

LIVER CANCER SEGMENTATION AND CLASSIFICATION

A project report submitted for the partial fulfillment of academic
requirements for the award of degree of

MASTER OF COMPUTER APPLICATIONS

Submitted by

DHRUVA KUMAR K M

4NI21MC013

Under the supervision of

Dr. SANJAY KUMAR C.K

**Assistant Professor and Head Of The Department
Department of MCA, NIE Mysuru**



**Department of Master of Computer Applications
The National Institute of Engineering**

(An Autonomous Institute under Visvesvaraya Technological University, Belagavi)

Manandavadi Road, Mysuru – 570 008, Karnataka, INDIA

2022-2023



Estd.: 1946

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS THE NATIONAL INSTITUTE OF ENGINEERING

(An Autonomous Institution)

Manandavadi Road, Mysuru – 570 008.

Phone: 0821-248475, 2481220, 4004900 Fax: 0821-2485802, E-mail: principal@nie.ac.in, Website: www.nie.ac.in

CERTIFICATE

Certified that the Project work entitled “**Liver Cancer Segmentation and Classification**” carried out by **Mr. DHRUVA KUMAR K M, 4NI21MC013**, a bonafide student of The National Institute of Engineering is submitted in partial fulfillment for the award of Master of Computer Applications Degree in **The National Institute of Engineering, Mysuru**, an autonomous institute under Visvesvaraya Technological University, Belagavi during the year 2022-2023. It is certified that all suggestions/ corrections suggested during Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the award of the said Degree.

Name & Signature of the Guide & HoD

Dr. Sanjay Kumar C.K
Assistant Professor & HoD
Dept. of MCA, NIE Mysuru

Department of Master of Computer Applications
The National Institute of Engineering

Mysuru-570 008

External Viva

Signature of the Principal

Dr. Rohini Nagapadma
Principal
NIE Mysuru

Sl. No. Name of the examiners

Signature with date

1.

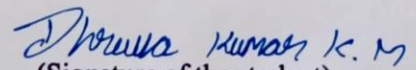
2.

DECLARATION

I, **DHRUVA KUMAR K M** bearing USN: **4NI21MC013** student of **The National Institute of Engineering**, Department of Computer Applications, The National Institute of Engineering, Mysuru hereby declare that the project work entitled "**LIVER CANCER SEGMENTATION AND CLASSIFICATION**" has been carried out by me under the guidance of **Dr. SANJAY KUMAR C.K, Assistant Professor and HoD, Dept. of MCA**. This project work is submitted to **The National Institute of Engineering**, Mysuru, (An Autonomous institute under VTU, Belagavi) in partial fulfillment of the course requirements for the award of degree in **MASTER OF COMPUTER APPLICATIONS** during the academic year 2022-2023. This written submission represents a record of original work and I have adequately cited and referenced the original sources.

Place: Mysuru

Date: 31/08/2023


(Signature of the student)

ACKNOWLEDGEMENT

I express my sincere thanks and indebtedness to my esteemed institution, **THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU** that has provided me with an opportunity to fulfill my desire and reach my goal.

I thank in a very special way to our beloved principal **Dr. Rohini Nagapadma** for her support and help throughout our course and for creating the right academic environment that made my task appreciable.

Thanks, in particular to our Head of the Department and my project guide **Dr. SANJAY KUMAR C.K, Assistant Professor & HoD, Dept. of MCA, NIE Mysuru** for their encouragement and valuable advice.

On a moral personal note, my deepest appreciation and gratitude to my beloved parents, who have been a fountain of inspiration and have provided unrelenting encouragement and support. Also, thanks to all my friends and all the members of my team.

I heartily thank all those who have helped me directly or indirectly in the successful completion of my project.

DHRUVA KUMAR K M

4NI21MC013

ABSTRACT

Liver cancer is an illness that could be fatal and also one of the world's fastest-growing cancers types. Lower death rates result through the early identification of tumors in the liver. By examining photos of tissue obtained from a biopsy of this tumor, the goal of this research is to build a model that can assist clinicians in identifying the type of tumor when it develops in the liver's area. To evaluate whether the tumor is malignant and requires therapy, a tissue expert must put up the effort, take the necessary time, and gather the necessary experience. Therefore, a histology specialist can utilise this model to establish a preliminary diagnosis. Convolutional neural networks (CNNs), which can transmit knowledge, will be used in this study to suggest a deep learning model.

CONTENTS

Details	Page No.
Title page	
Certificate	i
Declaration	ii
Acknowledgment.....	iii
Abstract... ..	iv
1. INTRODUCTION	
1.1 Overview and Motivation... ..	1
1.2 Aim... ..	1
1.3 Problem Statement... ..	1
2. LITERATURE SURVEY	
2.1 Literature review	3
3. SOFTWARE REQUIREMENT SPECIFICATION	
3.1 Introduction... ..	7
3.2 Purpose	7
3.3 Scope	7
3.4 Functional Requirements... ..	8
3.5 Non-Functional Requirements... ..	8
3.6 Tools and technologies used	10
3.6.1 Python	10
3.6.2 Anaconda	11
3.6.3 NumPy... ..	11
3.6.4 Pandas	12
3.6.5 OpenCV	12
3.6.6 Flask.....	13
3.6.7 Deep Learning Overview... ..	13
3.6.8 CNN	14
3.7 Software and Hardware Requirements... ..	17
3.7.1 Software Requirements... ..	17
3.7.2 Hardware Requirements... ..	17
4. METHODOLOGY	18

5. SYSTEM ANALYSIS

5.1	Existing System.....	20
5.1.1	Drawbacks of Existing System.....	20
5.2	Proposed System... ..	20
5.2.1	Advantages of proposed system	21
5.3	Feasibility Study.....	21

6. SYSTEM DESIGN

6.1	Introduction to Design document.....	24
6.2	Purpose	24
6.3	Scope	24
6.4	Flow of the project	24
6.5	Dataflow Diagram	25
6.6	Use Case Diagram... ..	26
6.7	Sequence Diagram	27
6.8	Activity Diagram.....	28

7. SYSTEM IMPLEMENTATION

7.1	Details of Implementation.....	31
7.1.1	main.py... ..	31

8. TESTING

8.1	Introduction... ..	36
8.2	Test Reports... ..	36

9. SCREENSHOTS..... 38

10.CONCLUSION41

11. FUTURE ENHANCEMENT 42

12. REFERENCES.....43

LIST OF FIGURES

Details	Page No.
Fig.No 3.1 CNN.....	14
Fig.No 3.2 Traditional Pattern Recognition.....	15
Fig.No 3.3 Fully Connected Layer	16
Fig.No 6.1 Flow of the project	25
Fig.No 6.2 DFD Level 0	26
Fig.No 6.3 DFD Level 1	26
Fig.No 6.4 Use Case Diagram	27
Fig.No 6.5 Sequence Diagram	28
Fig.No 6.6 Activity Diagram	29
Fig.No 9.1 Screenshot of Home Page	38
Fig.No 9.2 Screenshot of Overview Page	39
Fig.No 9.3 Screenshot of Prediction Page	40
Fig.No 9.4 Screenshot of Result	40

LIST OF TABLES

Details	Page No.
Table.No 8.1 Test Reports.....	37

CHAPTER 1

INTRODUCTION

1.1 Overview and Motivation

The need of an immediate and precise diagnosis can't be emphasised in the healthcare field where liver cancer cases are rising. Identification of liver abnormalities has benefited by the utilisation of diagnostic imaging procedures like computed tomography (CT) scans. However, this process often demands substantial human effort and expertise. Our project's goal is to utilise the potential of deep learning, more especially CNN, to handle this difficulty, in automating the intricate tasks of segmenting and classifying liver tumors within CT scan images. The potential ramifications of this endeavor extend to enhanced patient outcomes, optimized treatment strategies, and the overall efficiency of healthcare workflows.

1.2 Aim

The primary objective of this effort is to create a sophisticated deep learning system leveraging convolutional neural networks (also known as CNNs) that will automate the difficult process of segmenting and categorising liver cancer utilising computed tomography (CT) pictures. By leveraging the power of CNNs and transfer learning from pre-trained global models, our aim is to enhance the accuracy and efficiency of diagnosing liver tumors, thereby contributing to timely and precise medical interventions. Through the development of this automated solution, we seek to alleviate the burden on healthcare professionals, enabling them to focus on critical decision-making while improving patient outcomes and streamlining healthcare workflows in the context of liver cancer diagnosis.

1.3 Problem Statement

The core challenge at hand revolves around the laborious and time-intensive process of manually segmenting and classifying liver tumors within CT scans. Healthcare professionals face significant difficulties in meticulously identifying tumor regions and categorizing them based on type or stage. Our project seeks to tackle this challenge by developing an automated

solution that possesses the capability to precisely delineate liver tumors from CT scan images and subsequently categorize them into distinct types or stages. The intricate nature of this problem is accentuated by the complexities tied to distinguishing tumor areas from healthy tissue, stemming from variations in attributes like size, shape, texture, and location.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature review

An investigation of available sources, or literary works, on a specific topic, is known as a literature review. Educational publications, novels, and additional resources pertaining to a certain study topic are examined in the literature review. The purpose of writing a literature review is to convey to the reader basic details and concepts were produced on a topic, in addition to their drawbacks and benefits. There are four primary goals for a literature review.

- Exhibits knowledge of a body of knowledge and establishes the legitimacy of the work.
- shows how prior research ties to the current endeavour while summarising it.
- combines and integrates the knowledge on a topic.
- Demonstrates your use of others' knowledge and how the research served as a springboard for new ideas.

[1] "Performance analysis of liver tumor classification using machine learning algorithms", Munipraveena Rela, Suryakari Nagaraja Rao and Patil Ramana Reddy

The performance evaluation of liver tumour categorization using ML strategies being the main topic of this paper. It highlights the importance of medical imaging in tumor identification, diagnosis, surgical planning, and interventional therapy. The authors mention the challenges of liver tumor segmentation due to poor contrast and imprecise boundaries. They discuss via means of multiphase contrast-enhanced CT imaging for better characterization and liver disease detection abnormalities. The usage of multiple machine learning procedures, include the use of SVM and the k-nearest-neighbor technique(KNN), decision tree (DT), ensemble, and naive Bayes (NB) for liver tumor classification. The dataset used in the study consists of 68 CT images with 86 features. The performance analysis shows that the SVM classifier outperforms other classifiers in terms of accuracy, specificity, sensitivity, precision, Matthew correlation coefficient (MCC), F1-score, and kappa. The findings indicate that the SVM classifier achieves the highest accuracy, with 85% correct predictions over the total number of predictions. On the other side, DT and ensemble classifiers have the lowest accuracy for the dataset considered. The findings suggest that SVM is a promising algorithm for the

categorization of liver tumors using the assessed metrics.

