

# **A MINOR PROJECT REPORT**

ON

## **Liver Tumor Segmentation using U-Net**

*Submitted in partial fulfilment of the requirement  
for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING  
(Artificial Intelligence & Machine Learning)**

**BY**

**Arisela Meghana (20P61A6602)**

**Cheripally Jitendra Kumara Prasad (20P61A6611)**

**Toomu Sahithi (20P61A6652)**

*Under the esteemed guidance of*

***Mrs. P. Navya***

***Assistant Professor***

Counselling Code : **VBIT**



**VIGNANA BHARATHI**  
Institute of Technology

®

(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA & NAAC-A Grade, Affiliated to JNTUH)



Counselling Code : **VBIT**

**VIGNANA BHARATHI**  
Institute of Technology®

(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA & NAAC-A Grade, Affiliated to JNTUH)

Aushapur (V), Ghatkesar (M), Hyderabad, Medchal – Dist, Telangana – 501 301.

**DEPARTMENT  
OF  
COMPUTER SCIENCE & ENGINEERING  
(Artificial Intelligence & Machine Learning)**

**CERTIFICATE**

*This is to certify that the major project titled “Liver Tumor Segmentation using U-Net” submitted by Arisela Meghana (20P61A6602), Cheripally Jitendra Kumara Prasad (20P61A6611), Toomu Sahithi (20P61A6652) in B. Tech IV-I semester Computer Science & Engineering (Artificial Intelligence & Machine Learning) is a record of the bonafide work carried out by them.*

*The results embodied in this report have not been submitted to any other University for the award of any degree.*

**INTERNAL GUIDE**

**Mrs. P. Navya**

**HEAD OF THE DEPARTMENT**

**Dr. K. Shirisha Reddy**

**PORJECT CO-ORDINATOR**

**Mrs. P. Navya**

**EXTERNAL EXAMINER**

## **DECLARATION**

We, **Arisela Meghana, Cheripally Jitendra Kumara Prasad, Toomu Sahithi**, bearing hall ticket numbers **20P61A6602, 20P61A6611, 20P61A6652** hereby declare that the minor project report entitled “**Liver Tumor Segmentation using U-Net**” under the guidance of **Mrs. P. Navya**, Department of Computer Science Engineering (Artificial Intelligence & Machine Learning), **Vignana Bharathi Institute of Technology, Hyderabad**, have submitted to Jawaharlal Nehru Technological University Hyderabad, Kukatpally, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence & Machine Learning).

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**Arisela Meghana (20P61A6602)**

**Cheripally Jitendra Kumara Prasad (20P61A6611)**

**Toomu Sahithi (20P61A6652)**

## **ACKNOWLEDGEMENT**

We are extremely thankful to our beloved Chairman, **Dr. N. Goutham Rao** and secretary, **Dr. G. Manohar Reddy** who took keen interest to provide us the infrastructural facilities for carrying out the project work. Self-confidence, hard work, commitment and planning are essential to carry out any task. Possessing these qualities is sheer waste, if an opportunity does not exist. So, we wholeheartedly thank **Dr. P. V. S. Srinivas**, Principal, and **Dr. K. Shirisha Reddy**, Head of the Department, Computer Science and Engineering (Artificial Intelligence & Machine Learning) for their encouragement and support and guidance in carrying out the project.

We would like to express our indebtedness to the project coordinator, **Mrs. P. Navya**, Assistant Professor, Department of CSE (Artificial Intelligence & Machine Learning) for her valuable guidance during the course of project work.

We thank our Project Guide, **Mrs. P. Navya**, for providing us with an excellent project and guiding us in completing our mini project successfully.

We would like to express our sincere thanks to all the staff of Computer Science and Engineering (Artificial Intelligence & Machine Learning), VBIT, for their kind cooperation and timely help during the course of our project. Finally, we would like to thank our parents and friends who have always stood by us whenever we were in need of them.

## **ABSTRACT**

Liver cancer is a leading cause of cancer-related deaths worldwide. Detecting it early and planning treatments are crucial, and this often involves using computed tomography (CT) imaging. However, manually analyzing numerous CT scans can be time-consuming and prone to errors. To address this, automated semantic segmentation using deep learning is employed to precisely identify malignant liver tumors, aiding radiologists in their diagnosis.

To achieve liver tumor segmentation in CT scans, we constructed a U-Net, a deep learning architecture. The model is trained and evaluated on a dataset comprising 130 CT scans from The Liver Tumor Segmentation Benchmark (LiTS), which includes expert radiologist annotations. This approach enables the rapid and accurate identification of liver tumors, improving the efficiency and accuracy of the diagnostic process for liver cancer.

### **Keywords:**

Liver Tumor, Semantic Segmentation, U-Net, Deep Learning, LiTS, CT scans.

# **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## **(Artificial Intelligence & Machine Learning)**

### **VISION**

To achieve global standards of quality in technical education with the help of advanced resources and automated tools to bridge the gap between industry and academia.

