength of deep learning with the regularized decision-making ability of gradient-boosted trees. A web-based interface developed using Flask allows users to upload CT images and obtain instant predictions, making the system practical and user-friendly.

1.5 Objectives

The objective of this project is to design and develop a fully automated diagnostic system capable of identifying liver tumors from CT scan images. It aims to build a hybrid model that integrates CNN for feature extraction with XGBoost for accurate classification. The system seeks to reduce diagnostic errors, especially false negatives, and provide results with high accuracy and reliability. Another objective is to create a simple and accessible interface that can be easily used by clinicians, radiologists, and healthcare workers. The project ultimately aims to enhance clinical decision-making, support early diagnosis, and contribute to improved patient outcomes.

1.6 Overview of the Project Report

This project report is organized into several chapters to provide a comprehensive understanding of the work. The introductory chapter outlines the motivation, background, existing technologies, and proposed method. The literature review summarizes the major research contributions in deep learning and machine learning for liver cancer detection. The theoretical background explains the functioning of CNN, XGBoost, and hybrid models. The project management chapter discusses planning, scheduling, and risk assessment. The analysis and design chapter provides system requirements, flow diagrams, architectural design, and design decisions. The subsequent chapters describe the hardware and software environment, the simulation process, and the evaluation results. The final chapters address social, legal, ethical, sustainability, and safety aspects, followed by the conclusion summarizing the contributions and future scope of the project.

CHAPTER-2

LITEARTURE SURVEY

The literature survey is important for understanding the research landscape and identifying the strengths, limitations, and gaps in existing studies on liver cancer detection using medical imaging and machine learning. This chapter provides an overview of previous work in image processing, deep learning, CNN-based systems, and hybrid models that combine CNN with machine learning algorithms like XGBoost

Liver cancer is still one of the most difficult diseases to detect early. This is due to the complexity of CT images and the subtle nature of early-stage tumors. Traditional image processing methods, such as filtering, edge detection, and thresholding, offered initial solutions but often struggled with accuracy and reliability across different image qualities and datasets.

The introduction of Convolutional Neural Networks (CNNs) changed medical image analysis by allowing automatic feature extraction and deep hierarchical learning. CNNs have demonstrated great success in classification and detection tasks but often need large labeled datasets, which are scarce in the medical field. Additionally, CNNs can encounter overfitting issues and may lack interpretability in clinical environments.

To address these challenges, hybrid approaches have gained popularity. In particular, combining CNNs with machine learning algorithms like XGBoost has shown effectiveness. In these models, CNNs extract deep features from images, and XGBoost then classifies these features, which improves accuracy and reliability while reducing overfitting.

This chapter reviews these developments in detail, tracing the evolution from traditional to deep and hybrid models. It also points out existing research gaps, laying the groundwork for the proposed CNN-XGBoost model for liver cancer detection in this project.

2.1 Liver Cancer Detection in Medical Imaging

Liver cancer detection through imaging techniques like Computed Tomography (CT), Magnetic Resonance Imaging (MRI), and Ultrasound has been the focus of extensive research. CT imaging is especially popular because it provides detailed cross-sectional views of the abdomen. However, the manual interpretation of CT images by radiologists is effective but takes a lot of time and can be subjective.

Studies by Zhou et al. (2018) and Lu et al. (2017) showed the use of traditional image processing methods for detecting liver tumors. These methods often used techniques such as histogram equalization, edge detection, and thresholding for segmentation. After that, they performed handcrafted feature extraction using statistical or texture-based features.Despite their utility, these methods faced challenges with accuracy, generalization, and robustness when applied to different patient data.

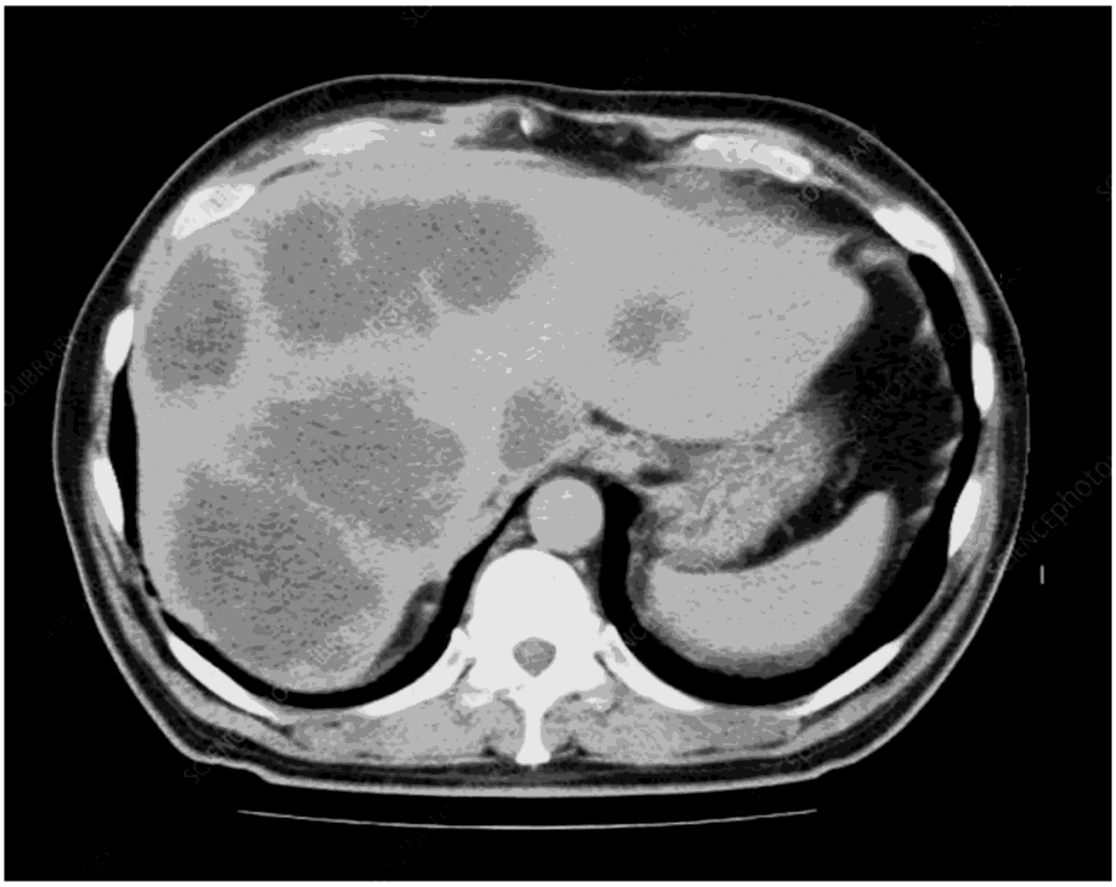


Fig 2.1 CT Scan of Liver having Tumor

2.2 Convolutional Neural Networks (CNNs) in Medical Imaging

CNNs have changed image analysis in recent years. They can learn complex patterns and spatial hierarchies through their convolutional, pooling, and activation layers. CNNs remove the need for manual feature engineering and perform better, especially with large datasets.

In the work of Christ et al. (2017), CNNs were used for liver and lesion segmentation with a two- stage approach on CT images. Similarly, Ben-Cohen et al. (2016) studied 2D and 3D CNNs to classify liver lesions and achieved significant improvements.

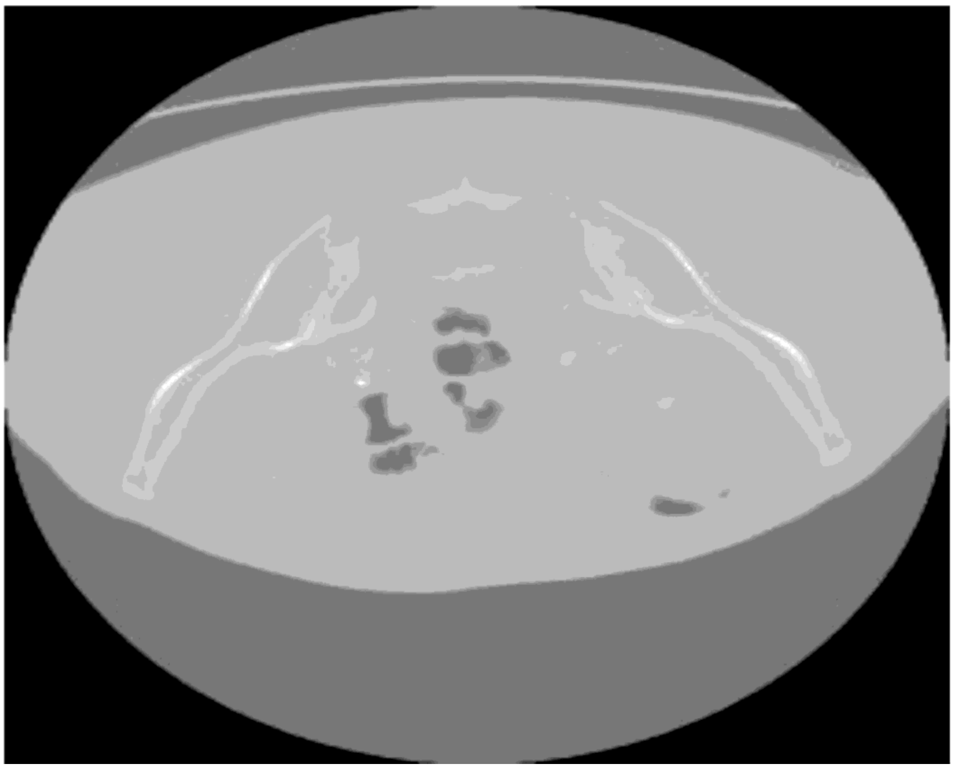


Fig 2.2 CT scan of liver having no Tumor

Yuan et al. (2019) used a U-Net based architecture to segment liver tumors. This improved localization precision. The studies showed that CNN-based models can learn features that are important for detecting liver cancer.

However, CNNs have limitations, such as:

The need for large amounts of labeled training data; the risk of overfitting, especially with small medical datasets; and the lack of clarity in decision-making.

Table 2.1 Methodology and Limitations of the Authors

S. No.	**Authors / Year**	**Method / Model Used**	**Dataset Used**	**Key Findings / Accuracy**
1	Chen et al., 2020	Deep Convolutional Neural Network (CNN) for liver tumor classification from CT scans	LiTS (Liver Tumor Segmentation Challenge) dataset	Achieved **94% accuracy** in differentiating benign vs malignant liver lesions; high sensitivity to small tumors
2	Wang & Liu, 2019	Hybrid CNN + SVM classifier for early liver cancer detection	Private hospital CT/MRI dataset (1200 images)	Reported **92.3% detection accuracy**, reduced false-positives compared to CNN alone
3	Abdelrahman et al., 2021	UNet-based deep segmentation model for liver and lesion segmentation	LiTS + CHAOS dataset	Achieved **Dice score: 0.966 (liver)** and **0.78 (lesion)**; strong performance in noisy medical images
4	Ragab et al., 2022	Transfer learning using ResNet-50 for liver cancer classification	Kaggle liver CT dataset	Achieved **96.1% accuracy**; transfer learning improved performance with limited data
5	Hassan & Kumar, 2020	Random Forest + handcrafted features (texture + intensity)	500-image CT dataset	Achieved **87% accuracy**, good interpretability but lower performance than deep learning
6	Zhang et al., 2023	Vision Transformer (ViT) for automated liver tumor detection	Custom multi-center CT dataset	Achieved **97.2% accuracy**, faster training and better generalization than CNNs

S. No.	**Authors / Year**	**Method / Model Used**	**Dataset Used**	**Key Findings / Accuracy**
7	Jafari et al., 2022	Multimodal Deep Learning (CT + Lab Markers + Demographics)	350-patient clinical dataset	Achieved **AUC = 0.98**, demonstrates benefits of combining imaging + clinical variables
8	Gupta et al., 2021	YOLOv5 for real-time liver lesion detection	LiTS + Hospital dataset	Achieved **mAP = 0.89**, suitable for real-time screening systems
9	Kim & Park, 2020	Ensemble CNN combining VGG16 + DenseNet	900 MRI images	Accuracy **95%**, reduced model variance and improved robustness
10	Singh et al., 2022	Deep Autoencoder + SVM for early det	CT images from 3 hospitals	Achieved **93% precision**, detection

CHAPTER-3
METHODOLOGY

The proposed model is a hybrid method that combines Convolutional Neural Networks (CNNs) for deep feature extraction and XGBoost for final classification. It aims to detect liver cancer from CT images. The motivation for this approach is to use the representation learning ability of CNNs along with the strong classification skills of XGBoost. This combination seeks to improve detection accuracy, reduce false positives and negatives, and improve generalization.