[2] "Support Vector Machine based Liver Cancer Early Detection using Magnetic Resonance Images", Lei Meng, Changyun Wen, and Guoqi Li

This is dedicated to crafting a computer-aided algorithm utilizing SVM for the timely identification of liver cancer through MRI. The core of this algorithm involves a histogram-centric method for extracting pertinent data from the original MRIs. The training of the SVM classification engine is conducted using a dataset comprising 100 confirmed liver cancer cases. Through this training, the SVM achieves an impressive 86.67% accuracy in effectively categorizing instances of early-stage liver cancer. This paper not only stresses the critical importance of early liver cancer detection but also highlights MRI's potential as a robust diagnostic tool. Prior studies in the realm of computer vision and machine learning have consistently demonstrated high accuracy rates in the classification of tumors and liver-related diseases using MRI scans. This novel algorithm aims to raise the bar in diagnosing early-stage liver cancer, potentially offering invaluable support to medical professionals, including doctors and radiologists, in delivering precise diagnoses. The authors also draw attention to MRI's broader utility in pinpointing irregularities across various organs, encompassing the brain, nervous system, and solid tissues. The fusion of machine learning techniques with MRI scans has already proven successful in classifying diverse conditions like brain tumors, prostate tumors, liver metastases, and liver fibrosis. Looking ahead, the authors have plans to explore advanced features derived from texture analysis, along with sophisticated classification methods, with the overarching goal of further refining the accuracy and effectiveness of their system.

[3] "Liver Tumor Segmentation and Classification: A Systematic Review", Munipraveena Rela, Nagaraja Rao Suryakari and P Ramana Reddy

This article provides a comprehensive literature survey on various methods for the early detection of liver tumors and discusses their merits and demerits. The survey highlights the challenges in liver tumor segmentation, such as unclear tumor boundaries and variations in tumor appearance and density. Several techniques are discussed in the paper, including morphological reconstruction and watershed transform for liver extraction and lesion detection,

NS and FCM for segmentation, FCN for automatic detection and segmentation, and Fast Greedy Snakes Algorithm (FGSA) for tumor region segmentation. The research explores the application of deep learning, specifically CNN, to segment colorectal liver metastases (CLMs), resulting in notable improvements in efficiency. Furthermore, the methods outlined in the study exhibit a remarkable degree of precision in classifying and segmenting liver tumors. The literature review within the current research conducts a thorough evaluation of various methodologies and strategies for the categorization and partitioning of liver tumors, underscoring their individual strengths and drawbacks.

[4] "Liver Tumor Segmentation using Deep Learning Approach", Pallavi Bhandary, Vaibhavi More, and Renuka Nagpure

This article presents an innovative deep learning-centered strategy to tackle the dual challenges of liver tumor segmentation and classification by leveraging CT scans. The authors introduce two key architectural frameworks: ResUNet and 2D-UNet. These architectures exhibit a well-structured encode-decoder layer hierarchy. The pressing need for precise, automated analysis of CT scans for early-stage liver cancer detection is underscored, given the severity of its impact as the second leading cause of cancer-related deaths. The authors accentuate the intricate nature of liver tumor segmentation, alongside the arduous and time-intensive character of conventional techniques. Notably, the application of convolutional neural networks (CNNs) in liver tumor segmentation is mentioned, along with the potency of deep neural networks in extracting meaningful insights from complex data. The paper delves into the preprocessing stages integral to the process, such as Hounsfield windowing, normalizing images and masks, and dataset augmentation. These steps are pivotal for the extraction of liver pixels from CT scans of the abdomen. An additional focus is placed on the U-Net architecture, recognized for its commendable performance in biomedical image datasets, particularly in the context of liver and tumor segmentation. Within the scope of this work, the limitations of model-based methodologies are also acknowledged. These conventional approaches demand specific parameterization steps, consequently curbing their broader applicability.

[5] "Hepatocellular Carcinoma (HCC) Liver Cancer prediction using Machine Learning Algorithms", Sanapala Rajesh, Nurul Choudhury, and Soumen Moulik

This research article centers around the enhancement of disease prediction through the synergistic implementation of genetic algorithms and machine learning. Structured and unstructured data are harnessed to optimize accuracy. To validate the efficacy of the proposed system, SVM, Naive Bayes, and KNN algorithms are employed for evaluation. The authors put forth a novel model geared towards predicting diabetes, leveraging the power of data mining. Key to this model are the Naive Bayes and KNN algorithms, strategically chosen to heighten precision and efficacy by generating supplementary data. The study encompasses a comprehensive exploration of diverse machine learning algorithms, each applied to disease prediction. This includes the utilization of SVM, random forests, logistic regression, and decision trees. In the pursuit of optimal outcomes, the authors further investigate the strategic employment of feature extraction and dataset partitioning techniques.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Introduction

The core aim of this document, known as the Software Requirement Specification (SRS), is to present a comprehensive outline of the essential prerequisites and specificities crucial for the construction of a software system. This system is tailor-made to execute liver cancer classification through the utilization of Convolutional Neural Network (CNN) algorithms. The central objective here is to thoroughly analyze medical images of the liver and precisely categorize them into two groups: normal or indicative of cancerous conditions. Contained within this document is a holistic overview of the entire project, encompassing its scope, functional attributes, as well as the non-functional imperatives that need to be fulfilled.

3.2 Purpose

The central objective in creating the Software Requirements Specification (SRS) is to outline the unique functional and non-functional intricacies necessary in order to create a console application. This comprehensive The application has been meticulously built to offer users the flexibility to uncover the most efficient and streamlined paths. To encapsulate, the principal aim of this SRS document is to demonstrate holistic overview of our software product, encapsulating its well-defined parameters and overarching objectives. Within the scope of this document, we set out on an exhaustive exploration of the precise target demographic that the project caters to, the user interface it offers, and the critical hardware and software prerequisites it necessitates. This document effectively captures the distinct perspectives of our esteemed client, the dedicated development team, and the broader audience, shedding light on how they perceive the product and its multifaceted functionalities.

3.3 Scope

This framework's purpose is to give a feedback on information-digging techniques applied to the expectation of user details for Liver Cancer detection. The framework makes it clear that information mining strategy.

3.4 Functional Requirements

A functional requirement describes the function of a system or component. A role is described as a grouping of inputs, operations, and outcomes. Functional needs include things like calculations, technical information, data processing, and manipulation.

The system uses the following methodologies:

Image Input: The system should be able to accept CT scan images as input for segmentation and classification.

Preprocessing: Automatically preprocess the input images, including resizing, normalization, and any other relevant transformations required for effective model training and prediction.

Tumor Segmentation: Implement a CNN-based segmentation module that can accurately outline liver tumors in the input CT scan images.

Tumor Classification: Develop a CNN-based classification module capable of categorizing segmented tumors.

Transfer Learning Integration: Incorporate pre-trained CNN models for transfer learning, allowing the system to leverage knowledge from broader datasets.

Model Training: Train the segmentation and classification models on labeled datasets, optimizing for high accuracy and generalization.

Validation and Testing: Provide mechanisms to validate the models using validation datasets and test their performance on unseen data.

Visualization: Generate visual outputs that display segmented tumor regions overlaid on the original CT scan images.

User Interface: Create a user-friendly interface that enables users to upload CT scan images, initiate segmentation and classification, and view results.

3.5 Non-Functional Requirements

A non-functional requirement establishes the criteria for assessing the performance of a system, differing from the specific behaviors outlined in functional requirements. This stands in opposition to functional requirements that precisely detail functions or actions. The comprehensive approach for implementing functional requirements is laid out during the system design phase. Conversely, the strategy for meeting non-functional requirements is intricately elaborated within the system architecture, primarily due to their classification as

Architecturally Significant Requirements. Non-functional requirements include:

Performance:

- The system's processing speed is of paramount importance. It should swiftly process and generate results, ensuring that medical professionals receive prompt insights for timely decision-making.
- Latency in the segmentation and classification processes will be minimized to enable real-time or near-real-time analysis, crucial for immediate medical interventions.
- The system's responsiveness will remain consistent even under varying workloads, guaranteeing that users experience smooth performance regardless of concurrent usage.

Accuracy and Reliability:

- A high degree of precision in the segmentation model's results will be achieved, as measured by the Dice coefficient. This will provide medical experts with accurate tumor region delineation for effective diagnosis.
- The classification model will exhibit a robust accuracy rate, ensuring that the system can reliably differentiate between different tumor types and stages, thus aiding in accurate treatment planning.
- The system's reliability will be validated across diverse datasets, reflecting its consistent performance in various clinical scenarios, enhancing medical professionals' trust in its outcomes.

Scalability:

- The system's architecture will be designed to accommodate an expanding user base and a growing volume of medical images, allowing healthcare facilities to seamlessly scale their operations without compromising efficiency.
- As the size of the dataset or the complexity of the model increases, the system's performance will remain stable, ensuring that the accuracy and speed of analysis are preserved.

Usability and User Experience:

- The user interface will be thoughtfully designed, considering the user's perspective and minimizing the learning curve. This approach will empower medical professionals to

effortlessly navigate the system.

- Intuitive visual representations of classification results will be provided, enabling medical experts to quickly interpret and make informed decisions based on the presented data.

Security and Privacy:

- Adherence to stringent data privacy regulations such as HIPAA will ensure that sensitive patient data is rigorously protected throughout the system's operation.
- Robust authentication and authorization mechanisms will restrict access to authorized personnel only, guaranteeing that patient information remains confidential and secure.

Interoperability:

- The system will seamlessly integrate with prevalent medical imaging formats, ensuring that healthcare facilities can leverage their existing infrastructure without the need for complex adaptations.
- By fostering compatibility with existing healthcare systems, the system will facilitate smooth data exchange and collaboration across different tools and platforms.

Maintainability:

- Comprehensive code documentation and modular design practices will make future updates and maintenance tasks efficient and straightforward, reducing downtime and potential disruptions.
- Model updates and improvements will be seamlessly integrated into the system's architecture, ensuring that advancements in medical imaging analysis can be harnessed without hindrance.

