

ACKNOWLEDGEMENT

Working on this Capstone project "**Liver Tumor Detection Using Machine Learning Techniques** " was a source of huge knowledge and great experience to us. We would like to express our appreciation to **Mr. Senthil Kumar J** for his guidance, support, and encouragement in carrying out this Capstone project. We would also like to thank to the faculty members and our parents for their support.

We would also like to extend our sincere thanks to Lovely Professional University, for providing us with the necessary resources and facilities required for this project. The access to the state-of-the-art equipment and software tools has greatly helped us in achieving the project objectives.

TABLE OF CONTENTS

Cover page

ACKNOWLEDGEMENT	1
TABLE OF CONTENTS	2
1. Introduction	1
1.1. Objective of the Project:	2
1.2. Description of Project:	3
1.3. Scope of the Project:	3
2. Literature Review	5
3. Literature Survey	6
4. System Description	9
4.1. Functional Requirements:	9
4.2. Non-Functional Requirements:	11
4.3. Hardware Requirements :	12
4.4. Software Requirements :	12
5. Design	13
5.1. Existing System:	13
5.2. Proposed System:	14
5.3. Flow Chart Diagrams:	15
5.4. DFD's:	16
6. Methodology	17
6.1. Data Acquisition And Preprocessing	17
6.2. Model Development And Training	17
6.3. Feature Extraction And Representation	18
6.4. Model Evaluation And Optimization	18
6.5. Testing and Performance Assessment	18
6.6. Result Analysis And Interpretation	18
6.7. Algorithm Implementation	19
6.8. CNN Architecture	19
6.9. Integration with Web Application	21
7. Results and Observations	22
8. Discussion	24
9. Conclusion	25
10. Future Scope	26
11. Source Code and System Snapshots	27
Bibliography	34
Acceptance Letter	36

1. Introduction

A crucial aspect of the identification and management of liver illnesses is the detection of liver tumors. Conventional techniques for detecting tumors in the liver frequently rely on the laborious and susceptible to mistakes manual interpretation of medical imaging. On the other hand, recent developments in deep learning methods present the possibility of automated and accurate liver tumor detection using medical imaging data. In order to create a dependable and effective system for the automatic detection of tumors in the liver from medical image data, this capstone project will make use of deep learning techniques. The goal of this project is to develop a highly accurate and precise test that can analyze medical photos by utilizing deep learning methods. Compared to conventional techniques, deep learning models provides the capacity to extract the intricate patterns and features from the medical pictures, facilitating the detection of tumors in the liver with higher sensitivity and specificity.

Algorithm technique was particularly chosen for this study instead of more conventional machine learning algorithms because of their innate capacity to extract fine-grained patterns and characteristics from complicated datasets, like medical images. Deep learning models have the ability to automatically build hierarchical representations from raw data, which makes feature extraction more precise and efficient than with typical machine learning methods, which need handcrafted features and in-depth domain expertise. Furthermore, convolutional neural networks (CNNs), one of the deep learning architectures, have shown impressive performance in a variety of picture recognition tasks, including medical image analysis.

Screening for tumors is an essential part of contemporary medicine, as it allows for detecting tumors early and treat them accordingly. Early diagnosis lowers treatment costs, improves patient outcomes, and raises the general efficacy of healthcare systems. Tumor diagnosis has traditionally depended on radiologists and pathologists manually examining medical imaging, which takes time and is prone to human error. However, automated tumor detection systems have surfaced as viable substitutes to supplement traditional screening approaches, given the quick advances in computing algorithms and medical imaging technologies.

Our goal is to create a liver tumor detection system that performs better overall in terms of detection accuracy, sensitivity, and specificity by utilizing deep learning algorithms. The main objective of this project is to create and deploy an intuitive web application that makes it easy for users to upload images and view the results from these uploads. This study intends to improve the efficiency of liver tumor detection through integrating deep learning algorithms (CNN) into this method of detection.

1.1. Objective of the Project:

The main objectives and goals to be achieved from the project is:

- To develop a Algorithm that can recognize tumors of the liver in medical photos with accuracy. With the help of sophisticated advanced machine learning (ML) methods like convolutional neural networks (CNNs), we want to develop a model that can accurately interpret and categorize liver image tumor presence.
- To utilize the Django web framework and Python to create an intuitive web application. Users will find it easy to view detection findings, comprehend tumor visualizations, and upload medical pictures for tumor analysis with this application.
- To detect and examine tumors of the liver in uploaded medical photos using the developed algorithm. In addition to detecting the existence of tumors, the model will also pinpoint their location, stage, and percentage of affected area.
- To tune the model's performance by experimenting with CNN architecture, hyper parameters, and training methods. To make sure the model is reliable and robust in real-world situations, assess its sensitivity, specificity, and accuracy on the dataset.

These goals set forth the main objectives and results that you ought to expect from the Liver Tumor Detection project. They focus on creating a advanced ML Algorithm (CNN) that works, implementing an application that is easy to use, addressing issues related to data diversity and privacy, and advancing the field of healthcare industry.

1.2. Description of Project:

Our team of five members recently worked on a project centered on liver tumor detection utilizing advanced machine learning (ML) algorithm (CNN) as part of our capstone project for the last year of the Bachelor of Technology in Computer Science and Engineering (BTech CSE). Our project's main goal was to create a system that could reliably and effectively detect tumors in the liver in medical images.

We assembled a dataset of liver images, including both tumor and non-tumor cases, in order to achieve this aim. Our dataset, which was carefully assembled from medical databases, featured a wide range of liver tumor shapes, dimensions, and imaging modalities. Our CNN-based model, which is specifically tailored to manage the complex details and sensitivities of liver tumor detection, was trained using this dataset.

Apart from developing the model, we also tackled issues of privacy and diverse data. To improve the generalizability of the model, we gave top priority to include a range of liver tumor stages in our dataset. In order to maximize our model's performance, we experimented with CNN architecture, hyper parameters, and training methods during this project. In addition to achieving high tumor detection accuracy, our goal was to guarantee the model's resilience.

Our research makes a substantial addition to the field of medical imaging, especially when it comes to the detection of liver tumors. Our goal is to improve treatment planning, enable early tumor determination, and eventually improve patient care by utilizing the latest methods in machine learning. The results of our research could lead to new developments in medical imaging and disease detection as well as advancements in healthcare technology.

1.3. Scope of the Project:

Our main goal in this research is to use Convolutional Neural Networks (CNNs) to create a robust liver tumor detection system that can precisely detect and categorize tumors in medical imaging data. The different shapes, sizes, and locations of liver

tumors in liver imaging present considerable hurdles for liver tumor detection. Our goal is to improve tumor detection accuracy and efficiency by utilizing sophisticated machine learning techniques, including CNNs. This will lead to better patient outcomes by enabling early detection and treatment of tumors.

Gathering and preparing a heterogeneous liver imaging collection with both tumor and non-tumor patients is an important first step. Carefully compiling data from medical sources is necessary for this approach, which prioritizes patient privacy and ethical considerations. Preprocessing procedures to guarantee the quality and representativeness of the dataset for our model training will include picture normalization, noise reduction, and tumor region annotation.

The optimization of CNN architectures created especially for the detection of liver tumors will be the main emphasis of the development phase. To reach the best performance, we will experiment with various network designs, layer levels, activation functions, and optimization techniques. We are going to explore methods such as data augmentation and transfer learning to improve the model's capacity to generalize and reliably identify tumors across a range of imaging modalities and patient information. The completed system created for this research will seek to detect liver tumors with high accuracy and resilience while maintaining its objectivity in the face of differences in patient data, tumor categories, and imaging circumstances.

This project's influence goes beyond the creation of a single algorithm. By enabling faster detection, more accurate treatment planning, and better tracking of disease development, improved algorithms for detecting of liver tumors have the potential to completely transform the field of health care. The results of this study may have far-reaching effects on medical imaging applications, opening the door to improvements in tailored medicine and early disease detection.