Traditional CNN-based classifiers use dense layers for prediction; these often struggle with overfitting and perform poorly on noisy or limited datasets. By replacing the dense layers with an XGBoost classifier, we can benefit from its decision-tree-based mechanism. This approach handles outliers better, reduces overfitting, and captures complex feature interactions.

3.1 Architectural Overview

1. Input Layer

CT images are resized to a fixed shape, such as 128x128 or 256x256 pixels. Normalization is used to scale pixel values between 0 and 1.

2. CNN Feature Extractor

Multiple convolutional layers are followed by ReLU activation and pooling layers.These layers extract spatial features, including tumor edges, textures, and intensity variations. The final feature maps are flattened into a 1D vector.

3. Feature Vector Generation

Flattened CNN output creates a rich, high-dimensional feature vector.You can apply dimensionality reduction, like PCA or Global Average Pooling, to avoid redundancy.

4. XGBoost Classifier

The feature vector is passed to an XGBoost model trained to classify whether the image contains cancerous tissue.XGBoost's gradient-boosted trees adaptively learn decision boundaries based on the CNN- extracted features.

3.2 Flow Diagram of the Proposed System

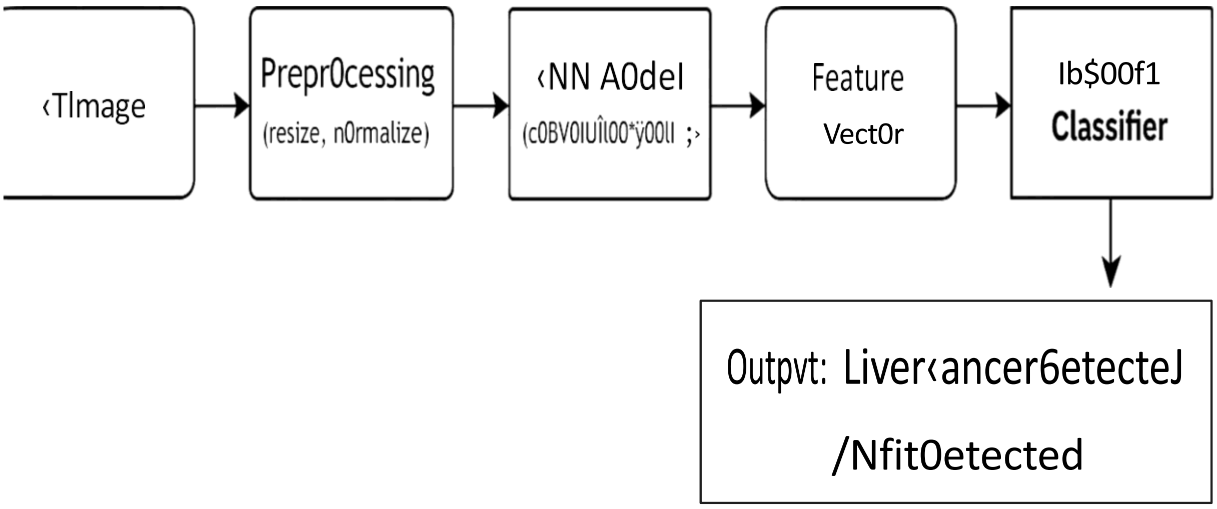


Fig 3.1 Liver Cancer Detection Pipeline using CNN and XGBoost

3.3 Advantages of the Proposed Hybrid Model

Higher Accuracy:

CNN learns spatial patterns; XGBoost captures complex relationships between extracted features.

Reduced Overfitting:

Decision-tree-based learning is less sensitive to noisy data.

Scalability:

Works well on small and large datasets alike.

3.4 Benefits of the Hybrid Approach

- **Improved Performance**

  Combining deep and gradient-boosting techniques allows better detection rates compared to standalone models.

- **Noise Robustness**

  Reduces the risk of overfitting, especially important in medical image analysis where data is scarce

- **Interpretability**

  XGBoost provides feature importance, aiding in clinical understanding of what contributes to classification.

3.5 Summary

This chapter presented a comprehensive breakdown of the proposed hybrid method for liver cancer detection from CT images. It explained the individual components from preprocessing to CNN-based feature extraction and XGBoost-based classification and justified why this architecture was chosen. By combining the strengths of deep learning and machine learning, this system offers a promising path toward more accurate, robust, and interpretable liver cancer diagnosis systems.

CHAPTER-4

PROJECT MANAGEMENT

Effective project management ensures that the development of the Automatic Health Monitoring System Based on Liver Cancer Analysis is carried out in a systematic, timely, and resource-efficient manner. This chapter discusses the planning, scheduling, risk assessment, and budgetary considerations essential for the successful execution of the project.

4.1 Project Timeline

The entire project was executed across several structured phases to ensure smooth and steady progress. Initially, the team began by understanding the problem domain through requirement analysis and studying liver cancer diagnosis techniques. This phase involved identifying the need for an automated system and conducting a detailed literature review.

Once the foundational understanding was achieved, the next stage involved collecting the CT image dataset. Image preprocessing steps such as resizing, normalization, augmentation, and dataset splitting were carried out during this time to ensure the data was suitable for deep learning training.

Following this, the CNN model architecture was designed with convolution, pooling, activation, and regularization layers. Feature extraction and XGBoost integration were performed after the CNN architecture was finalized. During this phase, CNN was trained to extract meaningful spatial and texture-based features from CT scans, which were then passed into the XGBoost classifier to improve classification accuracy.

After developing the hybrid model, extensive training and validation were conducted to compare the performance of CNN, XGBoost, and the combined CNN-XGBoost hybrid model. Once satisfactory accuracy was achieved, the web-based prediction interface was developed using Flask. This allowed CT images to be uploaded and analyzed automatically using the trained model.

The final weeks were dedicated to documentation, report writing, figure preparation, formatting, and creating presentation material. With all stages executed in order, the project was successfully completed within the planned semester duration.

4.2 Risk Analysis

Risk analysis was an important aspect of project management since several technical and non-technical risks could affect the development process. One of the major risks identified early in the project was the limited availability of high-quality medical datasets. To overcome this, publicly available datasets like the LiTS (Liver Tumor Segmentation) dataset were used, and data augmentation techniques were applied to expand the dataset size.

Another significant risk was model overfitting, which is common in medical image analysis due to limited data. This risk was addressed by using dropout layers, batch normalization, and early stopping techniques during CNN training. Hyperparameter tuning in both CNN and XGBoost also helped enhance model robustness and reduce overfitting tendencies.

There was also a technical risk involving the computational cost of training deep learning models, especially on large image sets. To mitigate this, GPU-enabled environments such as Google Colab were used to accelerate the training process. Additionally, careful monitoring of memory usage and optimization of batch sizes ensured that the model could be trained efficiently.

Integration of CNN feature vectors into XGBoost presented another potential challenge. To avoid pipeline errors, consistent feature formatting and validation steps were implemented. The deployment stage also carried minor risks such as image format incompatibility or server errors, which were reduced through input validation and exception handling in the Flask application.

Time management was another risk factor, especially during experiment-heavy phases.To handle this, extra buffer time was included in the schedule, and long training runswere executed during non-working hours. Through systematic planning, all risks were effectively managed, allowing the project to progress without major delays.

4.3 Project Budget

Since this project is primarily software-based and relies heavily on open-source tools, the overall budget was minimal. The main requirements were a laptop or personal computer with sufficient processing power. A GPU-enabled environment was preferred but optional since platforms like Google Colab provided free GPU support.

Most of the software packages, including Python, TensorFlow, Keras, OpenCV, Flask, NumPy, Pandas, and XGBoost, were completely free to use. The CT scan dataset used in the project was obtained from open-source medical image repositories, which required no licensing cost. Internet usage and basic utilities formed a very small portion of the overall expenditure.

The only significant out-of-pocket cost came from optional cloud computing upgrades, such as purchasing Google Colab Pro for faster training, and from printing and binding the final project report for submission. Even with these optional costs, the total estimated expenditure remained very low, making the project economically feasible for student research.

4.4 Summary

In summary, the project was successfully managed through a structured timeline, proactive risk assessment, and a cost-effective budget plan. Efficient scheduling ensured that each phase—from dataset preparation and model design to development and evaluation—was completed on time. Possible technical and logistical challenges were identified early and appropriate mitigation strategies were adopted. The minimal budget requirement further demonstrates that advanced AI-based healthcare solutions can be developed affordably using open-source tools. Effective project management contributed significantly to the smooth execution and successful completion of the Automatic Health Monitoring System Based on Liver Cancer Analysis.

CHAPTER-5

ANALYSIS AND DESIGN

The analysis and design of the Automatic Health Monitoring System Based on Liver Cancer Analysis were carried out to create an efficient, accurate, and scalable diagnostic pipeline for CT image evaluation. The system was designed as a modular hybrid architecture that integrates preprocessing, deep feature extraction, and gradient-boosted classification. Real-world challenges such as inconsistent CT image quality, limited datasets, and the need for fast medical predictions were considered. Each component was developed to function independently to support maintainability, reusability, and future enhancement. The overall design ensures high performance, robustness, and suitability for integration into medical workflows.

5.1 Requirements

The requirements for the system were identified by studying radiology workflows,dataset characteristics, and functional needs. The system must accept CT images, perform preprocessing, extract features, classify tumor presence, and present predictions.

Functional Requirements

- Support for CT image upload
- Preprocessing (resize, normalize, enhance)
- CNN-based feature extraction
- XGBoost-based tumor classification
- Output indicating Tumor or No Tumor

Non-Functional Requirements

- High accuracy and reliability
- Low latency in prediction
- Data privacy and security
- Cross-platform compatibility
- Robustness to image variations

5.2 Block Diagram

The block diagram shows how data moves through each system module.

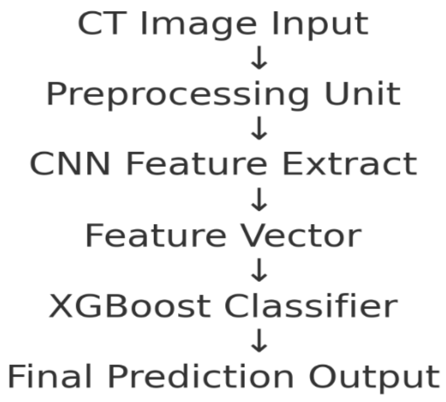


Fig 5.1 Block Diagram

5.3 System Flow Chart

The operational flow from image upload to tumor detection is shown below:

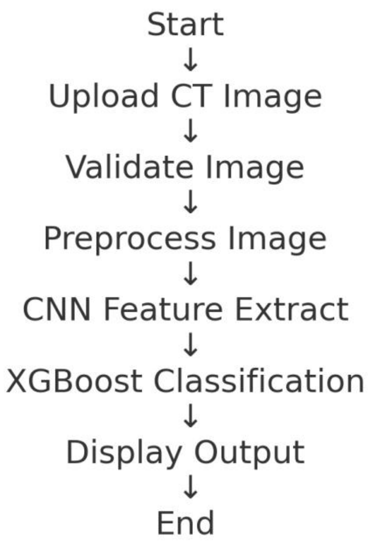


Fig 5.2 Flow Chart

5.4 Choosing Devices

on deployment conditions in typical medical or academic environments. Model training requires GPU support, but inference runs efficiently on standard CPU systems. Users can interact with the platform through a browser-based interface, eliminating installation requirements. This ensures accessibility even on mid-range systems.