3.6 Tools and technologies used

3.6.1 Python

The Python is a well-understood, very practical programming platform. Guido van Rossum's plan reasoning in Python and released for the first time in 1991, emphasises the significance of the code by the frequent utilise vital whitespace. The language's advancements and an object-

oriented methodology are created to help software professionals write clear, consistent code for projects of all sizes.

Python is constructed piece by piece, and trash is gathered. Numerous programming standards, such as procedural, object-oriented, and practical programming, are supported by it. The vast standard library of Python has produced frequent claims that it is a "batteries included" language. Computer language Python has several different worldviews.. A substantial a few of its characteristics assist helpful programming and angle situated programming (counting by metaprogramming and metaobjects (enchantment techniques)). Article organised programming and organised authoring computer programmes are fully supported. Expansions uphold a range of principles, such as reason programming and plan by agreement.

3.6.2 Anaconda

The programming languages R and Python are available through Anaconda, which is free and open-source. The Python interpreter, and other bundles related to AI and information science are distributed together with them. The main goal of Anaconda is to simplify the introduction of all (or most) of the necessary bundles with a single establishment for people interested in those sectors. Conda is an open-source software programme framework for creating and introducing bundles and conditions for executives, and it makes it simple to do both. AI libraries, such as Theano, Scikit-Learn, and TensorFlow. Pandas, NumPy, and Dask are examples of information science libraries. Bokeh, Datashader, Matplotlib, and Holoviews are examples of perception libraries. Live code, visuals, and text are all combined in Jupyter Notebook, a sharing notepad.

3.6.3 NumPy

The main Python bundle supporting logical registering was NumPy. This additionally includes, between other things as well:

- an amazing N-dimensional cluster object
- sophisticated (broadcasting) capacities
- tools for integrating Fortran and C/C++ programming
- Straight polynomial maths, Fourier change, and arbitrary number knowledge are all useful skills.

3.6.4 Pandas

- Pandas stands out as an open-source library, operating under the BSD license, and it delivers high-performance data structures and user-friendly data analysis tools meticulously crafted for Python programming.
- An additional significant facet is Pandas' association with Num FOCUS, which extends support to enhance the growth of pandas as a prominent open-source initiative and facilitates opportunities for contributing to the project's growth.

3.6.5 OpenCV

OpenCV, recognized as the Open-Source Computer Vision Library, is a preeminent open-source software library acclaimed for its focus on computer vision and ML applications. The fundamental objective that drove the development of OpenCV was the creation of cohesive platform catering to a variety of computer vision applications, thereby facilitating the seamless integration of machine perception into commercial products. A distinctive advantage is underscored by OpenCV's adoption of the BSD license, endowing enterprises with the liberty to utilize and adapt the code according to their specific requirements.

At the nucleus of this library resides an extensive assemblage of over 2500 meticulously optimized algorithms, spanning both conventional and cutting-edge computer vision and machine learning domains. These algorithms fulfill diverse roles, encompassing facial detection and recognition, object identification, human action classification within videos, monitoring camera movements, tracking object motion, constructing 3D models of objects, generating 3D point clouds through stereo cameras, merging images to craft panoramic vistas, facilitating content-based image retrieval, mitigating red-eye effects in flash photography, tracking ocular movements, recognizing landscapes, and establishing markers to elevate augmented reality experiences, among an array of other applications.

With an engaged user community exceeding 47 thousand and an estimated download count surpassing 18 million, OpenCV transcends the realm of a conventional software library. It stands as a dynamic hub of innovation, its influence extending across corporate sectors, research enclaves, and governmental entities, underscoring its far-reaching impact and significance.

3.6.6 Flask

The little web application Flask was made with Python. It is provided with a microframework because no specialised tools or libraries are required. It is deficient in the knowledge basic layer of deliberation, structural approval, and any other areas where earlier external libraries give the anticipated capabilities. In any case, Flask allows for use with extensions to support application includes much like Flask would. Improvements are provided regarding object-social mappers, structure approvals, transfer dealing with several open confirmation improvements, and several basic system-related devices. A lot less regularly than augmentations, the central Flask software is updated.

3.6.7 Deep Learning Overview

Deep Learning, commonly referred to as Deep Neural Networks, encompasses a class of Artificial Neural Networks (ANNs) characterized by their multi-layer structure. Over the past decades, it has emerged as a potent tool, gaining considerable prominence within literature due to its remarkable ability to handle vast and complex datasets. More recently, the appeal of incorporating deeper hidden layers has outshined conventional methods, particularly in fields like pattern recognition.

Within the realm of deep neural networks, one prominent variant is CNN. The term "Convolutional" originates from the mathematical operation known as convolution, involving the interaction of matrices. A CNN consists of diverse layers, including the convolutional layer, non-linearity layer, pooling layer, and fully connected layer. Notably, while convolutional and fully connected layers possess parameters, pooling and non-linearity layers operate without them. Demonstrating exceptional performance in ML scenarios, CNNs find particular favor in applications dealing with image data. tasks like image categorization fall under this on expansive datasets such as ImageNet, computer vision challenges, and natural language processing endeavors, with consistently astonishing outcomes.

This paper is devoted to providing comprehensive insights into the intricacies of CNNs, elucidating crucial elements and critical aspects of their operation. Moreover, the paper will expound upon the parameters that wield influence over the efficiency of CNNs. It's assumed that readers possess a solid foundation in both ML and artificial neural networks.

Convolutional Neural Networks have made monumental strides over the past decade, achieving breakthroughs across diverse domains related to pattern recognition. These span from image

processing, where CNNs excel in tasks like object detection, to voice recognition, revolutionizing speech-to-text technologies. Among the most transformative feats of CNNs is their capacity to substantially reduce the quantity of parameters within ANNs. This pivotal advancement has propelled researchers and developers towards embracing larger models, unlocking the potential to tackle complex tasks that were hitherto beyond the reach of classical ANNs.

A foundational premise that underpins problems tackled by CNNs is their insensitivity to spatially dependent features. For instance, in applications such as medical image analysis, the exact location of anomalies might be less crucial than their accurate detection. This emphasizes the network's ability to focus on abstract patterns regardless of their placement in the input data. Additionally, CNNs are revered for their capability to extract increasingly abstract and complex features as data propagates through their deeper layers. This hierarchical feature extraction is a cornerstone of their effectiveness in a wide array of tasks, contributing to their remarkable success.

3.6.8 CNN

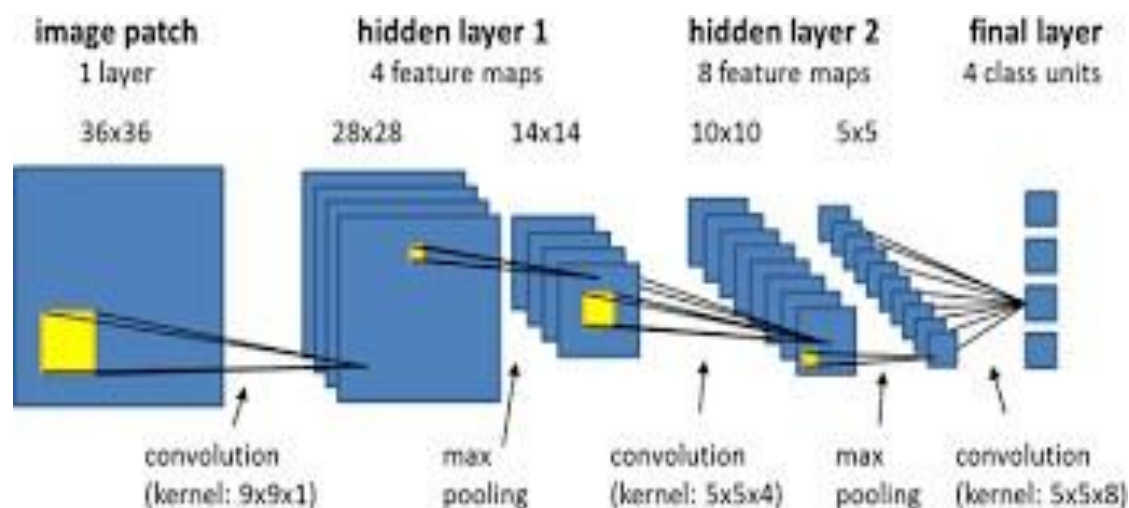


Fig.No 3.1: CNN

Convolutional Neural Networks has the following layers:

- Convolutional
- ReLU Layer
- Pooling
- Fully Connected Layer

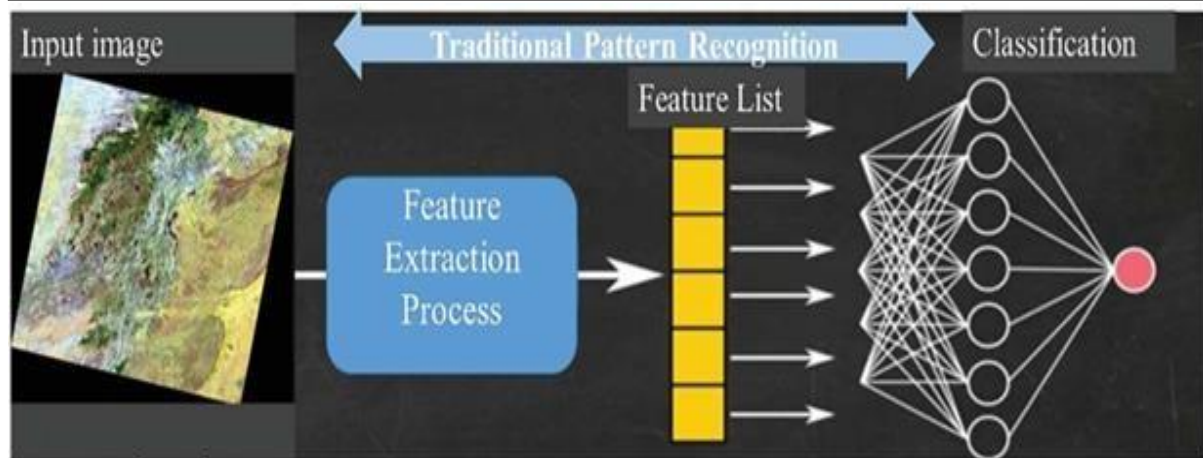


Fig.No 3.2: Traditional Pattern Recognition

Convolution Layer

In the initial phase of extracting information from an input image, convolution takes the lead. This convolutional process adeptly upholds pixel connections while assimilating visual features from small data patches. Facilitating this mathematical operation requires the integration of two essential inputs: the image matrix itself and a filter, colloquially known as a kernel.