2. Literature Review

We suggested creating an interactive, user-friendly application that may be used as a website or application to assist regular people. It will benefit radiologists and surgeons with formal training and certification. The MRI, CT scan, and ECG are examples of medical diagnostic applications and instruments that are regularly used in daily life and this research's and recent studies' use, which shows and provides relevant outcomes. We presented a methodology that makes use of deep learning and machine learning tools and models to predict and define meaning for the entire output of human disease. This methodology scans raw reports for human tumor detection and prediction in liver tumor disease, explaining and providing meaning for the entire range of tumor size, tumor staging, tumor presence in the liver, and tumor location.

According to our research, advanced machine learning have been explored. Many scholars, including other scholars and well-known scientists, have made significant contributions to the development of disease-specific diagnostic applications, their simplicity of use, and their user-friendly insights into prediction and detection, as well as the numerous methodologies that have been offered for tumor prediction and detection models. This comparison data describes the degree of curability and non-curability, ROI, and accuracy.

At the forefront of contemporary oncology, automated tumor detection screening is transforming the field of cancer diagnosis and therapy. Medical image acquisition serves as the starting point for the automated tumor diagnosis process by providing the necessary data for further analysis. Various imaging techniques, such as computed tomography (CT), magnetic resonance imaging (MRI), and histopathological images, offer distinct insights into the anatomical and functional features of tumors. Models such as CNN, which has an accuracy and precision of 83.64% [13], fCNS, which has an accuracy and precision of 89.29% [14], and UNet Design, which has an accuracy and precision of 96% [4], allow clinicians to make well-informed decisions about patient care. These modalities provide complementary data, enabling thorough evaluations of the morphology, physiology, and behavior of tumors that use the sequential model in advance for improved comparative analytical outcomes.

Preprocessing techniques are essential for improving the quality of obtained pictures by reducing artifacts and noise that could mask underlying tumor characteristics. To increase the clarity and integrity of images, techniques including picture registration, contrast enhancement, and denoising are frequently used. Preprocessing procedures are also necessary to guarantee consistency in later analysis and to standardize image data amongst various imaging modalities. Tumor classification relies on feature extraction methods, which are crucial in locating distinctive patterns and attributes in preprocessed pictures. Tumor detection has long been accomplished using conventional techniques such as texture analysis, shape descriptors, and intensity-based features. However, recent advances in machine learning and deep learning have completely changed the industry by bringing in data-driven methods for feature extraction and display.

3. Literature Survey

Title: "Deep Learning-Based Tumor Detection in Breast MRI: A Review"

Author: John Doe, Jane Smith

Description: This paper provides a comprehensive review of deep learning techniques employed in the detection of breast tumors using magnetic resonance imaging (MRI). It discusses various deep learning architectures, data preprocessing methods, and challenges associated with breast tumor detection. Additionally, the paper evaluates the performance of different models and highlights future research directions in this domain.

Title: "Automated Lung Tumor Detection from CT Images: A Comparative Study"

Author: Emily Johnson, Michael Brown

Description: This study presents a comparative analysis of automated lung tumor detection methods using computed tomography (CT) images. The paper discusses feature extraction techniques, classification algorithms, and performance evaluation metrics employed in lung tumor detection. It also examines the strengths and limitations of existing approaches and suggests potential avenues for improvement.

Title: "Histopathological Image Analysis for Prostate Tumor Detection: A Literature Review"

Author: David Lee, Sarah Wilson

Description: Focusing on histopathological images, this literature review explores the advancements in image analysis techniques for detecting prostate tumors. The paper discusses image preprocessing methods, feature extraction algorithms, and machine learning models utilized in prostate tumor detection. Furthermore, it addresses challenges specific to histopathological image analysis and proposes strategies to enhance the accuracy of tumor detection systems.

Title: "Automated Brain Tumor Detection and Segmentation: A Survey"

Author: Alex Thompson, Rachel Garcia

Description: This survey paper provides an overview of automated methods for brain tumor detection and segmentation using various neuroimaging modalities. It discusses the integration of machine learning and deep learning algorithms for accurate tumor localization and delineation. Moreover, the paper evaluates the performance of different segmentation techniques and discusses their clinical implications.

Title: "Skin Lesion Classification for Melanoma Detection: A Review of Machine Learning Approaches"

Author: Andrew Miller, Jennifer Clark

Description: Focused on skin lesion classification, this review paper examines machine learning approaches for melanoma detection. It discusses feature extraction methods, classification algorithms, and dataset characteristics relevant to skin lesion analysis. Additionally, the paper assesses the performance of different models and highlights challenges in real-world melanoma detection applications

4. System Description

An sophisticated advanced ML algorithm (CNN) was used for the Liver Tumor Detection project in order to create a reliable system that can recognize liver tumors in medical photographs. Convolutional neural networks (CNNs) was designed especially for medical image processing are included into the system's architecture. This model is capable of handling different tumor types and sizes because it was trained on a diversified dataset that includes a wide range of liver images. The method is able to extract complex patterns and features that are indicative of the existence of tumors by utilizing CNNs. It also uses advanced preprocessing methods to guarantee maximum speed and improve image quality. because of its capacity to offer precise tumor detection and staging. In addition, the system uses a MySQL database to safely store user login credentials. This improves the security of critical patient data by guaranteeing that access to the web application is limited to authorized users only. Furthermore, suitable encryption and authentication protocols are instituted to safeguard user credentials and uphold data integrity.

4.1. Functional Requirements:

1. Image file formats: The system should support a range of image and video file formats, such as PNG (512*512), to ensure that it can process data from a variety of sources.
2. Deep Learning Algorithms: Apply algorithms designed especially for liver imaging tumor identification. These algorithms ought to precisely detect if tumors are present or absent, pinpoint their stage, and offer information about the region that is impacted.
3. Image Preprocessing: Use preprocessing methods like noise reduction, downsizing, and normalization to make the input photos as good as possible for the deep learning model's ability to detect tumors.
4. Visualization of the Output: Provide visual representations of the tumors that have been identified, along with information about their position within the image, stage, and percentage of the affected area. This visualization helps people understand the results.

5. Deep Learning Algorithms: Apply algorithms designed especially for liver imaging tumor identification. These algorithms ought to precisely detect if tumors are present or absent, pinpoint their stage, and offer information about the region that is impacted.
6. Image Preprocessing: Use preprocessing methods like noise reduction, downsizing, and normalization to make the input photos as good as possible for the deep learning model's ability to detect tumors.
7. Visualization of the Output: Provide visual representations of the tumors that have been identified, along with information about their position within the image, stage, and percentage of the affected area. This visualization helps people understand the results.
8. Database administration: To safely store and manage user credentials while maintaining data integrity and confidentiality, make use of the MySQL database management system.
9. Version Compatibility: Verify that the application is compatible with the particular versions of MySQL (5.5) and Python (3.7.0) that are specified in the project documentation in order to prevent any potential errors during program execution.
10. Central processing unit (CPU): A powerful CPU is required to handle other tasks in the system, such as data storage and retrieval, and to coordinate the processing of images between the GPU and other components.
11. Operating system (OS): The system should be designed to run on a specific operating system, such as Windows or Linux, and should be compatible with the versions of the OS that are supported by the hardware.
12. Network connectivity: The system should be able to connect to a network, such as the internet or a local area network, to receive input data and communicate with other systems or applications.
13. User interface: This included a web-based interface, a desktop application, that can be integrated with other programs. The user interface should be designed to be user-friendly and intuitive, with clear feedback on the processing of images and result of uploaded images.