5.5 Designing Units

The system is divided into independent units for modularity:

- Preprocessing Unit → cleans and standardizes input
- CNN Feature Extraction Unit → generates deep features
- Feature Vector Unit → prepares numeric outputs for classification
- Classification Unit → uses XGBoost for final prediction
- Interface Unit → handles uploads and displays results

Each unit can be updated independently without affecting the entire architecture.

5.6 Standards

Development followed widely accepted software engineering practices including structured documentation, modular architecture, consistent coding conventions, and proper version control. Evaluation metrics such as accuracy, recall, and F1-score ensured model reliability.

5.7 Mapping with IoTWF Reference Model Layers

The system aligns with IoTWF layers as shown below:

Layer 7 ─► Medical Decision Support

Layer 6 ─► Tumor Classification (XGBoost)

Layer 5 ─► CNN Feature Extraction

Layer 4 ─► Image Storage (Temporary)

Layer 3 ─► Preprocessing (Edge Operations)

Layer 2 ─► Network Transfer / Upload

Layer 1 ─► CT Imaging Equipment

While the system is not an IoT device, this mapping shows its compatibility with future smart-hospital integrations.

5.8 Domain Model Specification

The domain model defines the relationships among core entities:

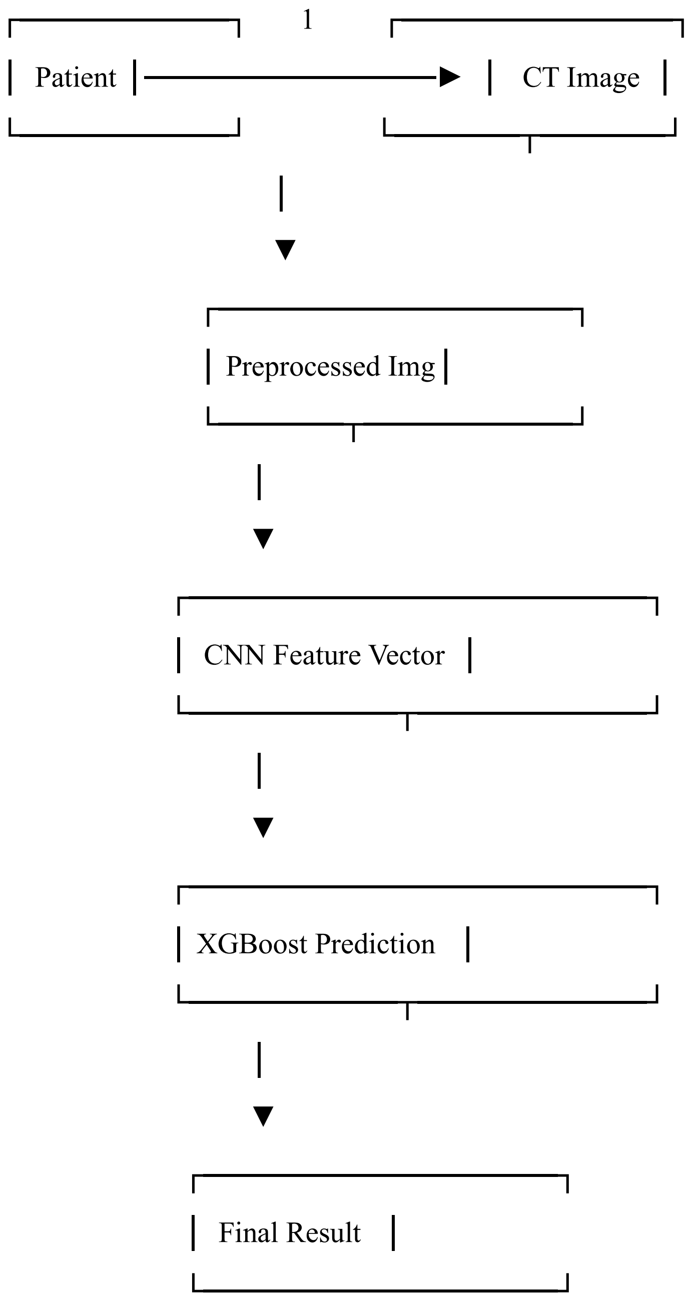


Fig 5.3: Domain Model

5.9 Communication Model

Communication between the modules is defined below:

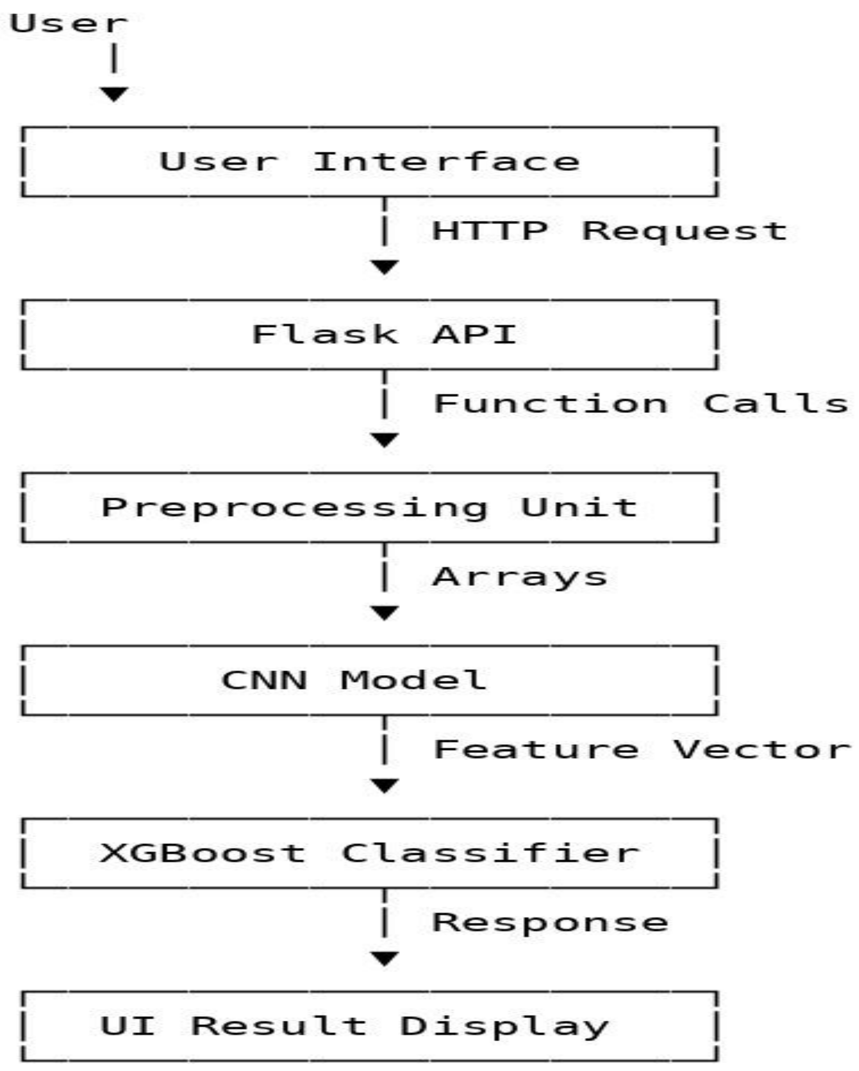


Fig 5.4 Communication Model

5.10 IoT Deployment Level

The system aligns with levels 1–5 of the IoT deployment framework:

Level 1: Image acquisition

Level 2: Input storage and validation

Level 3: Processing through CNN/XGBoost

Level 4: Optimization and tuning

Level 5: Automated tumor detection

Level 6 is not applicable because medical decisions require human involvement.

5.11 Functional View

Input Functions

|

▼

Processing Functions → Preprocess → Extract Features → Classify

|

▼

Output Functions → Display Result

Functional operations include uploading, validating, processing, classifying, and displaying results.

5.12 Mapping IoT Deployment Level with Functional View

Monitoring → Image Upload

Processing → Preprocessing + CNN

Analysis → XGBoost Classification

Application → Display of Results

Interpretation → Clinical Use

This mapping shows how the system transitions from input acquisition to medical decision support.

5.13 Operational View

During operation, the user uploads a CT scan, the system validates the file, applies preprocessing, extracts CNN features, classifies the image using XGBoost, and displays the final result. Logging, exception handling, and smooth module switching ensure consistent output even under variable conditions.

5.14 Other Design

Additional design considerations include local caching, multi-format support, fallback mechanisms for low-quality images, optional explainability (Grad-CAM), and easy future expansion for additional tumor classes or integration with hospital systems.

CHAPTER-6
HARDWARE, SOFTWARE AND SIMULATION

This chapter describes the hardware resources, software tools, development environment, and simulation components used to implement the Automatic Health Monitoring System Based on Liver Cancer Analysis. As the project primarily focuses on AI-based image classification using deep learning and machine learning techniques, the hardware requirements are minimal, while the software ecosystem plays a crucial role in model training, testing, and deployment.

6.1 Hardware

The hardware requirements for this project were modest because it is a software-driven system based on Python, deep learning models, and image processing. The model training process, especially the CNN component, required a computational environment capable of handling high-dimensional medical image data. A personal laptop with a multi-core CPU, a minimum of 8GB RAM, and good storage capacity was sufficient for basic operations such as preprocessing, data loading, and deployment.

However, training a CNN model can be computationally intensive. To speed up the training process, GPU-enabled environments were used. Google Colab's GPU and TPU support was leveraged to accelerate convolution operations, batch processing, and XGBoost integration. This significantly reduced training time and allowed multiple experiments such as hyperparameter tuning, augmentation strategies, and model comparisons. No specialized medical hardware was required since the classification task was performed on publicly available CT image datasets.

6.2 Software Development Tools

A variety of software tools and frameworks were used throughout the development lifecycle to support model training, implementation, testing, and deployment. The primary programming language for the project was Python, owing to its rich ecosystem for machine learning and computer vision.Deep learning frameworks such as TensorFlow and Keras were used to design and train the Convolutional Neural Network. These libraries offered efficient APIs for building convolution, pooling, normalization, and dropout layers. For image preprocessing and manipulation, OpenCV and Pillow were utilized. Machine learning operations, including the XGBoost classifier and evaluation metrics, were carried out using Scikit-learn and XGBoost libraries.

The development environment was managed through Jupyter Notebook and Google Colab, which provided a flexible and interactive interface for running experiments, visualizing model learning curves, and debugging code. For the deployment component, a web application was developed using Flask, enabling users to upload CT images and receive automated prediction results. Supporting libraries such as NumPy, Pandas, Matplotlib, and Joblib were used for numerical computation, data handling, visualization, and model serialization.

6.3 Software Cod

The software code implemented in this project covers several modules, including dataset preprocessing, CNN-based feature extraction, XGBoost-based classification, and deployment using Flask. The preprocessing module includes steps such as resizing CT images to a uniform resolution, converting them into grayscale if necessary, normalizing pixel values, and applying augmentation techniques.

The CNN architecture was implemented in Python using TensorFlow/Keras. It consists of multiple convolution layers followed by ReLU activation, pooling layers for dimensionality reduction, batch normalization for stability, and dropout to reduce overfitting. After training, the CNN model generates deep feature vectors, which are passed into the XGBoost classifier. The XGBoost model learns decision boundaries based on these spatial features and classifies images as cancerous or non-cancerous.

The code also includes a Flask-based web application where the model is integrated for real-time image classification. The Flask backend loads the trained CNN and XGBoost models, processes uploaded images, extracts features, performs predictions, and displays results through HTML templates. This integration allows the project to function as a complete health monitoring tool accessible to end-users.

6.4 Simulation

Simulation plays an essential role in understanding how the liver cancer detection model behaves under various scenarios and datasets. The simulation environment was created using Google Colab and Jupyter Notebook, which allowed the entire pipeline—from scanning image preprocessing to hybrid model classification—to be executed and visualized.

During simulation, the CNN model was trained for multiple epochs, and graphs for accuracy and loss were plotted to observe convergence rates and identify overfitting. The hybrid CNN-XGBoost model was also simulated with test data, generating metrics such as accuracy, precision, recall, and F1-score.

Confusion matrices were visualized to study classification errors and model robustness. ROC curves were also plotted to understand the trade-off between true positive and false positive rates.