ReLU Layer

In this layer, we eliminate every negative value from the filtered photos and replaces them with zeros. To prevent the values from adding up to zero, it is happening. Rectified Linear Unit (ReLU) transform functions only activate a node if the input value is higher than a specific threshold. When the information rises above a threshold, the output changes from zero while the data is below zero. It and the dependent variable are related linearly.

Pooling Layer

In cases where the images exhibit considerable size, the layer pooling segment plays a pivotal role in constraining the parameter count. Spatial pooling, alternatively referred to as subsampling or downsampling, effectively reduces the dimensionality of each map while safeguarding essential information. Spatial pooling manifests in diverse forms, some of which encompass:

- Max Pooling
- Average Pooling
- Sum Pooling

Incorporating max pooling entails the identification of the utmost element within the rectified feature map. Yet, it's important to acknowledge the availability of alternatives beyond exclusively selecting the largest element. For instance, contemplating the option of average pooling involves calculating the mean of the elements. Alternatively, the concept of sum pooling arises, encompassing the accumulation of all constituent components within the feature map.

Fully-Connected Layer

A (typically) low-cost method of learning non-linear combinations of the high-level attributes represented by the outcome of the convolutional layer is to add a Fully-Connected layer. In that location, the Fully-Connected layer is now learning a function that might not be linear. A CNN network picture shown below Fig.No 3.3

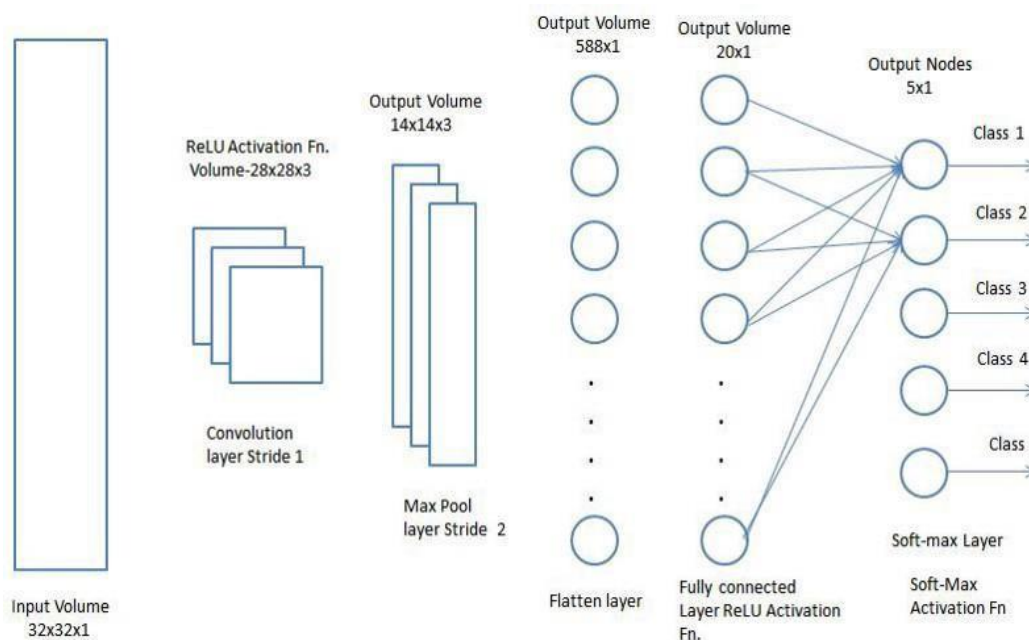


Fig.No 3.3: Fully-Connected Layer

3.7 Software and Hardware Requirements

3.7.1 Software Requirements

- IDE : FLASK
- Language : Python
- Tool : Jupyter Notebook
- Software : Anaconda
- Front End : HTML, CSS
- Libraries : Tensorflow, Keras, NumPy, Pandas

3.7.2 Hardware Requirements

- RAM : 2 GB
- Hard disk : 100 GB
- Process : 32/64 Pentium

CHAPTER 4

METHODOLOGY

Dataset Acquisition:

- The initial step involves the acquisition of the dataset required for conducting the predictive analysis. To achieve this, we obtain the dataset from a trusted source, specifically kaggle.com. This platform offers a diverse range of datasets suitable for various data analysis projects.
- For our purpose, we select a dataset that includes instances of liver disease and is structured into two distinct classes.

Data Pre-Processing:

- Once the dataset is sourced, it undergoes a crucial phase known as data pre-processing. This stage aims to enhance the quality of the dataset and prepare it for effective model training. Various image pre-processing techniques are applied to the selected dataset. These techniques serve to optimize the dataset's features and characteristics before feeding it into the model.
- Among the techniques employed, image resizing plays a significant role. By resizing the images, we ensure that all the data points have a consistent size, which is a fundamental requirement for training neural networks effectively. This step minimizes discrepancies and inconsistencies that might arise from varying image dimensions.
- Additionally, the dataset is divided into separate subsets to facilitate the model training and evaluation process. The data is divided into two subsets: one designated for training purposes and another reserved for testing the model's performance. This separation allows us to assess the model's ability to generalize its predictions to unseen data effectively.

Data Modeling:

- The core of our methodology revolves around data modeling using a Convolutional Neural Network (CNN) algorithm. CNNs are particularly well-suited for image-related tasks due to their ability to recognize spatial patterns within images. In this phase, the training subset of the dataset is utilized to train the CNN algorithm.
- During training, the algorithm learns to identify features and patterns present in the images that differentiate between the two classes of liver disease. This learning process involves adjusting the parameters of the network to minimize prediction errors.
- After the training phase, the model's effectiveness is assessed using the testing subset. This evaluation involves feeding the testing images through the trained model and comparing the predicted outcomes to the actual labels. This process generates accuracy metrics, such as precision, recall, and F1-score, that offer insights into the model's performance.

Model Building:

- Upon achieving a satisfactory level of accuracy during model evaluation, the final step involves building and saving the model. If the trained model demonstrates a high accuracy rate and generalizes well to unseen data, it is considered reliable for making predictions in real-world scenarios.
- The model, which has learned the distinguishing features between different classes of liver disease, is saved as a model file. This file encapsulates the network architecture, learned parameters, and patterns discovered during training.
- It serves as the foundation for future predictions and analysis, providing a ready-to-use tool for identifying and classifying liver disease based on medical images.

CHAPTER 5

SYSTEM ANALYSIS

5.1 Existing System

Health care industries today offer a number of advantages, including the ability to detect health insurance fraud, the availability of medical facilities at affordable rates, the discovery of smarter treatment methodologies, the development of effective healthcare policies, effective hospital resource management, better customer relations, improved patient care, and hospital infection control. One of the important fields of medical study is the diagnosis of liver cancer. For the prediction of liver cancer, there is no automation.

The performance of these aforementioned strategies was compared in the current system using ML techniques including Decision Tree, K-Nearest Neighbour, and Naive Bayes Classifier considering their execution times, but the results were inaccurate.

5.1.1 Drawbacks of Existing System

- **Manual Segmentation:** Relying on manual segmentation is time-consuming, subject to inter-observer variability, and might not be feasible for large datasets.
- **Limited Efficiency:** Existing automated methods might lack the accuracy needed for reliable diagnosis, leading to potential misinterpretation of results.
- **Complex Features:** Traditional feature engineering approaches might struggle to capture intricate patterns and variations present in liver tumors.
- **Scalability:** Scalability issues could arise when dealing with a large volume of data, potentially hindering the adoption of such methods in clinical practice.

5.2 Proposed System

The proposed approach involves an automated prediction system for Liver Cancer using advanced deep learning methods. The objective of this technique is to introduce a comprehensive deep learning framework employing a CNN architecture, thereby conducting a comprehensive analysis for the purpose of image classification related to Liver Cancer. This streamlined deep learning methodology is poised to offer valuable insights into the neural

intricacies connected with autistic children, while also contributing to the timely detection of early-stage Liver Cancer among pediatric patients.

We have taken some initial results on detecting Liver Cancer from Brain X-Rays using a deep-learning model. We needed demonstrated significant improvement in performance over Liver Cancer -Net, the only publicly maintained tool for classification of LIVER CANCER X-rays, on the same Brain-Xray-dataset. The results look promising, though the size of the publicly available dataset is small. We plan to further verify our strategy utilising larger LIVER CANCER-ray image datasets and clinical trials.

5.2.1 Advantages of proposed system

- Automation: Our model eliminates the need for manual segmentation, reducing the time and effort required for diagnosis.
- Accuracy: By leveraging deep learning techniques, our model has the potential to achieve higher accuracy and consistency in tumor segmentation and classification.
- Knowledge Transfer: Utilizing existing global models serves to facilitate the transference of insights from expansive image datasets, thereby enriching the model's adeptness in detecting nuanced features.
- Real-time Results: Automated segmentation and classification facilitate real-time diagnosis, enabling prompt medical decision-making.
- Scalability: Our model is capable of handling large volumes of data efficiently, making it suitable for use in clinical settings.
- Objective Results: Our model's results are less prone to inter-observer variability, providing objective and reproducible outcomes.
- Potential for Early Detection: Accurate and timely detection of liver tumors can potentially lead to earlier interventions and improved patient outcomes.

5.3 Feasibility Study

The feasibility analysis assumes a crucial role in pinpointing and resolving the intrinsic challenges embedded within the project. It presents remedies that sketch out the structure and attributes of the proposed novel system.

Technical Feasibility:

The project "Liver Cancer Segmentation and Classification" is deemed technically feasible due to the following key attributes:

- **Utilization of Python:** The project is built using the Python programming language, which is renowned for its versatility and suitability for various application domains. Python's extensive libraries and frameworks aid in efficient code development.
- **Local Server Implementation:** The "Liver Cancer Segmentation and Classification" website is hosted on a local server. This approach ensures control over the hosting environment and facilitates seamless interaction with the website.
- **Expert Coordination:** The design and code components are intricately coordinated within the local server environment. This coordination enhances the project's organization and readability, contributing to effective collaboration among team members.
- **Python Console Application:** The project offers a console application for designing and configuring the application. This not only streamlines the development process but also provides a user-friendly interface for application customization.
- **Interoperability, Availability, and Reliability:** The project is designed to exhibit high levels of interoperability, making it compatible with various systems and platforms.
- Additionally, the local server ensures the availability of the website. Moreover, the project's technical design prioritizes reliability to ensure consistent and accurate results.