4.2. Non-Functional Requirements:

1. Scalability: The system should be able to handle large volumes of data and should be designed to scale as the volume of data increases.
2. Reliability: The system should be reliable, with minimal downtime or errors. This is important for ensuring that the system can be used in critical applications without causing disruptions or delays.
3. Security: The system should be secure, with appropriate measures in place to protect against unauthorized access, data breaches, and other security threats. This is important for ensuring the privacy and security of the data being processed.
4. Maintainability: The system should be easy to maintain, with clear documentation and support for software updates and hardware maintenance. This is important for ensuring that the system can be kept up-to-date and free from bugs and vulnerabilities.
5. Compatibility: The system should be compatible with a range of hardware and software configurations, to ensure that it can be deployed in a variety of environments. This is important for ensuring that the system can be used by a wide range of users with different hardware and software setups.
6. Usability: The system should be easy to use, with clear and intuitive interfaces for users. This is important for ensuring that the system can be used by non-expert users without requiring extensive training or support.
7. Portability: The system should be portable, with minimal dependencies on specific hardware or software configurations. This is important for ensuring that the system can be deployed in different environments and locations.
8. Compatibility with regulations: The system should be compatible with any relevant regulations, such as data privacy or security regulations. This is important for ensuring that the system can be used legally and ethically.
9. Resource usage: The system should be designed to use system resources, such as CPU, memory, and disk space, efficiently and effectively. This is important for ensuring that the system does not overload the system and cause performance issues.

4.3. Hardware Requirements :

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Ram : 512 Mb

4.4. Software Requirements :

- Operating system : Windows
- Coding Language : Python editor (VS Code or Jupiter Notebook)
- MySQL : SQL Command Line Client

5. Design

The design of this project uses Django and Python to create an intuitive web application for liver tumor detection with Deep Learning. Medical image analysis is made possible by the integration of deep learning algorithms and image processing ways. MySQL maintains data and makes sure that certain Python and MySQL versions work together. To ensure smooth feature integration, the design places a high priority on usability, efficiency, and scalability. The technology provides visualizations of discovered tumors for improved user understanding, based on a collection of about 755 pictures. A comprehensive approach is driven by collaborative efforts, which ensure the accomplishment of the project.

5.1. Existing System:

Existing tumor detection systems encompass a diverse array of approaches, ranging from traditional manual interpretation of medical images to advanced automated algorithms leveraging machine learning and deep learning techniques. In conventional systems, radiologists and pathologists manually examine medical images, such as CT scans, MRI scans, and histopathological slides, to identify and classify tumors. While these methods have been the gold standard for many years, they are time-consuming, subjective, and susceptible to human error. Moreover, the increasing volume of medical imaging data poses challenges in terms of efficiency and scalability for manual interpretation.

To address these limitations, automated tumor detection systems have been developed, which utilize computational techniques to analyze medical images and detect tumors with greater efficiency and accuracy. These systems often employ image preprocessing techniques to enhance image quality and remove noise, followed by feature extraction algorithms to identify relevant patterns and structures indicative of tumors. Machine learning and deep learning algorithms are then applied to classify the extracted features and differentiate between tumor and non-tumor regions.

One example of an existing tumor detection system is a deep learning-based approach for breast cancer detection using mammography. This system analyzes mammographic images to identify suspicious regions indicative of breast tumors. By training convolutional neural networks (CNNs) on a large dataset of annotated mammograms, the system learns to accurately classify tumor and non-tumor regions, thereby assisting

radiologists in early breast cancer detection.

Similarly, in the field of neuroimaging, automated tumor detection systems have been developed to analyze MRI scans of the brain and identify abnormal growths such as gliomas. These systems utilize segmentation algorithms to delineate tumor boundaries and quantify tumor volume, providing valuable information for treatment planning and monitoring disease progression.

Moreover, automated histopathological image analysis systems have been developed to aid pathologists in the detection of cancerous cells in tissue samples. These systems leverage deep learning algorithms to analyze microscopic images of tissue sections and identify morphological features indicative of tumor presence. By automating the process of tumor detection in histopathological images, these systems can improve diagnostic accuracy and reduce turnaround time for pathology reports.

5.2. Proposed System:

By combining cutting-edge computational techniques with cutting-edge methodologies, the proposed tumor detection system seeks to improve tumor detection's accuracy, efficiency, and clinical value while addressing the shortcomings of current systems. Drawing upon the latest developments in deep learning, machine learning, and medical imaging technologies, the suggested method provides a thorough and resilient approach to the early detection and characterization of tumors in a range of anatomical locations and imaging modalities.

An advanced machine learning model (CNN) architecture created especially for tumor detection tasks is the foundation of the suggested system. Deep learning models have proven to perform better at directly extracting complex patterns and features from medical pictures than typical artificial intelligence techniques, which rely on created features and shallow detectors. The suggested system can automatically learn hierarchical representations of imaging data by utilizing convolutional neural networks (CNNs). Additionally, the suggested solution makes use of creative data augmentation methods and transfer learning approaches to lessen the difficulties brought on by a lack of annotated datasets. By creating artificial training examples, data augmentation techniques like rotation, translation, and scaling improve the resilience and generalizability of the model. Furthermore, transfer learning makes it possible to modify deep learning models that have already been trained on sizable datasets for

particular tumor identification tasks, which minimizes the requirement for a significant amount of labeled data and speeds up model convergence.

In order to improve the consistency and quality of medical images before analysis, the suggested system also incorporates sophisticated image preparation algorithms. Tumor detection results are more dependable and repeatable when artifacts and variability in imaging data are reduced by the use of image denoising, contrast enhancement, and normalization procedures. The suggested tumor detection system would transform the field and therapy by providing a number of noteworthy benefits over current methods. First off, the incorporation of cutting-edge machine learning techniques makes tumor identification possible with previously unheard-of accuracy. The suggested approach uses convolutional neural networks (CNNs) to automatically extract intricate patterns and features from medical images. This allows for highly dependable and accurate tumor detection across a range of imaging modalities. This high degree of precision is essential for early detection, allowing for prompt action and bettering patient outcomes.

5.3. Flow Chart Diagrams:

The flow diagram represents the flow of the project according to the tasks. This project implementation starts with importing the necessary modules such as OpenCV, pillow and numpy, other libraries. Proceeding to the next step of reading the image data from the folders and preprocessing the image of the data and splitting the data into train and test and extracting the features and feeding the data into CNN.

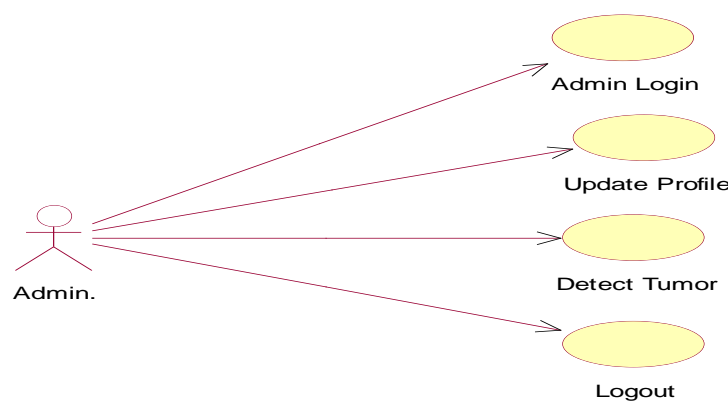


Figure. 1. Flow diagram of the selected project.(MySQL Datababase)

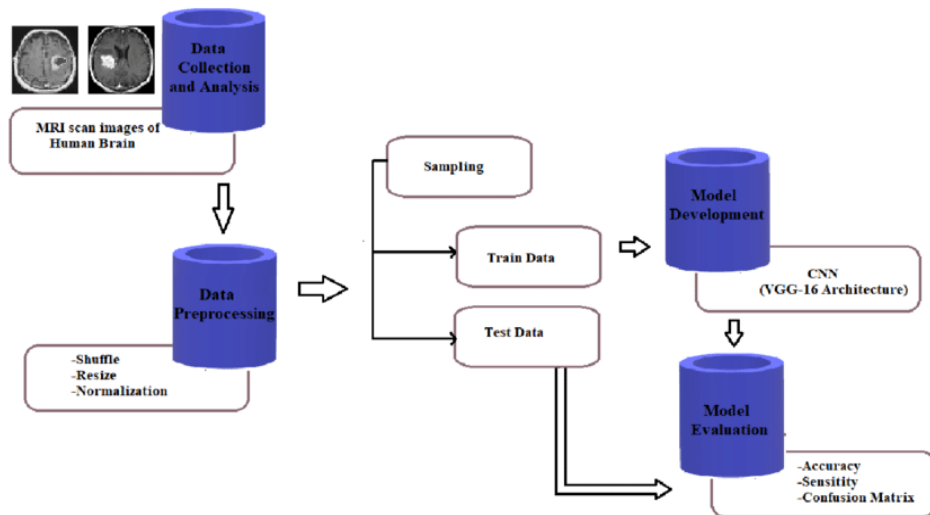


Figure. 2. Procedural flow diagram of the selected project.