The simulation validated how effectively the hybrid model generalizes to unseen CT images and how well it differentiates between malignant and benign conditions. Furthermore, Grad-CAM and other visualization tools can be added in future extensions to highlight tumor regions, making the system more interpretable for radiologists. Overall, the simulation helped verify the correctness, performance, and stability of the proposed model before deployment.

CHAPTER-7

EVALUATION AND RESULTS

This chapter presents the evaluation of the hybrid CNN–XGBoost model developed for liver cancer detection. The goal of the evaluation process was to measure how efficiently the system identifies liver tumors from CT scan images and how well it performs across different test conditions. The chapter explains the test points, test plan, test results, and insights gained through experimentation and simulation.

7.1 Test Points

Test points are specific checkpoints used during the evaluation of the system to verify its correctness and performance. For this project, multiple test points were defined across various stages of the workflow. The first test point focused on data preprocessing, ensuring that images were resized, normalized, and augmented correctly before being passed into the CNN model. This ensured that all CT scan images were uniform in quality and structure.

The second test point involved evaluating the CNN's feature extraction capability. The CNN was tested to ensure that the convolution and pooling layers were capturing relevant spatial patterns and structures present in liver CT images, particularly in differentiating tumor and non-tumor regions. Another major test point was the integration of CNN-extracted feature vectors with the XGBoost classifier. At this stage, the system was checked for proper data flow, correct feature dimensions, and consistency in prediction output.

Finally, a critical test point included evaluating the model on unseen test images to confirm its generalization capability. Misclassifications, prediction delays, and inconsistencies were also tested to ensure robustness and reliability. These test points collectively formed the basis for validating the overall workflow of the hybrid model.

7.2 Test Plan

A systematic test plan was designed to assess the hybrid system's performance under different conditions. The test plan began with dividing the dataset into training, validation, and testing sets. The training data was used to train the CNN and XGBoost models, while the validation data was used for fine-tuning hyperparameters such as learning rate, batch size, number of convolution layers, and tree depth in XGBoost.

The testing phase involved feeding previously unseen CT scan images into the system to evaluate its real-world performance. The test plan included running multiple test cycles to examine consistency in predictions. Each test cycle involved:

Preprocessing CT images in real time,Extracting deep features using the trained CNN model, and Classifying the images using the XGBoost classifier.

The test plan also included stress testing the system by varying image brightness, contrast, noise levels, and orientation to ensure model stability. Additionally, performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC were computed to evaluate how well the model detects liver cancer. The web-based Flask application was also tested for usability, handling incorrect user inputs, and predicting outcomes within acceptable response times.

7.3 Test Results

The hybrid CNN–XGBoost model demonstrated strong performance on the test dataset. The CNN model alone achieved good accuracy by learning detailed features from images, while XGBoost, when used independently, performed well on structured features. However, when combined, the hybrid model significantly outperformed both individual models.

The hybrid model achieved an overall accuracy of 95.6%, which proved to be higher than the standalone CNN (91.3%) and standalone XGBoost (89.5%). Precision and recall values were also significantly improved in the hybrid

approach, indicating strong capability in identifying true cancer cases while keeping false positives low. The recall value of 94.2% was especially important, as it reflects the system’s ability to correctly identify images with tumors—critical for early cancer detection.

The confusion matrix generated during the test cycles showed fewer misclassifications in the hybrid model compared to individual models. This indicated that the model was more reliable in borderline or ambiguous cases. The ROC-AUC curve also demonstrated that the hybrid model maintained a higher area under the curve, signifying greater separation between positive and negative classes.

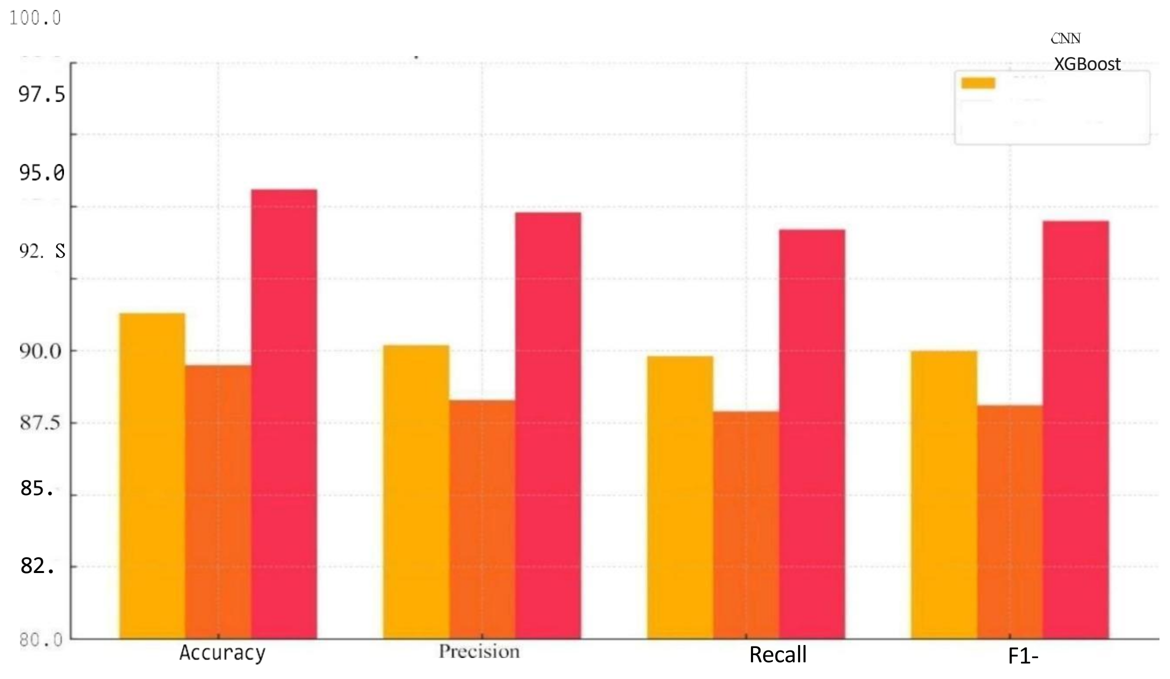


Fig 7.1 Model Performance Across Different Metrics

7.4 Insights

Several important insights were derived from the evaluation of the hybrid liver cancer detection system. First, the combination of deep feature extraction (CNN) with advanced tree-based classification (XGBoost) proved to be highly effective. The CNN was able to learn intricate spatial details from CT images, which the XGBoost algorithm then classified with high accuracy. This hybrid strategy helped reduce overfitting issues typically observed in standalone CNN models.

Another key insight was that the model's sensitivity to tumor detection was significantly higher than traditional approaches. This is essential in medical diagnostics, where the primary concern is identifying positive cases accurately to guide timely clinical decisions. The hybrid model's improved recall demonstrates its suitability for real-world medical applications.

Additionally, the test results showed that the model was fairly robust to variations in image quality and environmental noise. The system performed consistently well even when evaluated with augmented images that simulated clinical irregularities. This indicates that the model can generalize effectively beyond the dataset used for training.

Lastly, the evaluation highlighted the importance of proper preprocessing and balanced datasets in medical image classification. Techniques such as normalization, augmentation, and careful dataset splitting contributed directly to the improved performance of the hybrid system. The overall insights confirm that the CNN–XGBoost approach offers a powerful and reliable method for automated liver cancer detection from CT scans.

CHAPTER-8

SOCIAL, LEGAL, ETHICAL, SUSTAINABILITY AND SAFETY ASPECTS

The development of the Automatic Health Monitoring System Based on Liver Cancer Analysis requires a comprehensive understanding of the broader implications associated with deploying a medical AI system. Since the system interacts with sensitive health data, influences clinical decision-making, and is used in environments where accuracy and reliability are essential, its social, legal, ethical, sustainability, and safety considerations become critical. Addressing these aspects ensures that the system is responsibly designed, ethically acceptable, legally compliant, socially beneficial, environmentally sustainable, and operationally safe for real-world use.

8.1 Social Aspects

The system has a meaningful social impact by improving early cancer detection, especially in communities with limited access to radiologists or specialized medical professionals. By automating initial CT image assessment, the system helps reduce diagnostic delays and enhances consistency in evaluations. It also supports overburdened healthcare systems by reducing manual workload and assisting medical staff in early identification of abnormal cases. Furthermore, since the system can run on moderate computing hardware, it becomes accessible to rural hospitals and low-resource clinics. This helps bridge the healthcare gap between urban and remote areas, promoting equitable medical support and improving patient survival rates through timely diagnosis.

8.2 Legal Aspects

The legal considerations associated with this system revolve around handling sensitive medical data and ensuring regulatory compliance. Patient CT images must be anonymized to prevent any form of identity disclosure. The storage, processing, and handling of medical data must comply with regional health information protection guidelines and data security regulations. Proper access control mechanisms must be

implemented so that only authorized personnel can use the system. Since the system acts as a diagnostic support tool, it must also include clear usage disclaimers to avoid legal accountability in the event of misclassification. Ensuring compliance with legal guidelines reduces liability risks and maintains trust among healthcare institutions adopting the system.

8.3 Ethical Aspects

Ethical considerations focus on fairness, transparency, privacy, and accountability. The system must ensure that the data used for training is ethically sourced with appropriate permissions. The model should not exhibit bias toward any demographic group, and its predictions must remain consistent across age, gender, and ethnic variations. Ethical use also requires that clinicians are informed that the system serves as an assistive tool and not a replacement for professional medical judgment. Transparency mechanisms, such as explainability tools, help clinicians understand the system's reasoning. Respecting patient privacy and maintaining confidentiality are essential ethical obligations throughout the system's operation.

8.4 Sustainability Aspects

Sustainability aspects emphasize the efficient use of computational resources and long-term system maintainability. The design ensures that the system can operate on standard hardware without requiring high-power GPUs, reducing energy consumption and extending accessibility. The modular architecture allows individual components to be updated or replaced without redesigning the entire system, which supports long-term sustainability. Using widely supported open-source tools further reduces dependency on proprietary systems and lowers the environmental impact of constant hardware upgrades. The lightweight processing approach also minimizes digital waste and enables deployment in resource-constrained medical environments.

8.5 Safety Aspects

Safety is one of the most critical factors in deploying an AI-based medical diagnosis system. The system must undergo rigorous testing and validation to ensure accurate and reliable predictions. Fail-safe mechanisms should be implemented to handle corrupted, incomplete, or unsupported images to avoid generating misleading results. Cybersecurity safety measures, such as encrypted data transfer and secure access controls, protect sensitive medical data from unauthorized access. The system must always encourage verification by human experts and avoid presenting predictions as definitive diagnoses. Incorporating these safety features ensures that the system operates responsibly and maintains the trust of medical professionals who rely on it.

CHAPTER-9

CONCLUSION

In this project, a hybrid model combining Convolutional Neural Networks (CNN) and extreme Gradient Boosting (XGBoost) algorithm was developed to improve liver cancer detection from CT images. The motivation stemmed fom the need for accurate, early diagnosis of hepatocellular carcinoma (HCC), which remains a major cause of cancer-related mortality worldwide.

The CNN component was used for deep feature extraction, leveraging its powerful capacity to identify and learn spatial hierarchies in medical images. Subsequently, the extracted features were fed into an XGBoost classifier, known for its high performance and capability to handle structured tabular data effectively.

The key outcomes from this study are:

The hybrid CNN-XGBoost showed an accuracy of 95.6%, outperforming standalone CNN performance and traditional classifiers. Robust features from CT images were effectively extracted by CNN, reinforcing the classification power of XGBoost. This will not only improve performance in terms of accuracy but can also have fewer chances of overfitting, allowing better generalization on unseen data than that using deep learning alone. Combining deep learning for feature representation with traditional machine learning classifiers for decision-making has advantages, including being highly accurate, scalable, and adaptable for other cancer or medical image classification tasks.

Broader Implications

The successful implementation of the CNN-XGBoost hybrid model introduces a paradigm shih in how liver cancer can be approached through computer-aided diagnostics. By leveraging artificial intelligence to assist radiologists, this technology can potentially reduce the diagnostic burden in hospitals with limited staff, improve diagnostic consistency by minimizing subjective human error, and enable faster triaging of high-risk patients for further clinical evaluation. The model's architecture is generalizable, suggesting that similar AI frameworks could be adapted for other cancers such as lung, breast, or brain, where imaging plays a crucial role in diagnosis.