### **MISSION**

- Build the students technically competent on global arena through effective teaching learning process and world-class infrastructure.
- Inculcate professional ethics, societal concerns, technical skills and life-long learning to succeed in multidisciplinary fields.
- Establish competency center in the field of Artificial Intelligence and Machine Learning with the collaboration of industry and innovative research.

### **PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**The graduates of Computer Science and Engineering with Specialization in Artificial Intelligence & Machine Learning will be able to:**

**PEO 1: Domain Knowledge:** Impart strong foundation in basic sciences, mathematics, Engineering and emerging areas by Advanced tools and Technologies.

**PEO 2: Professional Employment:** Develop Professional skills that prepare them for immediate employment in industry, government, entrepreneurship and Research.

**PEO 3: Higher Degrees:** Pursue higher studies and acquire master's and research.

**PEO 4: Engineering Citizenship:** Communicate and work effectively, engage in teamwork, achieve professional advancement, exhibit leadership skills, and ethical attitude with a sense of social responsibility.

**PEO 5: Lifelong Learning:** Lead in their field and respond to the challenges of an ever-changing environment with the most current knowledge and technology.

## **PROGRAM OUTCOMES (POs)**

**The graduates of Computer Science and Engineering with Specialization in Artificial Intelligence & Machine Learning will be able to:**

- 1. Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem Analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- 3. Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
- 4. Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
- 6. The Engineer and Society:** Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to professional engineering practice.
- 7. Environment and Sustainability:** Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.
- 9. Individual and Teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend

and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

- 11. Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary Environments.
- 12. Life-long Learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

**The graduates of Computer Science and Engineering with Specialization in Artificial Intelligence & Machine Learning will be able to:**

**PSO1:** Understand and Apply Multi-Disciplinary and core concepts with emerging technologies for sustaining (endorse) with the Dynamic Industry Challenges.

**PSO2:** Design Automated Applications in Machine Learning, Deep Learning, Natural Language Processing and Relevant Emerging areas for visualizing, interpreting the datasets.

**PSO3:** Develop Computational Knowledge, project and Interpersonal skills using innovative tools for finding an elucidated solution of the real-world problems and societal needs.

### **COURSE OBJECTIVES**

1. Identify and compare technical and practical issues related to the area of course specialization.
2. Outline annotated bibliography of research demonstrating scholarly skills.
3. Prepare a well-organized report employing elements of technical writing and critical thinking.
4. Demonstrate the ability to describe, interpret and analyse technical issues and develop competence in presenting.

### **COURSE OUTCOMES**

CO1 - Demonstrate a sound technical knowledge of their selected topic.

CO2 - Undertake problem identification, formulation, and solution.



CO3- Design engineering solutions to complex problems utilizing a system approach.

CO4- Organize a detailed literature survey and build a report with respect to technical publications.

CO5- Make use of recent technology for building technical reports.

### **PROJECT OBJECTIVES**

1. Semantic image segmentation of tumors in livers.
2. Implementation of U-Net for competent segmentation.
3. Robust and accurate results.

### **PROJECT OUTCOMES**

1. **Accurate Liver Tumor Segmentation:** The project will achieve precise and reliable segmentation of liver tumors within medical images using the U-Net architecture. This ensures the accurate delineation of tumor boundaries.
2. **High IoU and Dice Coefficient Scores:** The project will aim for high Intersection over Union (IoU) and Dice Coefficient scores, reflecting the effectiveness of tumor segmentation, which is crucial for clinical applications and diagnosis.
3. **Robust and Generalizable Model:** The developed U-Net model will demonstrate robustness in handling variations in liver tumor images and generalize well to segment tumors in diverse patient cases.
4. **Optimized Data Preprocessing:** The project will involve effective data preprocessing techniques to enhance the quality of input images, leading to improved tumor segmentation results.
5. **Research Contribution:** The project outcomes will contribute to the field of medical image analysis and liver tumor segmentation, potentially advancing the development of automated tools for healthcare professionals.

## **PROJECT MAPPING**

<b>LIVER TUMOR SEGMENT A-TION USING U- NET</b>	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
	✓	✓	✓	✓	✓	✓		✓	✓		✓		✓	✓	✓

## **CO – PO MAPPING**

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2	PS O3
CO1	✓	✓	✓	✓	✓								✓	✓	
CO2		✓	✓	✓					✓				✓		✓
CO3			✓		✓	✓			✓				✓		✓
CO4		✓		✓	✓					✓			✓	✓	
CO5					✓					✓				✓	

## **TABLE OF CONTENTS**

<b><u>CHAPTER</u></b>	<b><u>PAGE NO</u></b>
<b>I. Title</b>	i
<b>II. Certificate</b>	ii
<b>III. Declaration</b>	iii
<b>IV. Acknowledgement</b>	iv
<b>V. Abstract</b>	v
<b>VI. List of Figures</b>	xii
<b>VII. List of Abbreviations</b>	xiii
<b>1. Introduction</b>	1
1.1 Existing System	
1.2 Proposed System	
1.3 Aim and Objective	
1.4 Scope	
<b>2. Literature Survey</b>	5
<b>3. Design</b>	10
3.1 Hardware Requirements	
3.2 Software Requirements	
3.3 System Architecture	
3.4 Model Architecture	
3.5 Algorithms	
3.6 Libraries	
3.7 Dataset	
3.8 Data Flow Diagram	
<b>4. Implementation</b>	19
4.1 Data Loading and Preprocessing	
4.2 Data Generator and Augmentation	
4.3 U-Net	
4.4 Testing and Evaluation	
4.5 Accuracy Calculation	
<b>5. Results and Discussion</b>	31
<b>6. Conclusion and Future Enhancement</b>	35
<b>References</b>	37