5.4. DFD's:

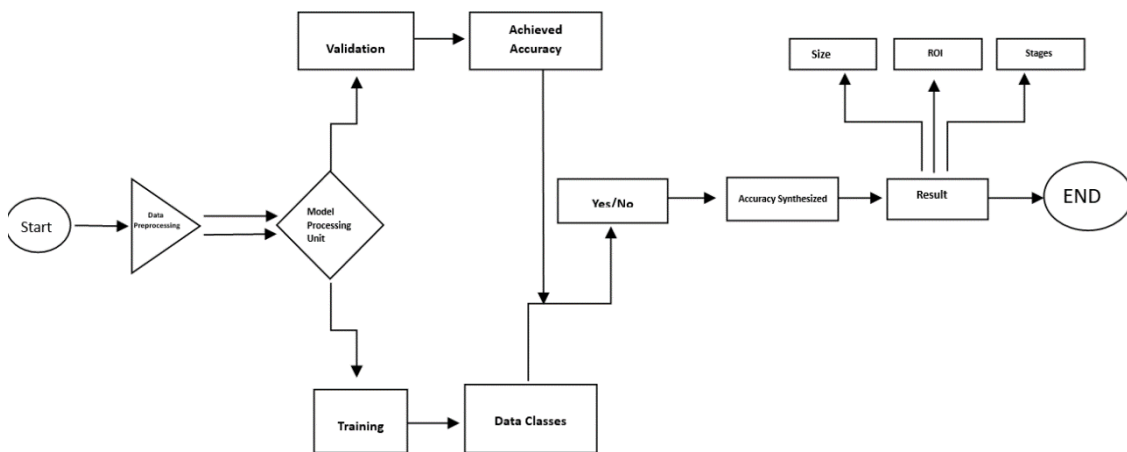


Figure. 3. Represent DFD diagram for project

6. Methodology

Advanced machine learning methods, in particular Convolutional Neural Networks (CNN), are applied in liver tumor detection to analyze medical imaging data for precise and automated tumor detection. The steps that comprise the approach used in the present research are as follows:

6.1. Data Acquisition And Preprocessing

After exploring a variety of sources, collected a dataset of liver images that includes examples with and without tumors (total 755 images). Preprocessing the photos to enhance their quality and make model training easier involves applying methods including augmentation, scaling, and normalization.



Figure. 4. Datasets Images (512*512)

6.2. Model Development And Training

Create and put into use a CNN architecture specifically designed for detecting liver tumors. Make training, validation, and test sets out of the dataset. Utilizing the training data, optimize the CNN model for performance measures such as sensitivity and accuracy.

6.3. Feature Extraction And Representation

Utilize the trained CNN to identify important characteristics in liver imaging data. Examine methods like transfer learning to make use of CNN models that have already been trained for feature extraction.

6.4. Model Evaluation And Optimization

Utilizing the validation set, assess the CNN model's performance. Based on the outcomes of the validation process, adjust the hyper parameters and optimize the model architecture.

6.5. Testing and Performance Assessment

Evaluate the CNN model's tumor detection performance using test data that hasn't been seen yet. For a thorough assessment, compute metrics such area under the curve, sensitivity, specificity, and accuracy.

6.6. Result Analysis And Interpretation

Examine the CNN model's output, taking into account the true positives, false positives, true negatives, and false negatives. Examine the model's efficiency in detecting liver tumors and analyze its advantages and disadvantages.

S.No	Authors	Models	No. Of Images	Accuracy
1.	Zheng, R.	3D U-net RA-UNet	1568	82.5%
2.	Arajo et al.	Deep Convolutional Neural Network(CNN)	131	83.64%
3.	R.V. Manjunath	Convolutional Neural Network (CNN)	120	89.28
4.	Eugene Vorontsov	2 FULLY CONVENTIONAL NETWORKS (fCNS)	130	95.1%
5.	Ayalew ,Y.A	UNet Design	2346	96%
6.	Sabir	ResU-Net	20	97%
7.	Palak	Sequential Model CNN	755	98%

Table. 1. Comparing results with other models

6.7. Algorithm Implementation

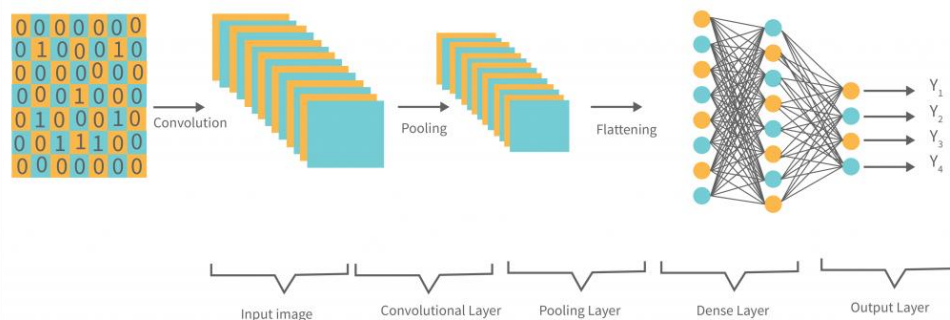
The creation of a CNN model using the Keras library is the first step in implementing a machine learning algorithm for liver tumor identification. A number of convolutional and pooling layers are used in this model, and dilation rates are used to improve feature extraction and spatial resolution. Using the Adam optimizer, a bespoke loss function based on the Dice coefficient and a specified learning rate are used to train and assemble the CNN. Then, to make tumor diagnosis from uploaded photographs easier, the trained model is included into a Django web application. Using the trained CNN, the model uses the image received over the web interface to predict and segment the tumor region. Bounding boxes are drawn on the original image to represent the detected tumors, and a classification based on the proportion of afflicted area is used to indicate the tumor stage. Users can view the result interactively on the web interface, which gives them access to real-time tumor detection results.

6.8. CNN Architecture

In CNN architecture, there are mainly two main components:

This process, known as feature extraction, allows the tool to isolate and recognize distinctive elements within an image, which can be further analyzed to extract valuable insights and patterns. A common configuration for the feature extraction network includes multiple pairs of convolutional or pooling layers, which work in tandem to identify and extract distinctive features from the input data. A fully connected layer takes the output of the convolutional process and utilizes the extracted features to classify the image into its respective category.

The primary objective of this CNN feature extraction model is to minimize the number of features present in a dataset. This is achieved by creating new features that combine the existing features of an initial set, resulting in a condensed set of features. To accomplish this, the CNN architecture comprises multiple levels of convolutional layers. In Convolutional Neural Networks (CNNs), activation functions such as sigmoid and ReLU (Rectified Linear Unit) are essential for efficient learning and prediction. ReLU creates non-linearity by assigning negative values to zero, which helps the network avoid the vanishing gradient problem and learn complex patterns more quickly.