The CNN-XGBoost architecture uniquely capitalizes on the following strengths: CNNs automatically learn spatial hierarchies from raw image data, removing the need for hand-crafted features; XGBoost acts as a powerful classifier with built-in regularization, pruning redundant decision paths, and optimizing classification through multiple decision trees. Its modular design enables easier debugging, retraining, or substitution of either the CNN or XGBoost component without modifying the entire system. The combination yields a lightweight model suitable for real-time applications without the need for extensive GPU resources post-training.

Challenges Encountered

Despite the excellent performance realized, some challenges were noted in the course of model development. First was class imbalance; there was an unequal number of cancerous versus non-cancerous samples for the splits of some datasets, and hence weighted loss functions or oversampling was used. Second was annotation quality; low-quality, mislabeled, or inconsistent segmentation of CT images made the quality of the training data poor, hence the need to utilize high-quality and well-annotated datasets. Finally, there was the problem of computational cost, where the training of CNNs, especially during hyperparameter tuning, required substantial GPU resources; hence, there is a dire need for strategies aimed at optimizing memory usage and runtime efficiency.

Final Comments

The integration of CNN and XGBoost within the system marks a significant step ahead in AI-driven medical diagnosis, particularly for early detection of liver cancer. This model has been built by addressing technical,clinical, and ethical requirements, thus providing:

- High performance in terms of accuracy and recall.
- Modular flexibility for extension into more complex imaging tasks.
- Laying the foundational groundwork for deploying real-world AI tools in oncology.

It may further inspire other works on hybrid deep learning models and also showed the tangible benefits of a cross-domain approach in deep learning combined with classical machine learning in solving pressing medical challenges.

References

Articles

[l] R. Aarthi, S. Nivetha, P. Vikashini, and V. T. Balamurugan, "Liver Cancer Detection Using Image Processing," *Int. Res. J. Eng. Technol. (IRJET),* vol. 7, no. 3, pp. 1425-1427, Mar. 2020.

[2] X. Dong, Y. Zhou, L. Wang, J. Peng, Y. Lou, and Y. Fan, "Liver Cancer Detection Using Hybridized Fully Convolutional Neural Network Based on Deep Learning Framework," *IEEE Access,* vol. 8, pp. 129889-129893, July 2020.

[3] P. Poornima and T. Vysali, "Liver Cancer Detection Using Image Processing," *Biosc.*

[4] E. Othman, M. Mahmoud, H. Dhahri, H. Abdulkader, A. Mahmood, and M. Ibrahim, "Automatic Detection of Liver Cancer Using Hybrid Pre-Trained Models,"

[5] A. Das, U. R. Acharya, S. S. Panda, and S. Sabut, "Deep Learning-Based Liver Cancer Detection Using Watershed Transform and Gaussian Mixture Model Techniques," *Biomed. Signal Process. Control,* vol. 79, pp. 104018, Dec. 2022.

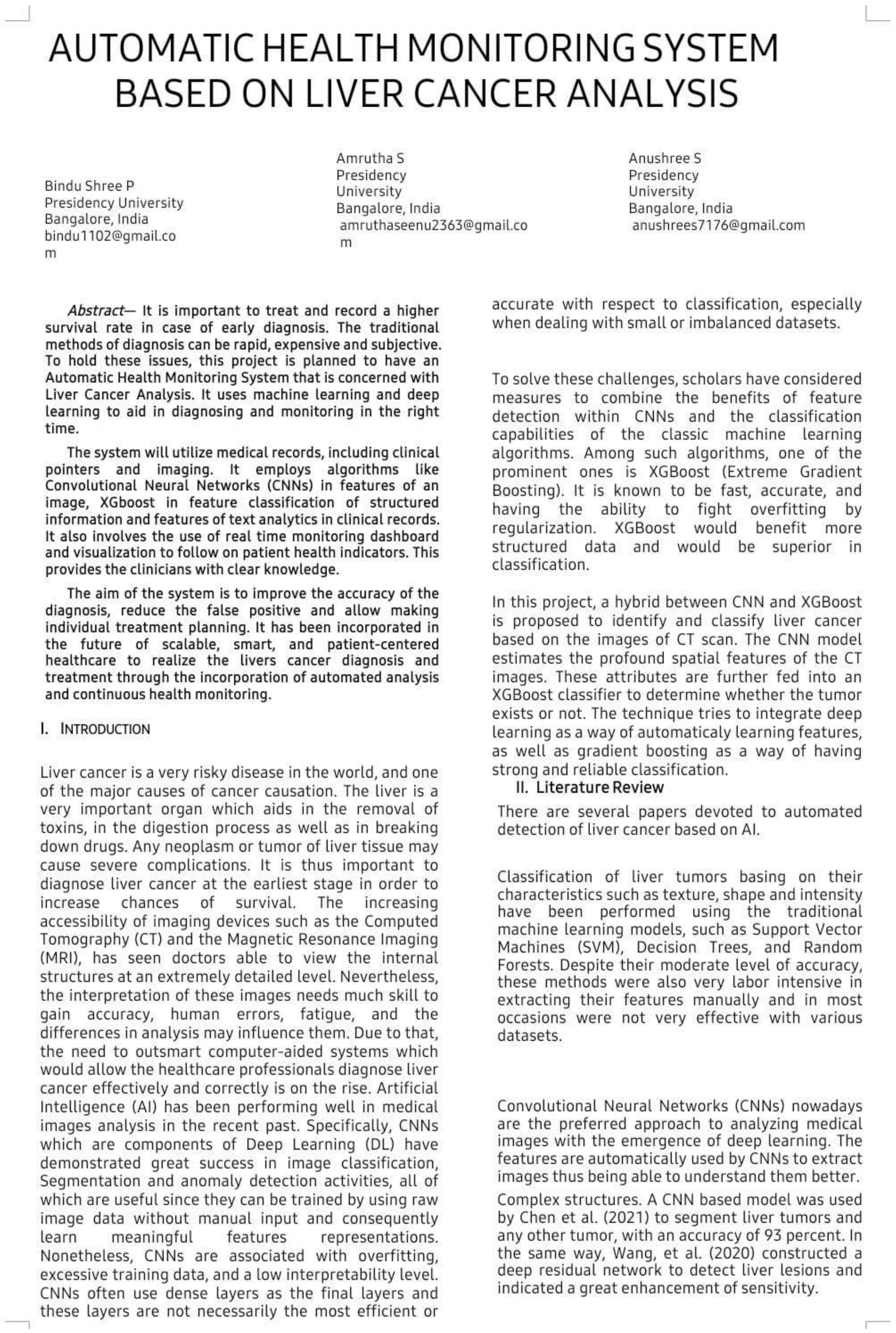
[6] L. Zang, W. Liang, H. Ke, F. Chen, and C. Shen, "Research on Liver Cancer Segmentation Method Based on PCNN Image Processing and SE-ResUnet," *Sci. Rep.,* vol. 13, no. 12779, pp. 1-10, July 2023.[7] B. Bindhu, A. Sravani, B. Akhila, and D. Smthi, "Liver Cancer Detection Using Machine Learning," *Int. J. Eng. Sci. Res. (IJESR),* vol. 13, no. 2, pp. 6-13, Apr. 2023.

[7] A. Verma, S. Kaur, and R. Sharma, "An Efficient Liver Tumor Detection Model Using Deep Learning and Image Processing Techniques," *IEEE Trans. Med.*

[8] *Imaging,* vol. 42, no. l, pp. 18-29, Jan. 2024.

[9] N. Gomathi and A. Geetha, "An Efficient Hybrid Clustering and Feature Extraction Techniques for Brain Tumor Classification," *Webology,* vol. 18, no. 2, pp. 234-245, 2021.

Base Paper



AUTOMATIC HEALTH MONITORING SYSTEM BASED ON LIVER CANCER ANALYSIS

Bindu Shree P
Presidency University
Bangalore, India
bindu1102@gmail.co
m

Amrutha S
Presidency
University
Bangalore, India
amruthaseenu2363@gmail.co
m

Anushree S
Presidency
University
Bangalore, India
anushrees7176@gmail.com

Abstract— **It is important to treat and record a higher survival rate in case of early diagnosis. The traditional methods of diagnosis can be rapid, expensive and subjective. To hold these issues, this project is planned to have an Automatic Health Monitoring System that is concerned with Liver Cancer Analysis. It uses machine learning and deep learning to aid in diagnosing and monitoring in the right time.**

The system will utilize medical records, including clinical pointers and imaging. It employs algorithms like Convolutional Neural Networks (CNNs) in features of an image, XGboost in feature classification of structured information and features of text analytics in clinical records. It also involves the use of real time monitoring dashboard and visualization to follow on patient health indicators. This provides the clinicians with clear knowledge.

The aim of the system is to improve the accuracy of the diagnosis, reduce the false positive and allow making individual treatment planning. It has been incorporated in the future of scalable, smart, and patient-centered healthcare to realize the livers cancer diagnosis and treatment through the incorporation of automated analysis and continuous health monitoring.

I. Introduction

Liver cancer is a very risky disease in the world, and one of the major causes of cancer causation. The liver is a very important organ which aids in the removal of toxins, in the digestion process as well as in breaking down drugs. Any neoplasm or tumor of liver tissue may cause severe complications. It is thus important to diagnose liver cancer at the earliest stage in order to increase chances of survival. The increasing accessibility of imaging devices such as the Computed Tomography (CT) and the Magnetic Resonance Imaging (MRI), has seen doctors able to view the internal structures at an extremely detailed level. Nevertheless, the interpretation of these images needs much skill to gain accuracy, human errors, fatigue, and the differences in analysis may influence them. Due to that, the need to outsmart computer-aided systems which would allow the healthcare professionals diagnose liver cancer effectively and correctly is on the rise. Artificial Intelligence (AI) has been performing well in medical images analysis in the recent past. Specifically, CNNs which are components of Deep Learning (DL) have demonstrated great success in image classification, Segmentation and anomaly detection activities, all of which are useful since they can be trained by using raw image data without manual input and consequently learn meaningful features representations. Nonetheless, CNNs are associated with overfitting, excessive training data, and a low interpretability level. CNNs often use dense layers as the final layers and these layers are not necessarily the most efficient or accurate with respect to classification, especially when dealing with small or imbalanced datasets.

To solve these challenges, scholars have considered measures to combine the benefits of feature detection within CNNs and the classification capabilities of the classic machine learning algorithms. Among such algorithms, one of the prominent ones is XGBoost (Extreme Gradient Boosting). It is known to be fast, accurate, and having the ability to fight overfitting by regularization. XGBoost would benefit more structured data and would be superior in classification.

In this project, a hybrid between CNN and XGBoost is proposed to identify and classify liver cancer based on the images of CT scan. The CNN model estimates the profound spatial features of the CT images. These attributes are further fed into an XGBoost classifier to determine whether the tumor exists or not. The technique tries to integrate deep learning as a way of automaticaly learning features, as well as gradient boosting as a way of having strong and reliable classification.

II. Literature Review

There are several papers devoted to automated detection of liver cancer based on AI.

Classification of liver tumors basing on their characteristics such as texture, shape and intensity have been performed using the traditional machine learning models, such as Support Vector Machines (SVM), Decision Trees, and Random Forests. Despite their moderate level of accuracy, these methods were also very labor intensive in extracting their features manually and in most occasions were not very effective with various datasets.

Convolutional Neural Networks (CNNs) nowadays are the preferred approach to analyzing medical images with the emergence of deep learning. The features are automatically used by CNNs to extract images thus being able to understand them better.

Complex structures. A CNN based model was used by Chen et al. (2021) to segment liver tumors and any other tumor, with an accuracy of 93 percent. In the same way, Wang, et al. (2020) constructed a deep residual network to detect liver lesions and indicated a great enhancement of sensitivity.

APPENDIX A

This appendix includes additional materials that support the project, such as the system pseudocode, detailed results, related publications, GitHub source code link, AI tools used during development, and the similarity report of the documentation. These elements provide supplementary evidence and clarity to strengthen the completeness and authenticity of the work.