Economical Feasibility:

The "Liver Cancer Segmentation and Classification" system is economically feasible due to the following reasons:

- **Open-Source Software:** The system's construction is based on open-source software tools.
- This choice eliminates the need for expensive proprietary software licenses, reducing development costs significantly.

Operational Feasibility:

The operational feasibility of the "Liver Cancer Segmentation and Classification" project is substantiated by the following elements:

- **Algorithmic Predictions:** The system leverages historical data to forecast results of CT scans. The algorithms employed quickly process input data and generate predictions, making the system operationally efficient.
- **Performance Comparison:** The project employs a systematic approach to compare the execution speed of each technique. Through graphical representation, the project demonstrates the relative performance of different techniques, aiding in operational decision-making.

Behavioral Feasibility:

The project titled "Liver Cancer Segmentation and Classification" is classified as behavioral feasibility due to the following rationale:

- **Objective Alignment:** The project is in harmony with its established objectives upon completion and implementation.
- The system effectively fulfills the intended purpose of identifying and classifying liver cancer, showcasing its behavioral alignment.

CHAPTER 6

SYSTEM DESIGN

6.1 Introduction to Design document

The Software Design phase acts as a crucial compass for Android application development, offering intricate guidance on the construction process. This encompassing Software Design entails both descriptive and graphical documentation, effectively charting out the software's architecture for the project. This documentation encompasses a number of elements, including employ sequence diagrams, case study models, and additional supporting details that reinforce the project's requirements.

6.2 Purpose

The system design phase lays the foundation for successful implementation. It defines the architecture, components, and interactions within the system. Through careful design, we ensure that each module functions cohesively, facilitating smooth data flow and user interaction. This phase acts as a blueprint, guiding developers when designing a functional and user-friendly application.

6.3 Scope

This software design document is for an essential system that will serve as a proof-of-concept for the creation of a system that offers an essential level of capability to demonstrate viability for usage in large-scale production. The design document for the programme puts a heavy emphasis on document generation and maintenance. The primary element of the system— In conjunction with various other current systems, a document's interaction facet—which abstracts separate handling of content properties and document interactions—will be used. Presented within this text are the design principles for intelligent translation.

6.4 Flow of the project

We discuss the proposal framework's stream layout, because it illustrates how the project begins with the dataset collecting. Collected Kaggle datasets of liver tumor and non-tumor.

Pre-processing of acquired datasets chooses characteristics that aid in identifying the liver cancer. Techniques for data preparation are utilised. to preprocess the dataset. Using the OpenCV library, all of the collected images are scaled to a specific size. Dataset is divided into two sections: one for both education and the other for testing. The CNN method serves to instruct the chosen datasets. The correctness of the training datasets is evaluated. Fig.No 4.1 depicts the project's flow.

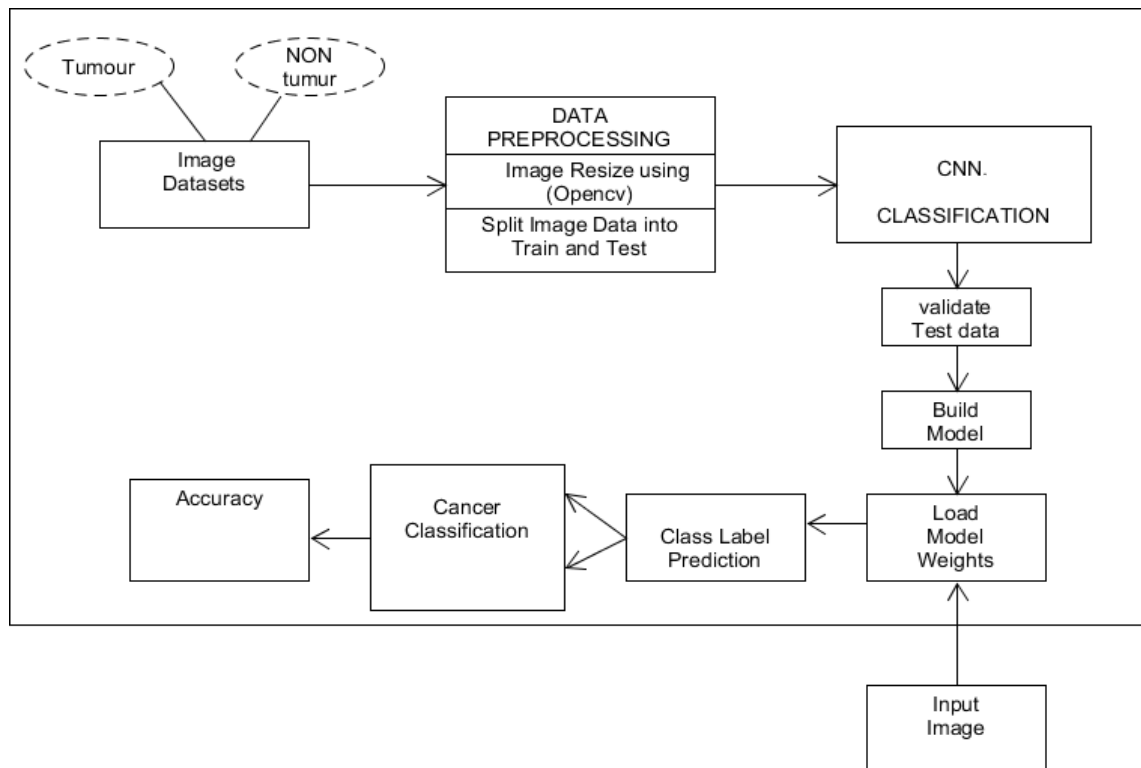


Fig.No 6.1: Flow of the project

6.5 Dataflow Diagram

A Data Flow Diagram (DFD) functions as a graphic illustration that tracks the movement of information within a designated process or system. Employing specific symbols like rectangles, circles, and arrows, complemented by concise text labels, it elucidates the course of data inputs, outputs, storage points, and the interconnections among them. The landscape of data flowcharts spans a continuum, ranging from fundamental hand-drawn process overviews to intricate multi-tiered DFDs that progressively delve into the intricacies of data manipulation. These visual aids serve a dual purpose: dissecting established systems and charting the path for new designs. Comparable to impactful visual tools, DFDs possess an inherent ability to communicate intricate concepts that might otherwise be challenging to convey solely through

text. Moreover, these visual representations effectively cater to both technical and non-technical audiences, bridging the comprehension divide from developers to high-level executives.

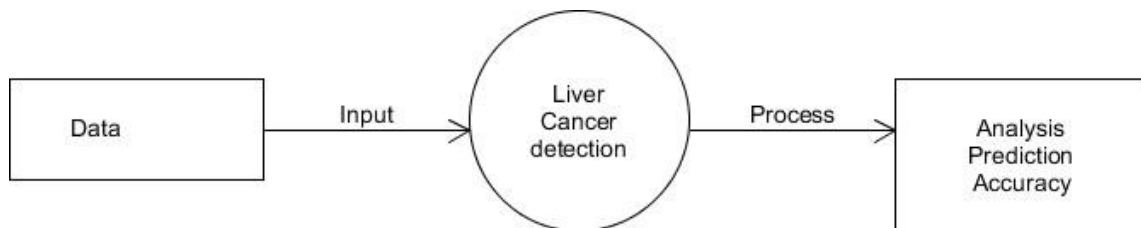


Fig.No 6.2: DFD Level 0

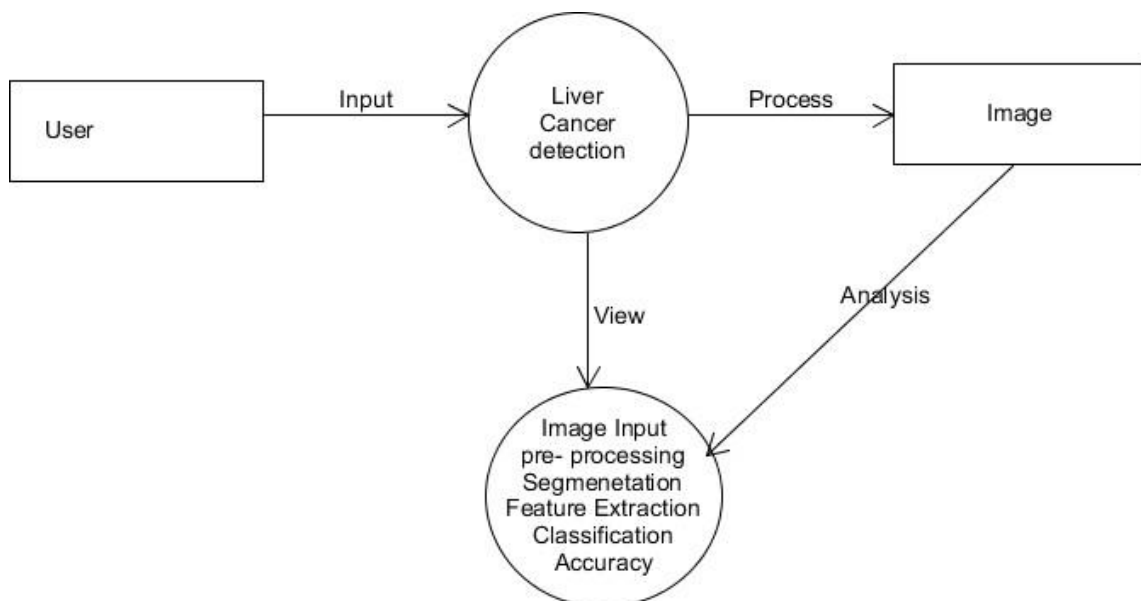


Fig.No 6.3: DFD Level 1

6.6 Use Case Diagram

The main tool when representing the system and software necessities of a developing software programme is a UML use case diagrams. These use cases describe anticipated behaviours (the "what") without getting into specific implementation mechanics (the "how"). Use cases can be presented verbally or graphically after they are defined, frequently using the format of a use case diagram. The primary idea of use case modelling is to influence the layout of systems with the viewpoint of the end user. This methodology explicitly defines all externally observable system operations in order to effectively convey system behaviour in language that users can understand.