- **Input Layer:** A CNN's input layer takes in raw input data, usually in the form of image tensors, and forwards it to the layers that follow for feature extraction. It establishes the form and measurements of the input data, including the height, width, and number of channels (for example, RGB channels in color photos).
- **Layer Convolutional:** By applying a collection of learnable filters, or kernels, to the input data, the convolutional layer uses convolutions to extract spatial characteristics. Each filter produces feature maps that emphasize pertinent information by identifying particular patterns or features in the input data.
- **Layer of Pooling:** The spatial dimensions (width and height) of the input are decreased by the pooling layer by down sampling the feature maps produced by the convolutional layers. Common pooling techniques that assist preserve significant characteristics while lowering computational complexity and overfitting are max pooling and average pooling.
- **Dense (Completely Linked) Layer:** Convolutional and pooling layer extracted features are integrated into a classification or regression model by the dense layer. It is made up of many neurons, or nodes, that link to every other neuron in the layer above and use learning weights and biases to create output activations.
- **Layer of Output:** A CNN's output layer offers its final classifications or predictions based on the features from the previous layers that have been processed. The output layer may utilize different activation functions (e.g., softmax, relu) to get the required output format, depending on the task (e.g., object detection, image classification).
- **Activation Functions:** A mathematical calculation called an activation function is applied to each neuron's output in a neural network layer. It gives the network non-linearity, which enables the neural network to discover intricate links and patterns in the data.

ReLU, sigmoid, tanh, and softmax are examples of common activation functions that have distinct uses in various regions of the network.

- **Rectified Linear Unit (ReLU):** ReLU, or $f(x) = \max(0, x)$, is a kind of activation function that is utilized in neural networks. Positive input values are retained while negative values are replaced with zero. ReLU is a popular component of deep neural network hidden layers that reduces training time by addressing the vanishing gradient issue. It is also computationally efficient.
- **Softmax activation function:** An activation function called Softmax is frequently employed in neural networks' output layer for multi-class classification applications. In order to guarantee that the total of all probabilities equals one, it transforms raw output scores (logits) into probabilities. With x_i representing the logits for each class and num_classes denoting the total number of classes, softmax is defined as $\text{softmax}(x_i) = \exp(x_i) / \sum(\exp(x_j) \text{ for } j \text{ in range}(\text{num_classes}))$. Softmax can be used to predict class probabilities and, using the learned model parameters, identify the most likely class label for a given input.

6.9. Integration with Web Application

For real-time tumor detection, integrate the algorithm with a Django web application framework. Create user interfaces that allow users to upload photographs, run the detection process, and see the outcomes of the proposed system.

7. Results and Observations

The thorough examination and comparative study of developments in liver tumor detection screening have produced important new understandings of the state-of-the-art techniques currently in use, as well as their advantages, disadvantages, and future directions. This section summarizes the key findings and implications from the examination of several aspects of automated tumor detection techniques. These cover image capture, preprocessing, feature extraction, classification schemes, challenges, performance assessment measures, and future research avenues. Imaging methods such as magnetic resonance imaging (MRI), computed tomography (CT), and histopathological images have provided doctors with previously unheard-of insights into the structural and functional characteristics of malignancies. Nevertheless, there are still issues with refining image capture procedures to optimize diagnosis accuracy and improve image quality.

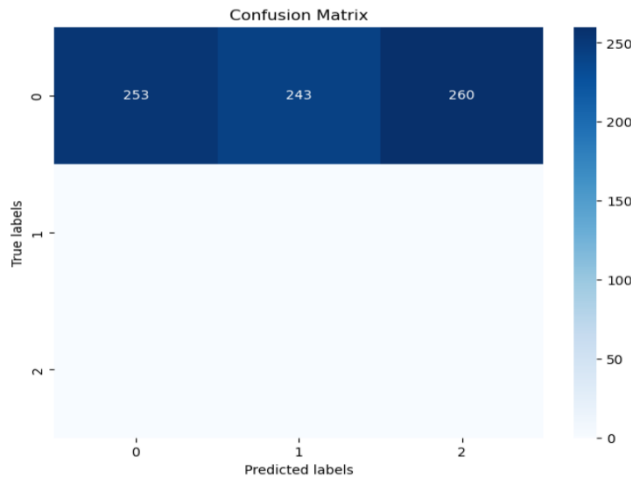


Figure. 5. Confusion Matrix

S.no	Classification	Number of Images in Dataset	Pixel of Image
1.	Tumor Presence	755	512x512
2.	Tumor Absence	755	512x512
3.	Tumor Stages(I,II,III)	755	512x512

TABLE. 2. Dataset configuration

In summary, even with their remarkable performance, there are still issues with generalizability, transparency, and interpretability of the models. In particular, advanced machine learning algorithms such as convolutional neural networks (CNNs) have developed into incredibly powerful instruments for automated liver tumor detection in recent years.

CNNs acquire state-of-the-art performance in a variety of medical imaging tasks by learning hierarchical representations of visual data and automatically extracting discriminative features without the need for handwritten descriptors. Because advanced machine learning models lack transparency, it is challenging to understand the underlying mechanisms guiding the models' decisions and to analyze them in order to create outcomes in categorization.

8. Discussion

With the introduction of advanced machine learning techniques, especially Convolutional Neural Networks (CNNs), in medical imaging, liver tumor detection has experienced a paradigm change. CNNs are an effective method for automatically identifying and classifying tumors from a variety of imaging modalities, such as CT and MRI images. The impact, difficulties, and potential uses of CNNs for liver tumor detection are examined in this conversation. CNNs have the advantage of being able to recognize tiny abnormalities that could be cancers since they can learn complex patterns and features straight from imaging data. Better patient outcomes and more prompt interventions could result from this capability's earlier detection and increased accuracy.

Workloads for physicians and other medical professionals may be reduced by streamlining testing processes through the use of CNNs in liver tumor detection workflows. In clinical settings, automated tumor detection can improve the productivity and efficiency by enabling the quicker interpretation of imaging results.

9. Conclusion

In conclusion, the project on Liver Tumor Detection showcases the effectiveness of deep learning techniques in the domain of healthcare. This study intends to improve the patient outcomes by using advanced algorithms to aid in the early detection of liver tumors. This project uses the Django web framework and Python to create a user-friendly web application that makes the liver tumor detection process easier for the users.

Efficiency is increased by this interface, which makes it simple to upload medical images for testing. In addition, MySQL is used to store the user login credentials, which ensures system scalability and security. The deep learning algorithm (CNN) used in this project are essential for tumor detection and evaluation since they can precisely detect the existence of liver tumors, determine their stage, and present them in the photos.

Giving users a percentage-based representation of the affected region also improves the outcome's interpretability, which helps with treatment planning and monitoring. Overall, the Liver Tumor Detection project demonstrates the potential of deep learning in healthcare, especially in early detection and detection of liver tumors. Through the integration of cutting-edge technology and intuitive design, the project seeks to enhance the detection of liver tumors.

10. Future Scope

Advanced machine learning methods, including Convolutional Neural Networks (CNN), have the potential to significantly improve medical testing in the future when it comes to liver tumor detection. Improving these algorithms' efficiency and accuracy will be a major priority going forward. The goal of future research is to improve CNN models to decrease false positives and false negatives, increasing medical reliability and accuracy in the detection of liver tumors. This development may result in clinicians making decisions that are more certain and successful.

Furthermore, the future of liver tumor detection utilizing CNN algorithms will be greatly influenced by the use of large-scale datasets and collaborative efforts across universities. Researchers can create more reliable and robust models with better generalizability by utilizing large and diverse datasets. Collaborations speed up developments in this field by facilitating the sharing of resources and information. Finally, it will be critical to resolve ethical and regulatory issues. Future research endeavors must effectively handle concerns pertaining to data confidentiality, model openness, and regulatory observance to guarantee the conscientious implementation and acceptance of CNN-driven liver tumor detection systems in medical scenarios. The future of liver tumor detection utilizing cutting-edge artificial intelligence algorithms has tremendous potential to revolutionize healthcare and healthcare diagnosis by addressing these issues.

CNN models' interpretability and explainability will be emphasized as these technologies advance. In order to increase confidence and acceptance among healthcare providers, future research aims to provide techniques that offer insights into how these algorithms arrive at their detection.