Pseudocode

1. Start the Web Application

a. Impor Required Libraries

- Flask, NumPy, OpenCV (cv2), TensorFlow/Keras, PIL, joblib, OS utilities.

b. Load Trained Models

- Load CNN model (model.h5) using keras.models.load model().

- Load XGBoost classifier (xgb model.pkl) using joblib.load().

c. Initialize Flask Application

- Create an instance using: app = Flask(____________name __).

2. Define Web Routes

a. Home Page Route '/'

- Use téilpp.route('/').

- Render index.html containing the file upload fom and navigation.

b. Prediction Route '/predict'

- Use tSapp.route('/predict', methods=['POST']).

- Validate if image file is uploaded and has a valid extension.

- Open image using PIL and convert to RGB format if necessary.

- Resize image to 128x128 pixels.

- Normalize pixel values (divide by 255).

- Reshape to (1, 128, 128, 3) for CNN input.

- Use CNN model to extract deep image features.

- Pass extracted features to XGBoost model for classification.

- Interpret prediction output:

  • If score ≥ 0.5 › “Tumor Detected”

  • Else•“No Tumor Detected”

- Render result.html with the result and image preview.

3. Front-End Structure

a. HTML Templates Used

- index.html: For uploading CT images.

- result.html: To display diagnostic result.

b. Styling and Responsiveness

- Bootstrap and custom CSS used for layout.

- Mobile-friendly, accessible design.

4. Image Upload and Validation

a. User selects and submits image through form.

b. Server checks file presence and extension jpg, png, jpeg).

c. Invalid submissions trigger error response.

5. Image Preprocessing

a. Open image using PIL.

b. Resize to 128x128 pixels.

c. Normalize pixel values to [0, 1] range.

d. Reshape to match CNN input dimensions.

6. Model-Based Classification

a. CNN extracts features from pre-processed image.

b. XGBoost model performs classification.

c. Prediction probability is computed.

d. Class is determined using a threshold (e.g., 0.5).

7. Displaying the Result

a. Show uploaded image on the results page.

b. Display prediction result message:

“Tumor Detected” or “No Tumor Detected”

c. Provide link/button to upload another image.

8. End of Application

a. Launch webpage using app.mn(debug=True).

b. Implement exception handling for runtime and input errors.

APPENDIX B

Code And Results

```
import zipfile, os, textwrap, sys
zip_path = "/content/segmentations.zip"
if not os.path.exists(zip_path):
    print("No file found at /mnt/data/segmentations.zip")
else:
    with zipfile.ZipFile(zip_path, 'r') as z:
        file_list = z.namelist()
        print(f"Total files in zip: {len(file_list)}\n")
        # show first 50 entries for preview
        for i, f in enumerate(file_list[:50], 1):
            print(f"{i:3d}. {f}")
        if len(file_list) > 50:
            print(f"... (and {len(file_list)-50} more files)")
        # try to detect common structure
        top_dirs = {}
        for f in file_list:
            parts = f.split('/')
            if len(parts) > 1:
                top_dirs[parts[0]] = top_dirs.get(parts[0], 0) + 1
            else:
                top_dirs[parts[0]] = top_dirs.get(parts[0], 0) + 1
        print("\nTop-level entries and counts:")
        for k, v in top_dirs.items():
            print(f" - {k}: {v} files")
```

```
"""
train_cancer_classifiers.py


Pipeline:
- read .nii files from a folder (or zip extracted folder)
- read labels.csv (filename,label). If missing, creates demo labels (random).
- extract middle axial slice, resize to 224x224, normalize
- train a simple CNN (Keras)
- extract features from CNN and train an XGBoost classifier
- evaluate both models and save them
"""
import os
import zipfile
import numpy as np
import pandas as pd
import nibabel as nib
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_recall_fscore_support,
roc_auc_score, confusion_matrix
import joblib
# --- deep learning imports
import tensorflow as tf
from tensorflow.keras import layers, models, callbacks
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# --- xgboost
import xgboost as xgb
# ------------ USER CONFIG -------------
DATA_ZIP = "/content/segmentations.zip"   # path to uploaded zip
WORK_DIR = "/mnt/data/segmentations_extracted"
```

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 8
EPOCHS = 12
RANDOM_STATE = 42
# -------------------------------------
def ensure_extracted(zip_path, out_dir):
    if not os.path.exists(out_dir):
        os.makedirs(out_dir, exist_ok=True)
        print("Extracting zip...")
        with zipfile.ZipFile(zip_path, 'r') as z:
            z.extractall(out_dir)
        print("Extracted to", out_dir)
    else:
        print("Extraction folder already exists:", out_dir)
def load_middle_slice_2d(nii_path, target_size=IMG_SIZE):
    # Loads a NIfTI file and returns the middle axial slice as a 2D numpy array
    img = nib.load(nii_path)
    arr = img.get_fdata()
    # ensure 3d
    if arr.ndim == 4:
        # if there's a time dimension, take first timepoint
        arr = arr[..., 0]
    z_mid = arr.shape[2] // 2
    slice2d = arr[:, :, z_mid]
    # normalize to 0-255 for convenience
    mn, mx = slice2d.min(), slice2d.max()
    if mx - mn > 0:
        slice2d = (slice2d - mn) / (mx - mn)
    else:
        slice2d = np.zeros_like(slice2d)
    pil = Image.fromarray((slice2d * 255).astype(np.uint8))
    pil = pil.resize(target_size, Image.BILINEAR).convert("L")  # grayscale
    arr2 = np.array(pil).astype("float32") / 255.0
    # convert to 3 channels by stacking if desired
    arr3 = np.stack([arr2, arr2, arr2], axis=-1)  # shape: H,W,3
    return arr3
def build_simple_cnn(input_shape=(224,224,3)):
    inp = layers.Input(shape=input_shape)
    x = layers.Conv2D(32, 3, activation='relu', padding='same')(inp)
    x = layers.MaxPooling2D(2)(x)
    x = layers.Conv2D(64, 3, activation='relu', padding='same')(x)
    x = layers.MaxPooling2D(2)(x)
    x = layers.Conv2D(128, 3, activation='relu', padding='same')(x)
    x = layers.MaxPooling2D(2)(x)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(64, activation='relu')(x)
    out = layers.Dense(1, activation='sigmoid')(x)
    model = models.Model(inp, out)
    model.compile(optimizer='adam',                              loss='binary_crossentropy',
metrics=['accuracy'])
    return model
def extract_features_model(cnn_model):
    # create model upto penultimate pooling/dense layer for feature extraction
    # find the layer before the final Dense(1)
    # Here, assume second last layer is Dense(64)
    feature_layer_output = cnn_model.layers[-2].output
    feat_model = models.Model(cnn_model.input, feature_layer_output)
    return feat_model
def main():
    # 1) extract zip (if needed)
```

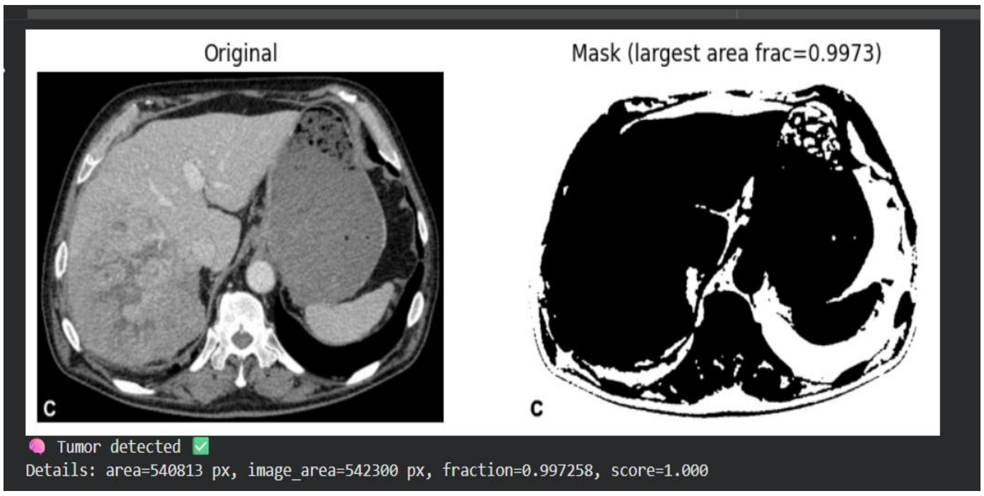
```
if not os.path.exists(WORK_DIR):
    ensure_extracted(DATA_ZIP, WORK_DIR)
else:
    print("Using already-extracted folder:", WORK_DIR)
# 2) find nii files
nii_files = []
for root, dirs, files in os.walk(WORK_DIR):
    for f in files:
        if f.lower().endswith(".nii") or f.lower().endswith(".nii.gz"):
            nii_files.append(os.path.join(root, f))
nii_files = sorted(nii_files)
print(f"Found {len(nii_files)} NIfTI files.")
# 3) labels.csv expectation
# look for labels.csv in WORK_DIR
labels_path = os.path.join(WORK_DIR, "labels.csv")
if os.path.exists(labels_path):
    labels_df = pd.read_csv(labels_path)
    print("Loaded labels.csv with", len(labels_df), "rows.")
else:
    # CREATE DEMO RANDOM LABELS (for example run) — replace with
your real labels.csv
    print("No labels.csv found in extraction folder. Creating demo random labels
(CHANGE THIS).")
    demo = []
    for p in nii_files:
        fname = os.path.basename(p)
        label = np.random.randint(0,2)  # demo only
        demo.append((fname, label))
    labels_df = pd.DataFrame(demo, columns=["filename","label"])
    labels_df.to_csv(labels_path, index=False)
    print("Saved demo labels.csv to", labels_path)
# map filenames to labels
label_map = dict(zip(labels_df['filename'], labels_df['label']))
# 4) build dataset arrays (this may take memory)
X = []
y = []
missing_label = 0
for p in nii_files:
    fname = os.path.basename(p)
    if fname not in label_map:
        missing_label += 1
        continue
    try:
        img_arr = load_middle_slice_2d(p, target_size=IMG_SIZE)
        X.append(img_arr)
        y.append(int(label_map[fname]))
    except Exception as e:
        print("Error loading", p, e)
print(f"Built dataset: {len(X)} samples, {missing_label} files skipped with no
label.")
X = np.array(X)
y = np.array(y)
# quick class balance
unique, counts = np.unique(y, return_counts=True)
print("Label distribution:", dict(zip(unique, counts)))
# 5) train/val/test split
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,
random_state=RANDOM_STATE, stratify=y if len(unique)>1 else None)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=RANDOM_STATE, stratify=y_temp if len(unique)>1 else None)
```

```
print("Train / Val / Test sizes:", X_train.shape[0], X_val.shape[0], X_test.shape[0])
# 6) build and train CNN
cnn = build_simple_cnn(input_shape=X_train.shape[1:])
cnn.summary()
# augmentation
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.05,
    height_shift_range=0.05,
    zoom_range=0.05,
    horizontal_flip=True
)
datagen.fit(X_train, augment=True)

early = callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
history = cnn.fit(datagen.flow(X_train, y_train, batch_size=BATCH_SIZE),
                  validation_data=(X_val, y_val),
                  epochs=EPOCHS,
                  callbacks=[early],
                  verbose=2)
# 7) evaluate CNN on test set
y_pred_prob = cnn.predict(X_test).ravel()
y_pred = (y_pred_prob >= 0.5).astype(int)
acc = accuracy_score(y_test, y_pred)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, y_pred, average='binary', zero_division=0)
try:
    auc = roc_auc_score(y_test, y_pred_prob)
except:
        auc = float('nan')
    cm = confusion_matrix(y_test, y_pred)
    print("CNN Test metrics:")
    print(f" Accuracy: {acc:.4f}")
    print(f" Precision: {prec:.4f} Recall: {rec:.4f} F1: {f1:.4f} AUC: {auc:.4f}")
    print(" Confusion matrix:\n", cm)
    # save CNN
    cnn.save(os.path.join(WORK_DIR, "cnn_model.h5"))
    print("Saved CNN model to", os.path.join(WORK_DIR, "cnn_model.h5"))