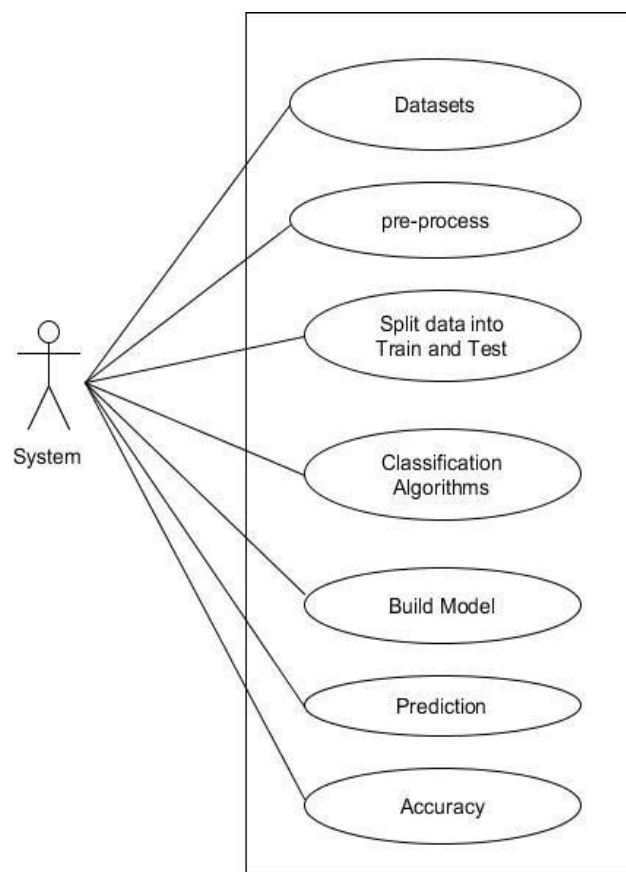


Fig.No 6.4: Use Case Diagram

6.7 Sequence Diagram

The sequence diagram functions as a connection diagram, visually illustrating the extensive connections between numerous elements inside a given sequence. The diagram below successfully illustrates the temporal procedure via depicting object interactions in a time-ordered manner. It reveals the key classes and objects pertinent to the situation and includes the messages that must be sent and received for smooth object activities. Notably, sequence diagrams and use case realisations commonly cross each other, acting as a bridge inside the system's logical framework. These diagrams might also be described as scenario diagrams or events scenarios. A sequence diagram is a useful tool for visualizing and verifying different runtime scenarios. These can aid in the prediction of system behaviour and the identification of potential class responsibilities during the modelling of innovative systems. Fig.No 4.5 shows the flowchart.

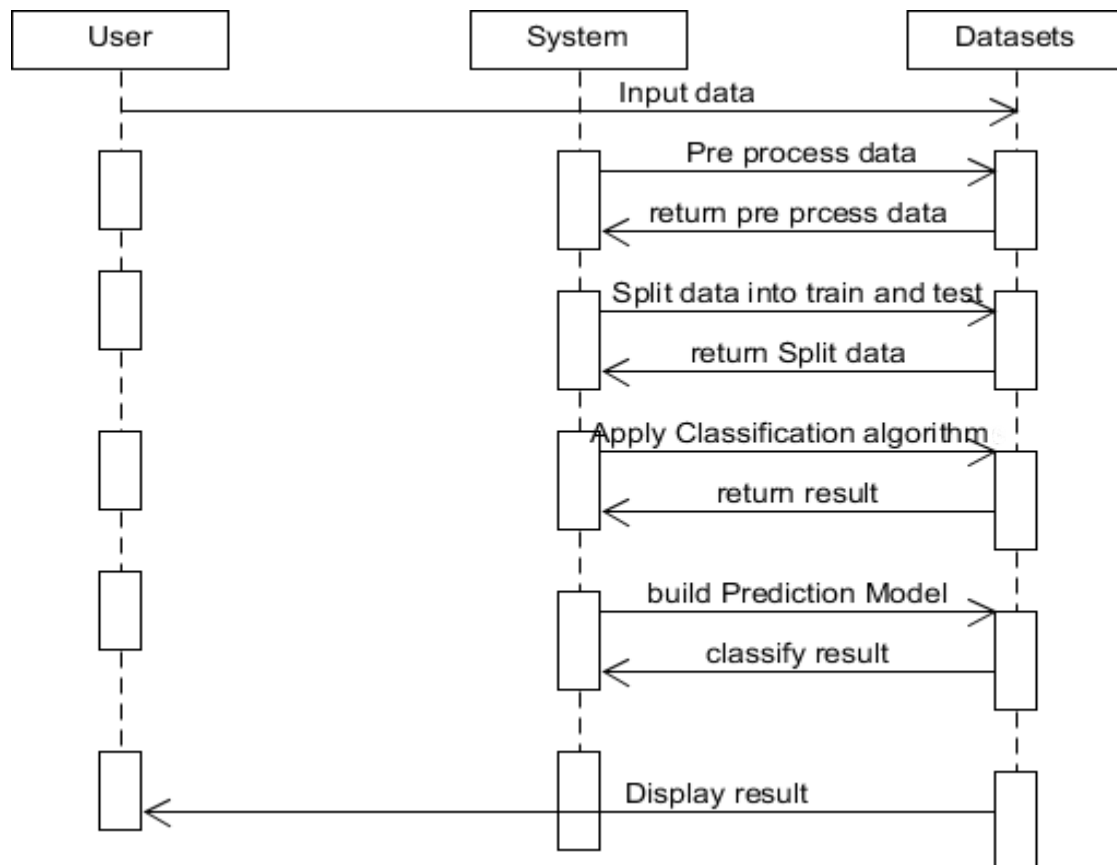
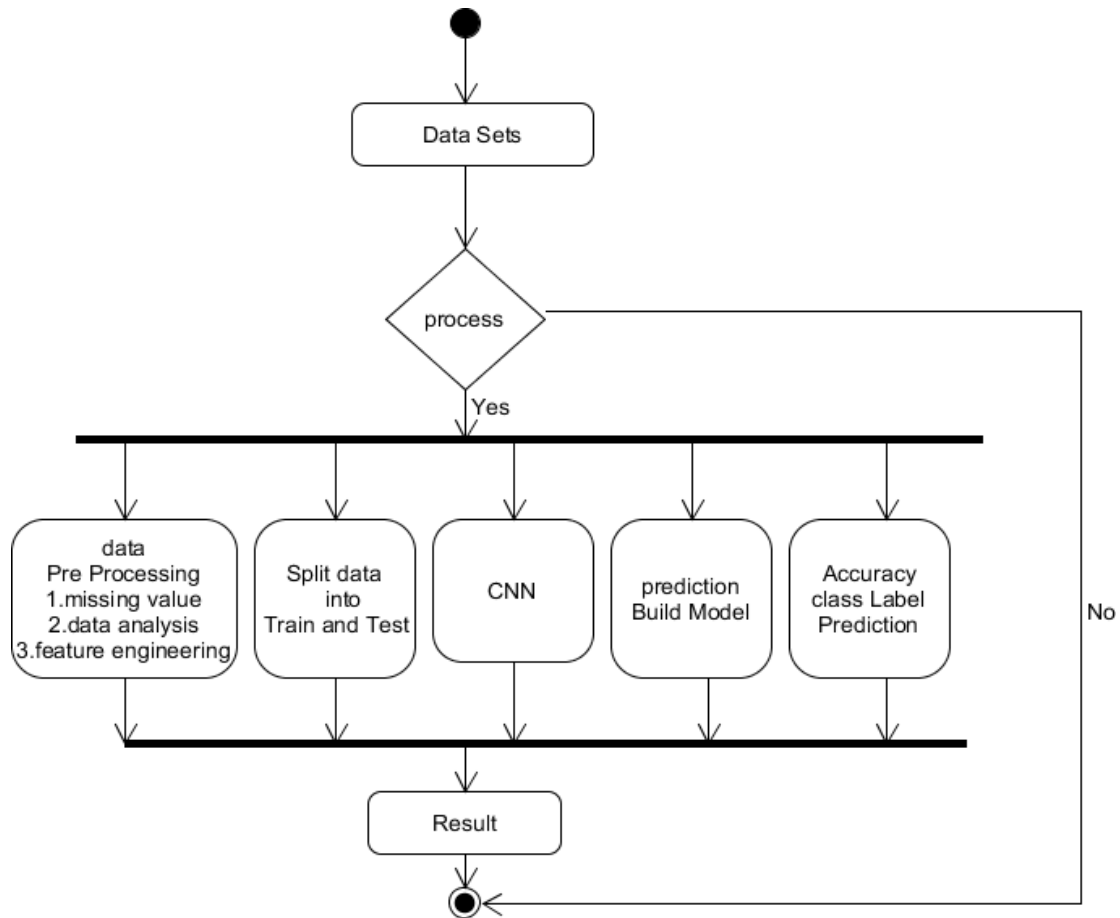


Fig.No 6.5: Sequence Diagram

6.8 Activity Diagram

Using several levels for the realm of abstraction, activity diagrams demonstrate the planning of operations to produce a service. Usually, an event needs to be accomplished by a few operations, particularly if the operation is meant to accomplish multiple things that call for coordination, or the method that the events in a single instance of use relate from one another, especially in use cases where activities can intersect and call for coordination. The coordination of a set of use cases to reflect business processes may also be modelled using this-technique.

**Fig.No 6.6: Activity Diagram**

CHAPTER 7

SYSTEM IMPLEMENTATION

The project's implementation leverages Python, a programming language that seamlessly combines both object-oriented and procedural paradigms. Object-oriented programming, a distinctive approach employed here, involves creating segregated memory segments for data and functions. These segments serve as templates, readily adaptable to generate additional module instances as required. This method emerges as an effective means of modularizing programs, enhancing their overall structure and manageability.

The Python programming dialect is employed in the implementation of this task. Python has automatic garbage collection and dynamic typing. It enables functional, procedural, and object-oriented programming paradigms, among a wide variety of others. Python, which is frequently praised for having a "batteries included" mindset, has a sizable standardised libraries which provide an extensive list of features.

The target audience's enjoyment, system functionality, and installation of the software product in its final, intended state environment are all regarded as characteristics of software implementation. The programme is supposed to facilitate their work, but the individuals are unsure of this.

- The active user must comprehend the advantages of the system.
- They had more faith in the plan.
- The user is given the proper training to make him feel at ease with the application.

Before examining the system, the user has to understand that the server programme must be running in order to retrieve the outcomes, on the server. The real procedures won't happen if the server object isn't executing on the server.