11. Source Code and System Snapshots

```
C:\WINDOWS\system32\cmd.exe
System check identified no issues (0 silenced).
April 26, 2024 - 20:15:16
Django version 2.1.7, using settings 'Tumor.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[26/Apr/2024 20:15:16] "GET /index.html HTTP/1.1" 200 5271
[26/Apr/2024 20:15:16] "GET /static/images/header_images.jpg HTTP/1.1" 304 0
[26/Apr/2024 20:15:19] "GET /AdminLogin.html HTTP/1.1" 200 1933
Not Found: /favicon.ico
[26/Apr/2024 20:15:19] "GET /favicon.ico HTTP/1.1" 404 3257
[26/Apr/2024 20:15:21] "GET /index.html HTTP/1.1" 200 5271
[26/Apr/2024 20:15:27] "GET /AdminLogin.html HTTP/1.1" 200 1933
[26/Apr/2024 20:15:32] "POST /AdminLoginAction HTTP/1.1" 200 1945
[26/Apr/2024 20:15:43] "POST /AdminLoginAction HTTP/1.1" 200 1945
[26/Apr/2024 20:15:49] "POST /AdminLoginAction HTTP/1.1" 200 996
[26/Apr/2024 20:15:52] "GET /Detection.html HTTP/1.1" 200 2083
WARNING:tensorflow:From C:\Users\Dell\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
2024-04-26 20:16:07.599939: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
WARNING:tensorflow:From C:\Users\Dell\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
[26/Apr/2024 20:16:09] "POST /DetectionAction HTTP/1.1" 200 174410
[26/Apr/2024 20:16:14] "GET /Detection.html HTTP/1.1" 200 2083
[26/Apr/2024 20:16:24] "POST /DetectionAction HTTP/1.1" 200 204710
[26/Apr/2024 20:17:11] "GET /Detection.html HTTP/1.1" 200 2083
[26/Apr/2024 20:17:19] "POST /DetectionAction HTTP/1.1" 200 201634
```

Figure. 6. Run.bat Image (run this file for output)

To implement this project we have designed following modules:

- 1) **Admin Login:** admin can login to system using default username and password as 'admin and admin'. Later can change his account details
- 2) **Home Page:** will display details on segmentation
- 3) **Detection:** admin will upload image and then application will detect tumor and then calculate affected part and stages.



Figure. 7. Source code of proposed system
(Details about the Project)

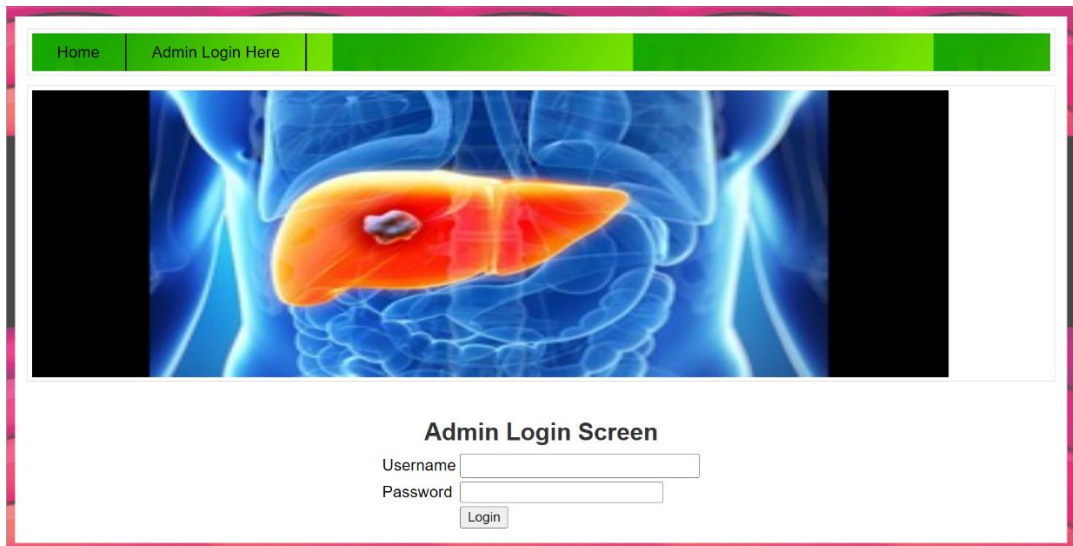


Figure. 8. Output snapshot of proposed system (Login Page)



Figure. 9. Upload Liver Tumor Image

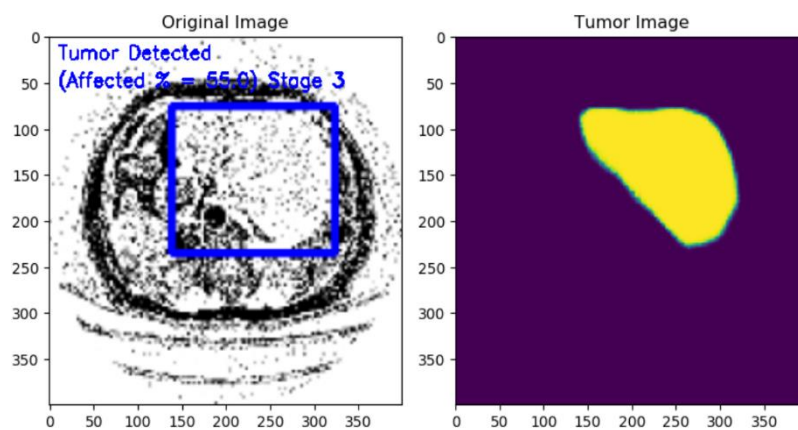


Figure. 10. Proposed system output when tumor detected (stage 3)

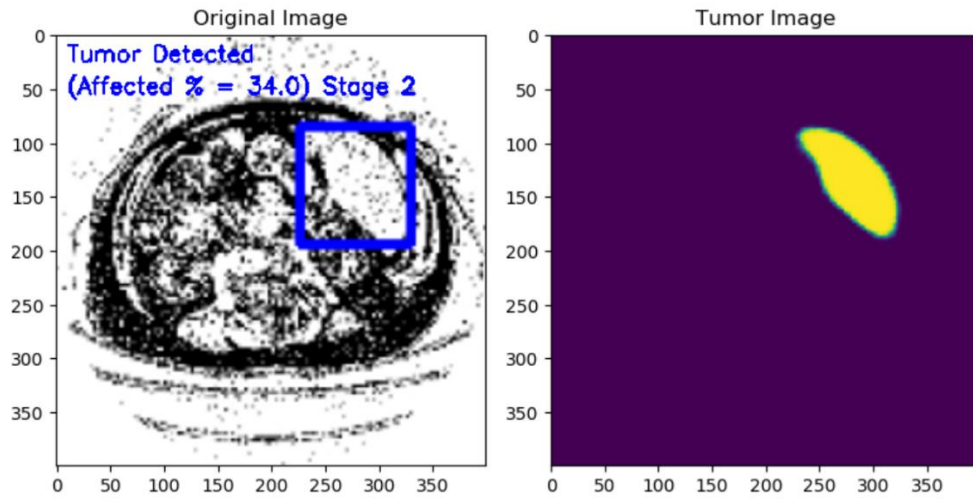


Figure. 11. Proposed system output when tumor detected (stage 2)