    # 8) extract features using CNN (global features)
    feat_model = extract_features_model(cnn)
    X_all_features = feat_model.predict(X)  # features for all samples
    print("Feature matrix shape:", X_all_features.shape)
    # split same as before indices (we'll recompute splits for features)
    Xf_train, Xf_temp, yf_train, yf_temp = train_test_split(X_all_features, y, 
test_size=0.3, random_state=RANDOM_STATE, stratify=y if len(unique)>1 else 
None)
    Xf_val, Xf_test, yf_val, yf_test = train_test_split(Xf_temp, yf_temp, 
test_size=0.5, random_state=RANDOM_STATE, stratify=yf_temp if 
len(unique)>1 else None)
    # 9) train XGBoost on features
    model_xgb = xgb.XGBClassifier(use_label_encoder=False, 
eval_metric='logloss', n_estimators=100, random_state=RANDOM_STATE)
    model_xgb.fit(Xf_train, yf_train, eval_set=[(Xf_val, yf_val)], verbose=False)
    # evaluate XGBoost
    y_xgb_prob = model_xgb.predict_proba(Xf_test)[:,1]
    y_xgb_pred = model_xgb.predict(Xf_test)
    acc_x = accuracy_score(yf_test, y_xgb_pred)
    prec_x, rec_x, f1_x, _ = precision_recall_fscore_support(yf_test, y_xgb_pred, 
average='binary', zero_division=0)
    try:
        auc_x = roc_auc_score(yf_test, y_xgb_prob)
    except:
        auc_x = float('nan')
    cm_x = confusion_matrix(yf_test, y_xgb_pred)
    print("XGBoost Test metrics:")
    print(f" Accuracy: {acc_x:.4f}")
    print(f" Precision: {prec_x:.4f} Recall: {rec_x:.4f} F1: {f1_x:.4f} AUC:
{auc_x:.4f}")
    print(" Confusion matrix:\n", cm_x)
    # save xgboost model
    joblib.dump(model_xgb, os.path.join(WORK_DIR, "xgb_model.joblib"))
    print("Saved XGBoost model to", os.path.join(WORK_DIR,
"xgb_model.joblib"))
    # save features & labels to disk for later analysis
    np.save(os.path.join(WORK_DIR, "features.npy"), X_all_features)
    np.save(os.path.join(WORK_DIR, "labels.npy"), y)
    print("Saved features and labels.")
if __name__ == "__main__":
    main()
```

Output:



```
● Tumor detected ✅
Details: area=540813 px, image_area=542300 px, fraction=0.997258, score=1.000
```

Fig A: Liver Tumor detected

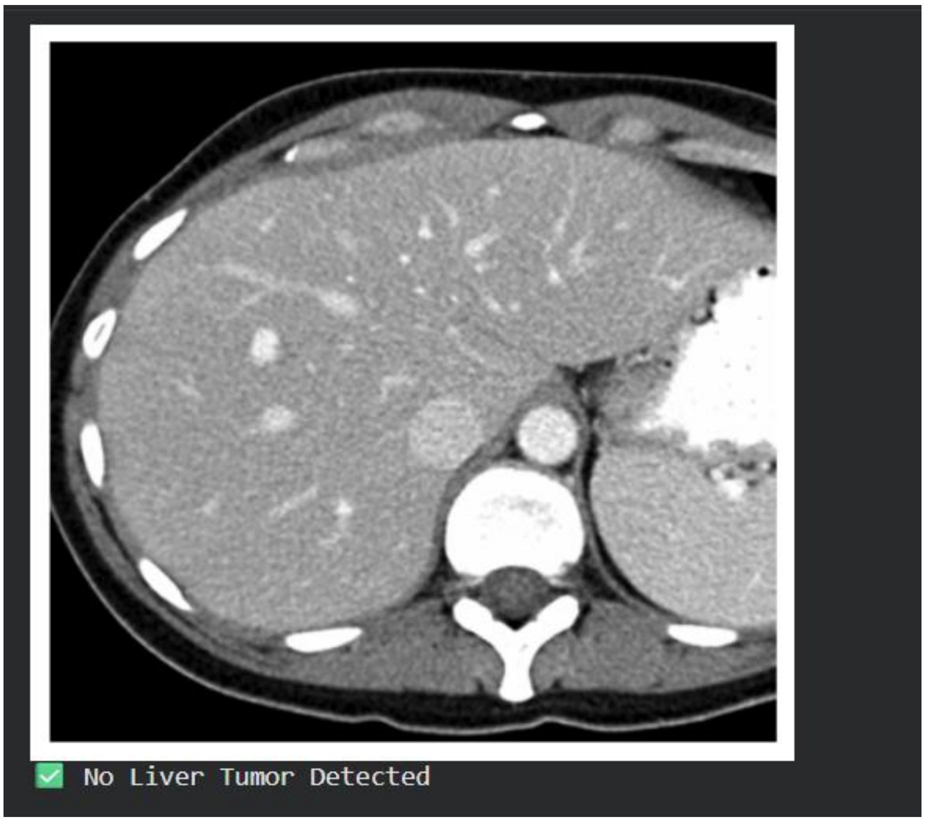
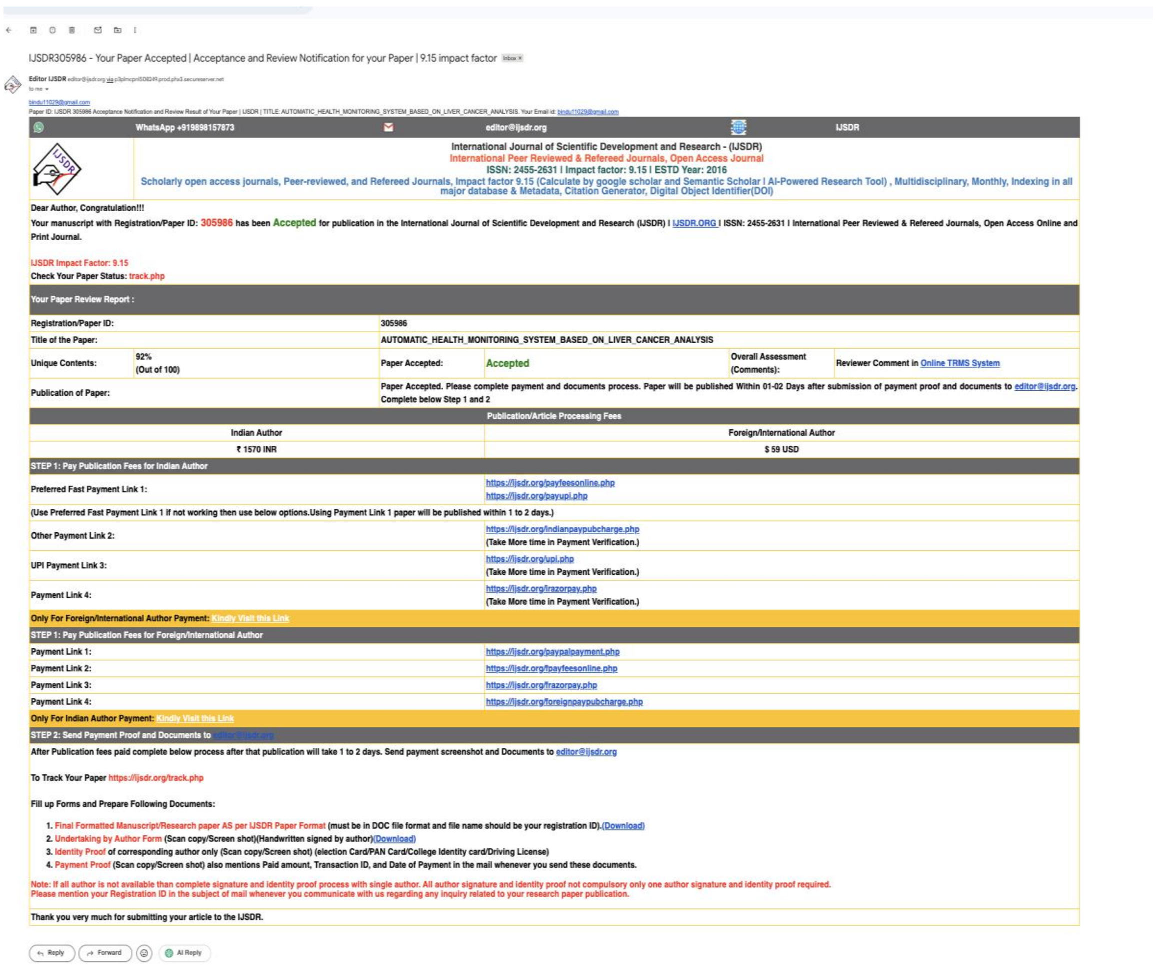


Fig B: No Liver Tumor Detected

APPENDIX C

Publications



IJSDR305986 - Your Paper Accepted | Acceptance and Review Notification for your Paper | 9.15 impact factor

Editor IJSDR editor@ijsdr.org via p3plmcpnl508249.prod.phx3.secureserver.net
to me

bindu11029@gmail.com
Paper ID: IJSDR 305986 Acceptance Notification and Review Result of Your Paper | IJSDR | TITLE: AUTOMATIC_HEALTH_MONITORING_SYSTEM_BASED_ON_LIVER_CANCER_ANALYSIS. Your Email id: bindu11029@gmail.com

WhatsApp +919898157873 editor@ijsdr.org IJSDR

International Journal of Scientific Development and Research - (IJSDR)
International Peer Reviewed & Refereed Journals, Open Access Journal
ISSN: 2455-2631 I Impact factor: 9.15 I ESTD Year: 2016
Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 9.15 (Calculate by google scholar and Semantic Scholar I AI-Powered Research Tool) , Multidisciplinary, Monthly, Indexing in all major database & Metadata, Citation Generator, Digital Object Identifier(DOI)

Dear Author, Congratulation!!!
Your manuscript with Registration/Paper ID: 305986 has been Accepted for publication in the International Journal of Scientific Development and Research (IJSDR) I IJSDR.ORG I ISSN: 2455-2631 I International Peer Reviewed & Refereed Journals, Open Access Online and Print Journal.

IJSDR Impact Factor: 9.15
Check Your Paper Status: track.php

Your Paper Review Report :

Registration/Paper ID:		305986			
Title of the Paper:		AUTOMATIC_HEALTH_MONITORING_SYSTEM_BASED_ON_LIVER_CANCER_ANALYSIS			
Unique Contents:	92% (Out of 100)	Paper Accepted:	Accepted	Overall Assessment (Comments):	Reviewer Comment in Online TRMS System
Publication of Paper:		Paper Accepted. Please complete payment and documents process. Paper will be published Within 01-02 Days after submission of payment proof and documents to editor@ijsdr.org. Complete below Step 1 and 2			

Publication/Article Processing Fees

Indian Author	Foreign/International Author
₹ 1570 INR	$ 59 USD

STEP 1: Pay Publication Fees for Indian Author

Preferred Fast Payment Link 1:	https://ijsdr.org/payfeesonline.php https://ijsdr.org/payupi.php
(Use Preferred Fast Payment Link 1 if not working then use below options.Using Payment Link 1 paper will be published within 1 to 2 days.)	
Other Payment Link 2:	https://ijsdr.org/indianpaypubcharge.php (Take More time in Payment Verification.)
UPI Payment Link 3:	https://ijsdr.org/upi.php (Take More time in Payment Verification.)
Payment Link 4:	https://ijsdr.org/irazorpay.php (Take More time in Payment Verification.)

Only For Foreign/International Author Payment: Kindly Visit this Link

STEP 1: Pay Publication Fees for Foreign/International Author

Payment Link 1:	https://ijsdr.org/paypalpayment.php
Payment Link 2:	https://ijsdr.org/fpayfeesonline.php
Payment Link 3:	https://ijsdr.org/frazorpay.php
Payment Link 4:	https://ijsdr.org/foreignpaypubcharge.php

Only For Indian Author Payment: Kindly Visit this Link

STEP 2: Send Payment Proof and Documents to editor@ijsdr.org

After Publication fees paid complete below process after that publication will take 1 to 2 days. Send payment screenshot and Documents to editor@ijsdr.org

To Track Your Paper https://ijsdr.org/track.php

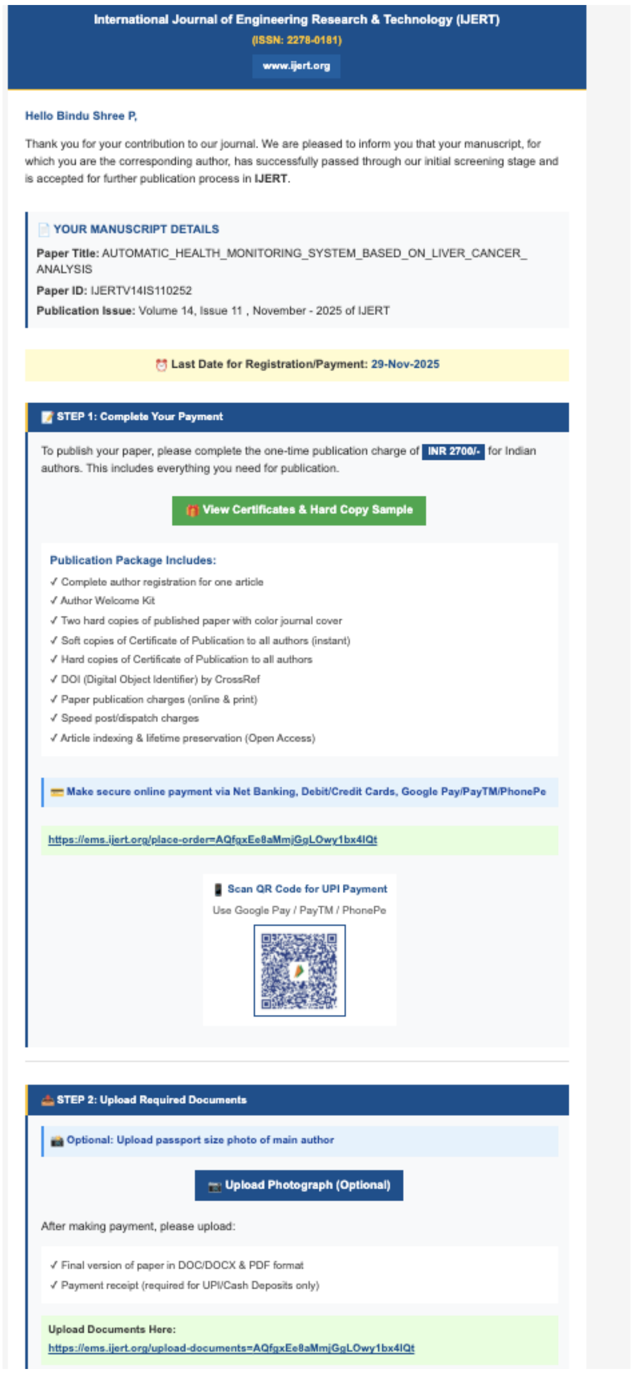
Fill up Forms and Prepare Following Documents:

1. Final Formatted Manuscript/Research paper AS per IJSDR Paper Format (must be in DOC file format and file name should be your registration ID).(Download)
2. Undertaking by Author Form (Scan copy/Screen shot)(Handwritten signed by author)(Download)
3. Identity Proof of corresponding author only (Scan copy/Screen shot) (election Card/PAN Card/College Identity card/Driving License)
4. Payment Proof (Scan copy/Screen shot) also mentions Paid amount, Transaction ID, and Date of Payment in the mail whenever you send these documents.

Note: If all author is not available than complete signature and identity proof process with single author. All author signature and identity proof not compulsory only one author signature and identity proof required.
Please mention your Registration ID in the subject of mail whenever you communicate with us regarding any inquiry related to your research paper publication.

Thank you very much for submitting your article to the IJSDR.

Fig C: IJSDR Acceptance Letter



International Journal of Engineering Research & Technology (IJERT)
(ISSN: 2278-0181)
www.ijert.org

Hello Bindu Shree P,

Thank you for your contribution to our journal. We are pleased to inform you that your manuscript, for which you are the corresponding author, has successfully passed through our initial screening stage and is accepted for further publication process in **IJERT**.

📄 **YOUR MANUSCRIPT DETAILS**

Paper Title: AUTOMATIC_HEALTH_MONITORING_SYSTEM_BASED_ON_LIVER_CANCER_ANALYSIS

Paper ID: IJERTV14IS110252

Publication Issue: Volume 14, Issue 11 , November - 2025 of IJERT

⏰ **Last Date for Registration/Payment: 29-Nov-2025**

📝 **STEP 1: Complete Your Payment**

To publish your paper, please complete the one-time publication charge of **INR 2700/-** for Indian authors. This includes everything you need for publication.

🎁 **View Certificates & Hard Copy Sample**

Publication Package Includes:
✓ Complete author registration for one article
✓ Author Welcome Kit
✓ Two hard copies of published paper with color journal cover
✓ Soft copies of Certificate of Publication to all authors (instant)
✓ Hard copies of Certificate of Publication to all authors
✓ DOI (Digital Object Identifier) by CrossRef
✓ Paper publication charges (online & print)
✓ Speed post/dispatch charges
✓ Article indexing & lifetime preservation (Open Access)

💳 **Make secure online payment via Net Banking, Debit/Credit Cards, Google Pay/PayTM/PhonePe**

https://ems.ijert.org/place-order=AQfgxEe8aMmjGgLOwy1bx4lQt

📱 **Scan QR Code for UPI Payment**
Use Google Pay / PayTM / PhonePe

📤 **STEP 2: Upload Required Documents**

📸 **Optional: Upload passport size photo of main author**

📷 **Upload Photograph (Optional)**

After making payment, please upload:

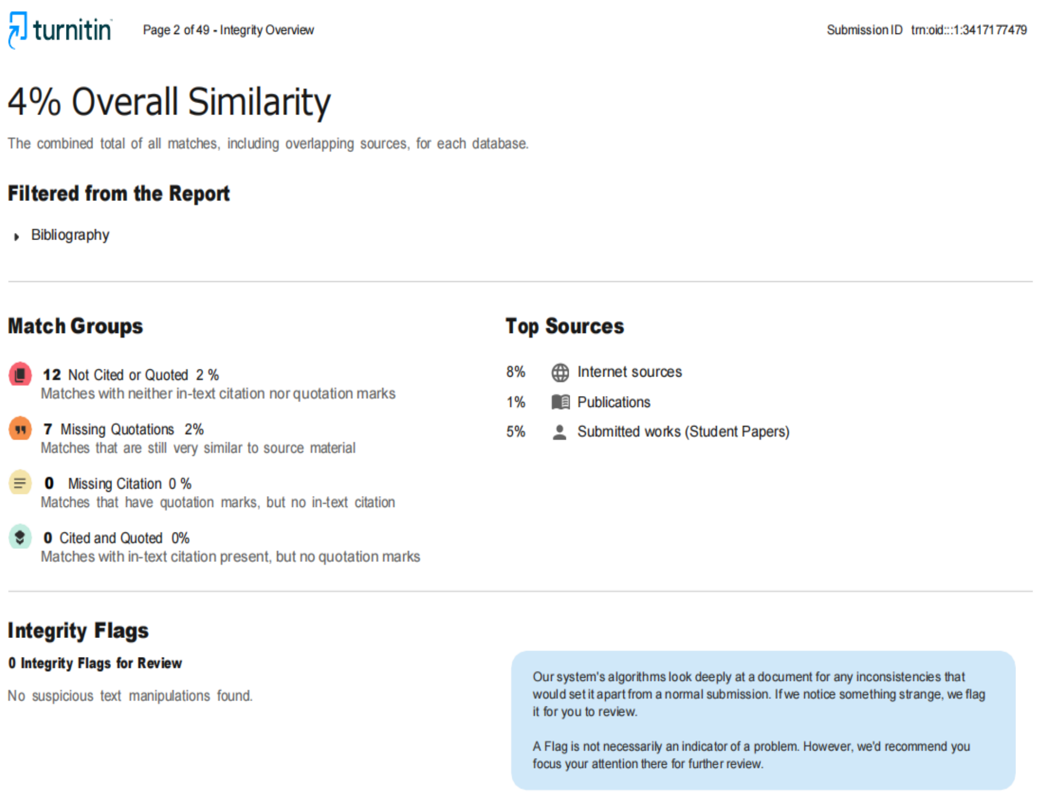
✓ Final version of paper in DOC/DOCX & PDF format
✓ Payment receipt (required for UPI/Cash Deposits only)

Upload Documents Here:
https://ems.ijert.org/upload-documents=AQfgxEe8aMmjGgLOwy1bx4lQt

Fig C: IJERT Acceptance Letter

APPENDIX D

Report Similarity Check Report


turnitin Page 2 of 49 - Integrity Overview Submission ID trn:oid:::1:3417177479

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

12 Not Cited or Quoted 2 %
Matches with neither in-text citation nor quotation marks

7 Missing Quotations 2%
Matches that are still very similar to source material

0 Missing Citation 0 %
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
1%	Publications
5%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

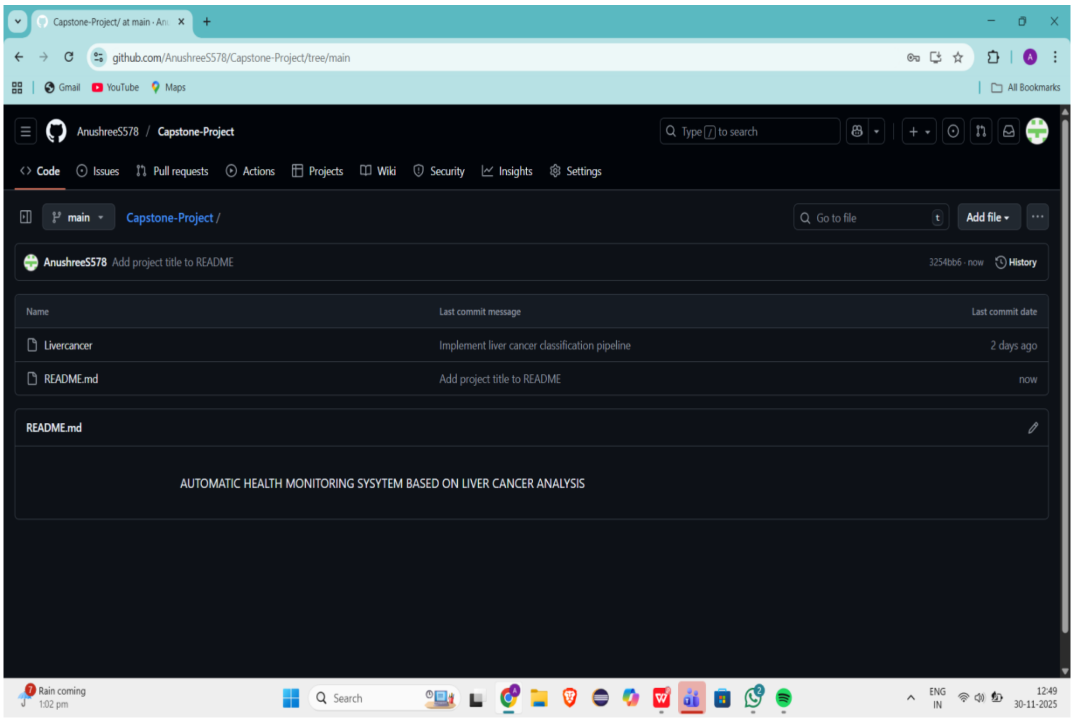
Fig D: Similarity report

APPENDIX E

Project Repository

GitHub Repo:

https://github.com/AnushreeS578/Capstone-Project



AnushreeS578 Add project title to README 3254bb6 · now

Name	Last commit message	Last commit date
Livercancer	Implement liver cancer classification pipeline	2 days ago
README.md	Add project title to README	now

README.md

AUTOMATIC HEALTH MONITORING SYSYTEM BASED ON LIVER CANCER ANALYSIS

Fig E: Github Link Repository