7.1 Details of Implementation

7.1.1 main.py

Importing Libraries

Importing essential libraries and modules for building a web application and performing image analysis tasks. Flask is employed to create web pages and manage user interactions, while various libraries such as NumPy, OpenCV (cv2), and Matplotlib are used for numerical computations, image manipulation, and visualization. TensorFlow and Keras are utilized to construct and train NN models for machine learning applications. Lastly, the 'secure_filename' function from Werkzeug ensures safe handling of uploaded files by preventing potentially malicious filenames from causing security vulnerabilities. This comprehensive set of imports equips the application with a range of capabilities, from web interface development to machine learning-based image analysis.

Flask Setup

In this, code segment configures key settings and variables for the Flask application. The 'UPLOAD_FOLDER' variable specifies the folder in which uploaded files will be kept within the 'static' folder of the project. The 'app' instance of the Flask class is created. The 'config' attribute of the 'app' instance is used to set 'UPLOAD_FOLDER' and 'MAX_CONTENT_LENGTH', which restricts the maximum file upload size to 16 megabytes for security reasons. The 'secret_key' attribute is assigned a value for enhancing the security of session management. The 'ALLOWED_EXTENSIONS' set defines a list of acceptable file extensions for uploaded files. The 'class_names' list contains labels for classes in a classification task, specifically 'non-cancerous' and 'cancerous'. The 'img_height' and 'img_width' variables set the dimensions for input images in this model. The 'allowed_file' function checks whether a filename has a valid extension by examining the last part of the filename after the dot and comparing it against the allowed extensions list. This code thus establishes essential configuration parameters and helper functions for the Flask application, enabling file uploads, security measures, and image-related tasks.

Login Credentials

In this, code defines a route in the Flask application that handles both GET and POST requests at the root URL ("/"). The 'login' function is associated with this route. Inside the function, the 'msg' variable is initialized to an empty string. The function first to see if the request method is POST and if 'username' and 'password' fields which exist in the submitted form data. If these conditions are met, the 'username' and 'password' values are extracted from the form data.

The next subsequent block of code verifies if the provided 'username' and 'password' match the values 'admin' and 'admin'. If the conditions are satisfied, the 'loggedin' key is set to 'True' in the 'session' object, and additional 'id' and 'username' keys are assigned specific values. The function then renders the 'index.html' template, which typically serves as the home page after successful login. Alternatively, the commented-out line using 'redirect' and 'url_for' could be utilised to redirect to another route, such as 'upload_image'.

In case the provided 'username' and 'password' fail to match the expected values, the 'msg' variable is set to 'Incorrect username/password!'. Finally, if the request is not a POST request or if the login credentials are incorrect, the 'login.html' template is rendered with the 'msg' variable. This code implements a basic user authentication mechanism, utilizing session management and rendering templates to handle login attempts and display appropriate messages to users.

Files

In this section, code defines three separate routes within the Flask application:

Home Route ('/home'):

The 'home' route is established using the `@app.route('/home')`. This route is designed to handle requests made to the URL '/home'. Inside the 'home' function, the code calls the 'render_template' function to render the 'index.html' template. This template is typically used as the home page of the application. The primary purpose of this route is to render the home page, allowing users to access certain content.

About Route ('/about'):

The 'about' route is defined using the `@app.route('/about')`. This route is intended to handle

requests directed to the URL '/about'. Within the 'about' function, the code renders the 'about.html' template. This template includes details regarding the application.

Home1 Route ('/home1'):

The 'home1' route is created using the `@app.route('/home1')`. Similar to the 'about' route, the code renders the 'home.html' template. This route provides users access to a page identified by 'home1'.

Uploading Images

In this, code defines a Flask route '/upload_image' that handles POST requests from the user, typically when attempting to upload an image. The 'upload_image' function is associated with this route.

First, the code to see if the key 'file' resides in the uploaded files section of the request. If not, a flash message is displayed to the person indicating that no file was provided, and he/she is redirected back to the same URL.

Next, the code retrieves the uploaded file using the 'file' key from the uploaded files. It then to see if the filename of the uploaded file is an empty string, which suggests that the person did not select an image for uploading. Given this, a flash message is shown, indicating that no image was selected, and he/she is redirected to the same URL.

If the uploaded file is present and has a valid filename (checked using the 'allowed_file' function), the code generates a secure filename using the 'secure_filename' function. It then saves the file to the specified directory within the 'UPLOAD_FOLDER' using the 'file.save' method. The 'path' variable is created by joining the 'UPLOAD_FOLDER' with the generated secure filename, and it is printed to the console for debugging purposes.

This code thus handles the uploading of images, ensuring that the uploaded file is present, has a valid filename, and is securely saved to the designated directory. It furthermore offers user comments through flash messages if there is missing files or incorrect file extensions.

CNN Implementation

In this, loads a pre-trained model of a NN for forecasting liver cancer from medical images. First, 'num_classes' is set to 2, representing the number of classes in the classification task: 'non-cancerous' and 'cancerous'. The 'model' is constructed as a sequential neural network, including layers for data preprocessing, convolution, pooling, flattening, and dense connections. The model's architecture is defined with specific layer configurations and activation functions.

The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss for training, with accuracy as a metric. Pre-trained weights are loaded from the 'liver_cancer.h5' file. The 'test_data_path' is set to the path of the uploaded image, and the image is loaded, converted to an array, and expanded as a batch.

The loaded model is then used to predict the image's class probabilities using the 'predict' function. The 'score' variable is computed as the softmax transformation of the predictions, representing the confidence scores for each class. The printed output displays the class with the highest confidence and the corresponding percentage confidence score.

Finally, the code renders the 'result.html' template, passing relevant data such as the filename of the uploaded image, the predicted class label, the highest confidence score, and a result indicator (res=1) to be displayed to the user. This completes the process of uploading an image, processing it through the trained model, and presenting the prediction results to the user through the template.

Displaying Image

In this, code segment defines a Flask route '/display/<filename>' that handles requests to display an uploaded image. The 'display_image' function is associated with this route. When a request is made to this route with a specific 'filename' parameter, the function uses the 'redirect' function along with 'url_for' to construct a URL that points to the uploaded image in the 'uploads' directory under the 'static' folder. The 301 status code in an HTTP response (Moved Permanently) is used to change the user's navigate to the image's URL. The function is defined to return this redirect response.

The final lines of code use the typical block to start the Flask application when the script is run directly. The 'app.run()' command starts the Flask development server to listen for incoming requests and serve the application. This block ensures that the application starts when the script is executed.

CHAPTER 8

TESTING

8.1 Introduction

Running a framework with the intention of finding errors requires testing. By identifying plan deviations and flaws in the framework, testing improves the framework's integrity. Testing focuses on identifying prom zones as mistakes. This helps to ensure that the framework is free of errors. By confirming the client's need, testing also raises the item's value.

The main goal is to identify errors and zones for error get-prom in a framework. The testing process must be rigorous and well-planned overall. Untested methodology is just as bad as one that's been employed a little. A framework that hasn't been tried and proven enough comes at a great expense. The final and most important step is execution. To make sure the structure is functioning well, it comprises client preparation.

8.2 Test Reports

When improvements are made in response to needs, users test the newly built system. Utilising multiple types of data, the constructed system is examined. during this process. The system is examined utilising data from the test after extensive testing is of the data carried out. To examine outputs with various sets of inputs, a test case is utilised. The below Table.No 8.1 shows the test reports.

Test Case	Test Purpose	Test condition	Expected outcome	Actual result	Pass or Fail
Login Credentials	Security purpose	Authentication	User must be logged in	Logged In successfully	Pass
Load Images data	Upload Liver Cancer CT scan data	Check for information	The upload was successful	Image is loaded Successfully.	Pass
Pre Processing	Apply methods like, noise removal, gray conversion	Pre-processing for image	Images are converted successfully	As Expected.	Pass
Apply feature classification using CNN	Feature extraction is done for pre-processed image	Check for feature extraction	Feature extraction data is complete	As Expected.	Pass
Detecting object using NN	Train model	Check for model loss and accuracy	Model trained successfully	Model trained successfully	Pass
Liver Cancer CT SCAN image as input	Detects the given object	Check for Liver classification with accuracy	Predicted result	Result is shown.	Pass

Table.No 8.1: Test Reports

CHAPTER 9

SCREENSHOTS

Home Page

File Name: [Index.html](#)

- The Home page is the essential tool for our website and often serves as the first impression to potential users, where visitors can find hyperlinks to other pages through navigation bars on the site.
- A tangible representation of logo is set on the second top bar of the header part parallel to this there will be a navigation bar which contains different pages, each page can be accessed by switching from home page.
- The below Fig.No 9.1 shows the screenshot of Home page

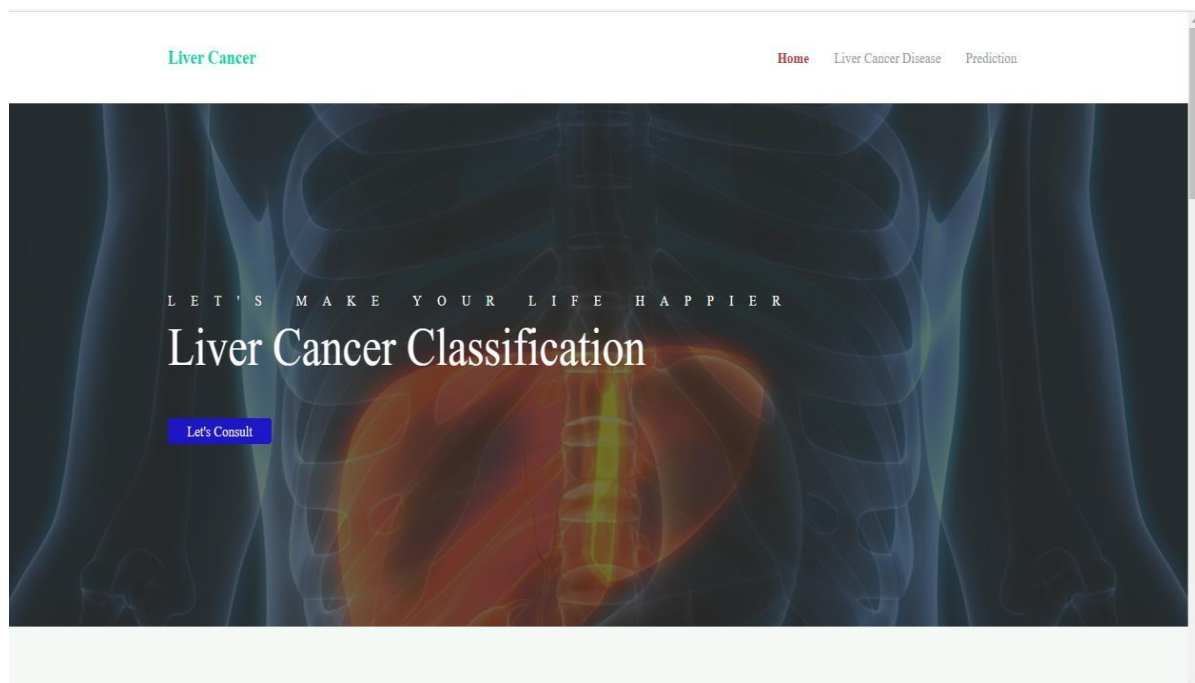


Fig.No 9.1: Screenshot of Home page

About Page

File Name: About.html

- The About Page is the complete details of the model and here users can learn additional model information
- The below Fig.No 9.2 shows the screenshot of Overview page