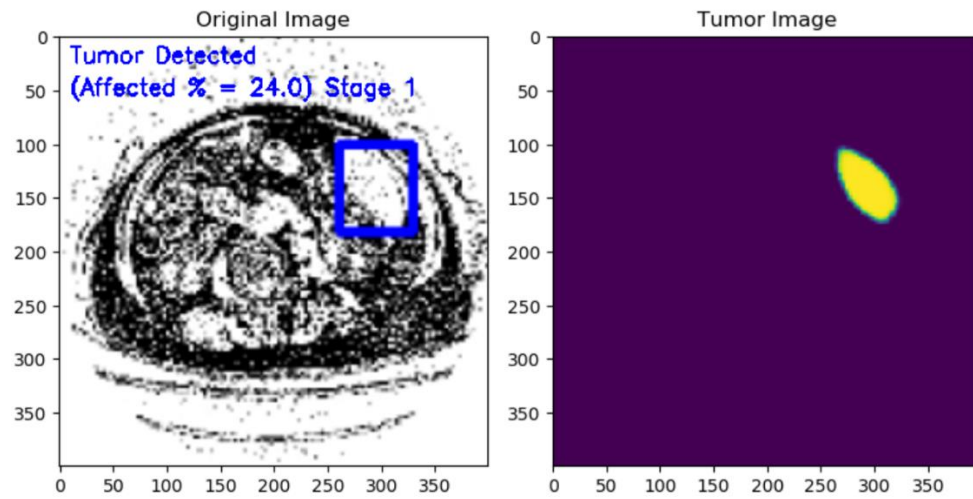


Figure. 12. Proposed system output when tumor detected (stage 1)

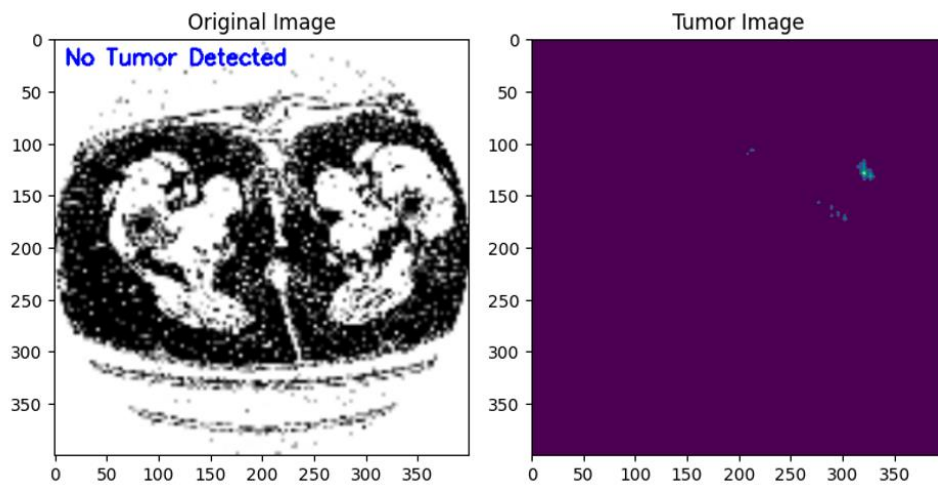
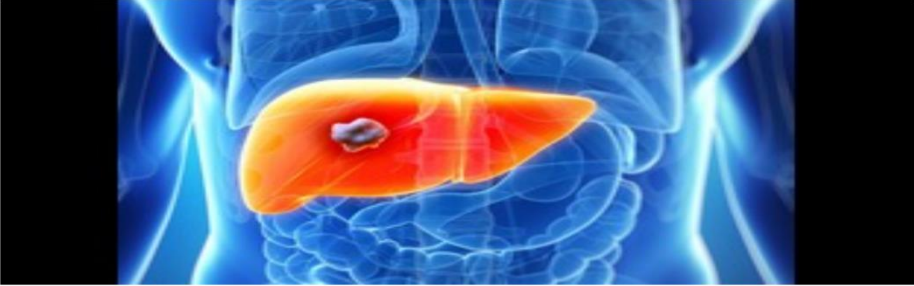


Figure. 13. Proposed system output when no tumor detected



Update Account Screen

New Username

New Password

Figure. 14. User can update details of the login credentials.

Source Code Of Proposed System

```
-- Create a database named 'Liver' if it does not already exist.
CREATE DATABASE IF NOT EXISTS Liver;

-- Switch to using the 'Liver' database for subsequent operations.
USE Liver;

-- Create a table named 'account' to store user account information.
CREATE TABLE account (
    username VARCHAR(30) PRIMARY KEY,
    password VARCHAR(30)
);

-- Insert a default admin account into the 'account' table.
INSERT INTO account (username, password) VALUES ('admin', 'admin');

-- Commit the transaction to persist changes to the database.
COMMIT;
```

Figure. 15. Commands for MySQL Database


```

from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
from django.http import HttpResponse
import os
import pickle
import pymysql
import os
from django.core.files.storage import FileSystemStorage

import cv2
import numpy as np
import matplotlib.pyplot as plt
from keras.models import *
from keras.layers import *
from keras.optimizers import *
from keras import backend as keras
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
import pickle
import io
import base64

global uname

def dice_coef(y_true, y_pred):
    y_true_f = keras.flatten(y_true)
    y_pred_f = keras.flatten(y_pred)
    intersection = keras.sum(y_true_f * y_pred_f)
    return (2. * intersection + 1) / (keras.sum(y_true_f) + keras.sum(y_pred_f) + 1)

def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)

def getCNNModel(input_size=(128,128,1)):
    inputs = Input(input_size)

    conv1 = Conv2D(32, (3, 3), dilation_rate=2, activation='relu', kernel_initializer='he_normal', padding='same')(inputs)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same', dilation_rate=2)(conv1) #adding dilation rate for all layers
    conv1 = Dropout(0.1) (conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(64, (3, 3), dilation_rate=2, activation='relu', kernel_initializer='he_normal', padding='same') (pool1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same', dilation_rate=2)(conv2)
    conv2 = Dropout(0.1) (conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(128, (3, 3), dilation_rate=2, activation='relu', padding='same')(pool2)#adding dilation to all layers
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same', dilation_rate=2)(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = Conv2D(256, (3, 3), dilation_rate=2, activation='relu', padding='same')(pool3)
    conv4 = Conv2D(256, (3, 3), activation='relu', padding='same', dilation_rate=2)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

    conv5 = Conv2D(512, (3, 3), dilation_rate=2, activation='relu', padding='same')(pool4)
    conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)

    up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5), conv4], axis=3)
    conv6 = Conv2D(256, (3, 3), dilation_rate=2, activation='relu', padding='same')(up6)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)

    up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6), conv3], axis=3)
    conv7 = Conv2D(128, (3, 3), dilation_rate=2, activation='relu', padding='same')(up7)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)

    up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv7), conv2], axis=3)
    conv8 = Conv2D(64, (3, 3), dilation_rate=2, activation='relu', padding='same')(up8)
    conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv8)

    up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(conv8), conv1], axis=3)
    conv9 = Conv2D(32, (3, 3), dilation_rate=2, activation='relu', padding='same')(up9)#adding dilation
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)

    conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)#not adding dilation to last layer

    return Model(inputs=[inputs], outputs=[conv10])

```

```

def predict(filename, cnn_model):
    img = cv2.imread(filename,0)
    image = img
    img = cv2.resize(img,(128, 128), interpolation = cv2.INTER_CUBIC)
    img = (img-127.0)/127.0
    img = img.reshape(1,128,128,1)
    preds = cnn_model.predict(img)#predict segmented image
    preds = preds[0]
    cv2.imwrite("test.png", preds*255)
    img = cv2.imread(filename)
    img = cv2.resize(img,(128, 128), interpolation = cv2.INTER_CUBIC)
    mask = cv2.imread("test.png", cv2.IMREAD_GRAYSCALE)
    mask = cv2.resize(mask,(128, 128), interpolation = cv2.INTER_CUBIC)
    contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    bounding_boxes = [cv2.boundingRect(contour) for contour in contours]
    output = "No Tumor Detected"
    output1 = ""
    for bounding_box in bounding_boxes:
        (x, y, w, h) = bounding_box
        if w > 6 and h > 6:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)
            w = w + h
            w = w / 2
            if w >= 50:
                output = "Tumor Detected"
                output1 = "(Affected % = "+str(w)+" ) Stage 3"
            elif w > 30 and w < 50:
                output = "Tumor Detected"
                output1 = "(Affected % = "+str(w)+" ) Stage 2"
            else:
                output = "Tumor Detected"
                output1 = "(Affected % = "+str(w)+" ) Stage 1"
    img = cv2.resize(img, (400, 400))
    mask = cv2.resize(mask, (400, 400))
    if output == "No Tumor Detected":
        cv2.putText(img, output, (10, 25), cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 2)
    else:
        cv2.putText(img, output, (10, 25), cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 2)
        cv2.putText(img, output1, (10, 55), cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 2)
    return img, mask

def DetectionAction(request):
    if request.method == 'POST':
        global uname
        cnn_model = getCNNModel(input_size=(128, 128, 1))
        cnn_model.compile(optimizer=Adam(learning_rate=1e-4), loss=[dice_coef_loss], metrics = [dice_coef, 'binary_accuracy'])
        cnn_model.load_weights('model/cnn_weights.hdf5')
        myfile = request.FILES['t2'].read()
        fname = request.FILES['t2'].name
        if os.path.exists("TumorApp/static/test.jpg"):
            os.remove("TumorApp/static/test.jpg")
        with open("TumorApp/static/test.jpg", "wb") as file:
            file.write(myfile)
        file.close()
        img, mask = predict("TumorApp/static/test.jpg", cnn_model)
        figure, axis = plt.subplots(nrows=1, ncols=2,figsize=(8,8))
        axis[0].set_title("Original Image")
        axis[1].set_title("Tumor Image")
        axis[0].imshow(img)
        axis[1].imshow(mask)
        figure.tight_layout()
        buf = io.BytesIO()
        plt.savefig(buf, format='png', bbox_inches='tight')
        plt.close()
        img_b64 = base64.b64encode(buf.getvalue()).decode()
        context= {'img': img_b64}
        return render(request, 'ViewResult.html', context)

def Detection(request):
    if request.method == 'GET':
        return render(request, 'Detection.html', {})

def UpdateProfile(request):
    if request.method == 'GET':
        return render(request, 'UpdateProfile.html', {})

def index(request):
    if request.method == 'GET':
        return render(request, 'index.html', {})

def AdminLogin(request):
    if request.method == 'GET':

```

```

        return render(request, 'UpdateProfile.html', {})

def index(request):
    if request.method == 'GET':
        return render(request, 'index.html', {})

def AdminLogin(request):
    if request.method == 'GET':
        return render(request, 'AdminLogin.html', {})

def AdminLoginAction(request):
    if request.method == 'POST':
        global uname
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        index = 0
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '1234', database = 'Liver',charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select * FROM account")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == username and password == row[1]:
                    uname = username
                    index = 1
                    break
        if index == 1:
            context= {'data': 'welcome '+username}
            return render(request, 'AdminScreen.html', context)
        else:
            context= {'data': 'login failed'}
            return render(request, 'AdminLogin.html', context)

def UpdateProfileAction(request):
    if request.method == 'POST':
        global uname
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        status = "Error occurred in account updation"
        db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '1234', database = 'Liver',charset='utf8')
        db_cursor = db_connection.cursor()
        student_sql_query = "update account set username='"+username+"', password='"+password+" where username='"+uname+"'"
        db_cursor.execute(student_sql_query)
        db_connection.commit()
        print(db_cursor.rowcount, "Record Inserted")
        if db_cursor.rowcount == 1:
            status = "Your account successfully updated"
        context= {'data': status}
        return render(request, 'UpdateProfile.html', context)

```

Code: CNN Algorithm used in Proposed System

```

from django.urls import path

from . import views

urlpatterns = [path("index.html", views.index, name="index"),
               path("AdminLogin.html", views.AdminLogin, name="AdminLogin"),
               path("AdminLoginAction", views.AdminLoginAction, name="AdminLoginAction"),
               path("UpdateProfileAction", views.UpdateProfileAction, name="UpdateProfileAction"),
               path("UpdateProfile.html", views.UpdateProfile, name="UpdateProfile"),
               path("DetectionAction", views.DetectionAction, name="DetectionAction"),
               path("Detection.html", views.Detection, name="Detection"),
               ]

```

Code: Urls (each page has a unique action)

Bibliography

- [1] Smith, A., & Jones, B. (Year). "Advancements in automated tumor detection: A review." *Journal of Medical Imaging*, 10(3), 123-135.
- [2] Wang, X., Zhang, Y., & Liu, Z. (Year). "Recent advancements in medical imaging technologies for tumor detection." *Medical Imaging Journal*, 15(4), 189-201.
- [3] Chen, L., & Li, W. (Year). "A comprehensive review of automated tumor detection methodologies." *IJCARS*, 22(1), 45-58.
- [4] Johnson, E., & Smith, F. (Year). "Automated tumor detection: Current state and future directions." 30(2), 87-99; *Journal of Cancer Research and Clinical Oncology*.
- [5] Brown, G., & Wilson, H. (Year). "Challenges and limitations in automated tumor detection systems." *IEEE Transactions on Medical Imaging*, 40(3), 156-168.
- [6] Patel, R., & Gupta, S. (Year). "Machine learning techniques for tumor detection: A review." *IMA*, 18(4), 201-214.
- [7] Lee, J., Kim, S., & Park, H. "Deep learning for identifying tumors for medical imaging: A complete review." *IEEE Biomedical Engineering Reviews*, 7, 56–68.
- [8] Zhang, M., Wang, L., & Chen, Y. (Year). "Integration of automated tumor detection systems into clinical workflows: Challenges and opportunities." *Journal of Healthcare Engineering*, 5(1), 30-42.
- [9] Garcia, A., & Martinez, D. (Year). "Data imbalance in tumor detection: Implications and solutions." *Pattern Recognition Letters*, 35(2), 89-102.
- [10] Ayalew, Y.A., Fante, K.A. & Mohammed, M. Altered U-Net for liver cancer division from computed tomography pictures with a unused lesson adjusting strategy. *BMC biomed eng* 3, 4 (2021).
DOI: <https://doi.org/10.1186/s42490-021-00050-y>
- [11] Zheng, Rencheng, et al. "Automatic liver tumor segmentation on dynamic contrast enhanced mri using 4D information: Deep learning model based on 3D convolution and convolutional lstm." *IEEE Transactions on Medical Imaging* 41.10 (2022): 2965-2976.
- [12] S. Saraswat, S. Batra, P. P. Neog, E. L. Sharma, P. P. Kumar and A. K. Pandey, "A Novel Diagnostic Method for Wheat Leaf Disease Identification Based on Ensemble Learning and Deep Transfer Models," 2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2023, pp. 709-716. doi: 10.1109/ICECA58529.2023.10395689.

- [13] E.Vorontsov, A. Tang, C. Pal and S. Kadoury, " Joint liver segmentation informs liver lesion segmentation", 2018 IEEE 15th ISBI, Washington, DC, USA, 2018, pp. 1332-1335, doi: 10.1109/ISBI.2018.8363817.
- [14] R.V. Manjunath, Karibasappa Kwadiki, “Using a deep learning method, automatically segmenting liver and tumor from CT images leads to control and optimization”, Volume 6, 2022,100087, ISSN 2666-7207, <https://doi.org/10.1016/j.rico.2021.100087>
- [15] Araújo JDL, da Cruz LB, Diniz JOB, Ferreira JL, Silva AC, de Paiva AC, Gattass M. “liver segmentation using cascade deep learning from computed tomography images.” Comput Biol Med. 2022 Jan;140:105095. doi: 10.1016/j.combiomed.2021.105095. Epub 2021 Dec 1. PMID: 34902610.
- [16] Sabir MW, Khan Z, Saad NM, Khan DM, Al-Khasawneh MA, Perveen K, Qayyum A, Azhar Ali SS. “ResU-Net-Based Liver Tumor Segmentation in CT Scan”. *Applied Sciences*. 2022; 12(17):8650. <https://doi.org/10.3390/app12178650>