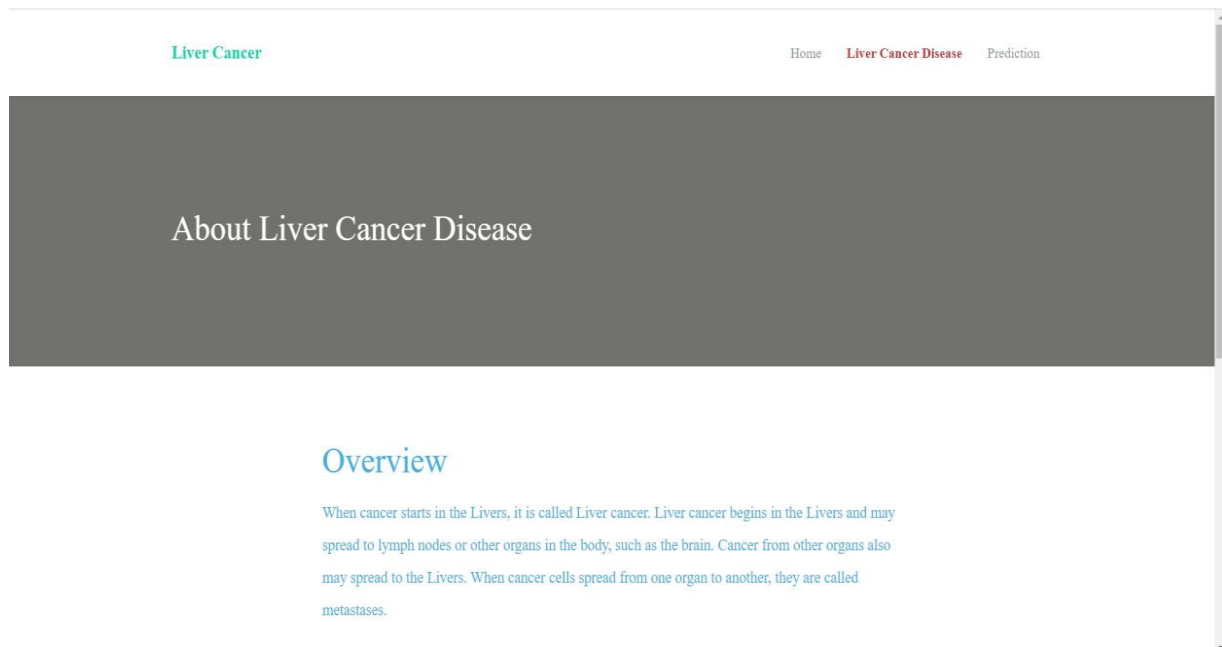


Fig.No 9.2: Screenshot of Overview Page

Prediction Page

File Name: Home.html

- This page is main page where we can find "Choose File" button, when click on that button, it allowing users to upload liver images from their local systems.
- The below Fig.No 9.3 shows the Screenshot of Prediction page



Fig.No 9.3: Screenshot of Prediction page

Result Page

File Name: Result.html

- Upon clicking the "Submit" button, the system seamlessly analyzes the uploaded image and promptly delivers predictions regarding cancerous or non-cancerous status, accompanied by a corresponding confidence level.
- The below Fig.No 9.4 shows the Screenshot of result

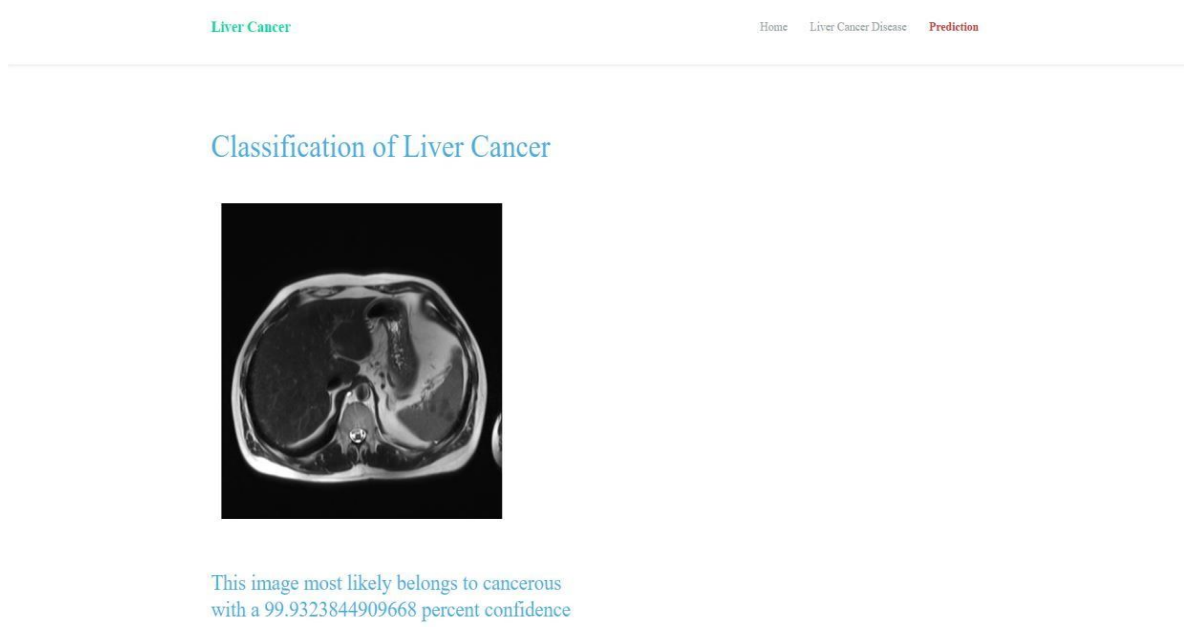


Fig.No 9.4: Screenshot of Result

CONCLUSION

The experiment successfully extracted features from liver tumors and classified them using CNN and deep learning approaches. The achievement of strong results was made possible by the use of CT scan image datasets for training and testing purposes. The effectiveness of the suggested technique was also examined through a thorough analysis of performance metrics. The combination of 2D feature maps and several slices revealed the potential for improving the precision of medical picture identification. Following the segmentation of tumors using voxel classification, the application of the Convolutional Neural Network showed promising results in the detection of tumor and healthy voxels within liver maskings. This experiment demonstrates the promise of deep learning while also laying the groundwork for future initiatives to improve the detection of liver illness.

FUTURE ENHANCEMENT

Incorporating Advanced Deep Learning Techniques:

- Explore state-of-the-art deep learning techniques, such as attention mechanisms, capsule networks, or generative adversarial networks (GANs), to improve the effectiveness of the liver cancer classification model.
- Investigate via means of pre-trained models, transfer learning, or fine-tuning on larger external datasets to improve generalization capabilities.

Handling Multi-modal Data:

- Consider integrating multiple imaging modalities, such as MRI, CT scans, or ultrasound, to leverage complementary information for more accurate liver cancer classification.
- Develop fusion techniques to mix features taken from many modalities effectively.

Handling Imbalanced Datasets:

- Address the challenges of imbalanced datasets by implementing techniques such as oversampling the minority class, under sampling the majority class to balance the training dataset.
- Explore advanced techniques like synthetic minority oversampling technique (SMOTE) or cost-sensitive learning to handle imbalanced data more effectively.

Explainability and Interpretability:

- Develop techniques to provide explanations or visualizations for the model's predictions, allowing medical professionals to understand the reasoning behind the classification results.
- Implement methods such as gradient-based class activation maps (CAM) or saliency maps to highlight regions of interest during liver images that contribute to the classification decision.

Integration with Clinical Systems:

- Enhance the software's integration capabilities to Integrate seamlessly with current clinical systems or electronic health records (EHR).
- Develop standardized interfaces or APIs to facilitate data exchange and interoperability with other medical imaging systems.

REFERENCES

- [1] "Performance analysis of liver tumor classification using machine learning algorithms", Munipraveena Rela, Suryakari Nagaraja Rao and Patil Ramana Reddy
- [2] "Liver Tumor Segmentation and Classification: A Systematic Review", Munipraveena Rela, Nagaraja Rao Suryakari and P Ramana Reddy
- [3] "Support Vector Machine based Liver Cancer Early Detection using Magnetic Resonance Images", Lei Meng, Changyun Wen, and Guoqi Li
- [4] "Liver Tumor Segmentation using Deep Learning Approach", Pallavi Bhandary, Vaibhavi More, and Renuka Nagpure
- [5] "Hepatocellular Carcinoma (HCC) Liver Cancer prediction using Machine Learning Algorithms", Sanapala Rajesh, Nurul Choudhury, and Soumen Moulik



The National Institute of Engineering


South Campus

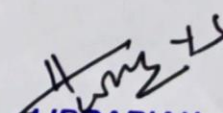
(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

• Recognised by AICTE, New Delhi • Grant-in-Aid by Government of Karnataka

CERTIFICATE ON PLAGIARISM CHECK

1.	Name of the Student	Dhruva Kumar K M
2.	USN Number	4NI21MC013
3.	Title of the Thesis / Project Report	Liver Cancer Segmentation and Classification
4.	Name of the Guide	Dr. Sanjay Kumar C K
5.	Department	Master of Computer Applications
6.	Course	PG - MCA
8.	Similarity Content (%) Identified	5%
9.	Date of Verification	28-08-2023

I have verified the content of Thesis / Project Report using  DrillBit Software. and the Similarity percentage is as mentioned above.


LIBRARIAN
The National Institute of Engineering
MYSURU-570008 