## **LIST OF FIGURES**

<b>S. No.</b>	<b>Figure Name</b>	<b>Page No.</b>
1.	System Architecture	12
2.	U-Net Model Architecture	13
3.	Dice Coefficient	14
4.	Intersection over Union	15
5.	Data Flow Diagram	17
6.	Data Classification	22
7.	Ground Truths to Binary Mask Conversion	24
8.	Training Metrics Plot	32
9.	Visualisation of Predictions	33

## **LIST OF ABBREVIATIONS**

<b>S. No.</b>	<b>Term</b>	<b>Abbreviations</b>
1.	API	Application Programming Interface
2.	BCE	Binary Cross Entropy
3.	CNN	Convolutional Neural Network
4.	Conv2D	Convolutional 2-Dimensional Layer
5.	CT	Computed Topography
6.	cv2	Open Computer Vision
7.	LiTS	Liver Tumor Segmentation Benchmark
8.	MCG	Multi-Scale Candidate Generation
9.	MICCAI	Medical Image Computing and Computer Assisted Intervention Society
10.	MRI	Magnetic Resonance Imaging
11.	PET	Positron Emission Tomography
12.	ReLU	Rectified Linear Unit
13.	ROI	Region of Interest
14.	RMSE	Root Mean Square Error
15.	IoU	Intersection over Union
16.	ISBI	IEEE International Symposium on Biomedical Imaging
17.	TCIA	The Cancer Imaging Archive
18.	FRN	Fractal Residual Network

# CHAPTER-1

# 1. INTRODUCTION

Liver cancer often presents a considerable challenge due to its aggressive nature and the limited treatment options available at advanced stages. Therefore, early diagnosis and accurate tumor characterization are essential for improving patient outcomes. To achieve these goals, medical imaging plays a pivotal role, with Computed Tomography (CT) scans standing out as a critical diagnostic tool. However, the manual analysis of numerous CT scans can be a time-consuming and error-prone task, particularly when dealing with intricate structures like the liver. To overcome these challenges and enhance the diagnostic process, researchers have turned to cutting-edge technologies like deep learning and artificial intelligence to develop automated solutions for the precise identification and segmentation of liver tumors within CT images.

In response to this challenge, this project focuses on the development and implementation of a deep learning-based approach for the semantic segmentation of liver tumors within CT images. We harness the power of U-Net, a deep learning architecture renowned for its prowess in image segmentation tasks. The U-Net architecture combines an encoder for contextual understanding and a decoder for precise localization, aided by skip connections that facilitate feature flow. With its potential for end-to-end training and adaptability for medical imaging applications, U-Net offers a promising foundation for the segmentation of liver tumors in CT volumes.

## 1.1 Existing System

The existing model used “An automated slice sorting technique for multi-slice computed tomography liver cancer images using convolutional network”. Which focuses on the multi-organ classification, such as the kidney, bladder, brain, lungs, and bones, of 3D CT images of liver cancer-suspect patients using a CNN. The dataset employed for validation comprises 63,503 CT images of liver cancer patients obtained from The Cancer Imaging Archive (TCIA).

The CNN is designed to classify these CT images for liver cancer detection, and various performance metrics, including accuracy, precision, sensitivity, specificity, true positive rate, false negative rate, and F1 score, are computed to evaluate the classification result. The overall test accuracy for the dataset consisting of data augmented volume slices is reported to be 93.1%. However, the system achieves impressive accuracy in classifying liver cancer using CT images it is better considering relative computational resources to enhance its practical applicability in clinical settings.

## 1.2 Proposed System

Considering the drawbacks of the existing system, we proposed U-Net architecture, a deep learning model for our project. The U-Net architecture is specifically designed for image segmentation tasks and offers several advantages over a CNN.

U-Net's effectiveness for this task lies in its unique structure, which combines an encoder and a decoder with skip connections. The encoder component of U-Net acts as a feature extractor. It processes the input CT images and progressively reduces their spatial dimensions through a series of convolutional layers and pooling operations. The decoder, which is symmetrical to the encoder, reverses the down sampling process by employing up-sampling layers, also known as transpose convolutions. These layers effectively increase the spatial resolution of the feature maps, allowing the network to recover fine-grained details. Skip connections connect corresponding layers between the encoder and decoder, enabling the network to combine both local and global information. This means that while the decoder is up-sampling and recovering spatial resolution, it also benefits from the high-level context learned by the encoder. By leveraging U-Net's end-to-end training capabilities and optimizing hyperparameters, we aim to achieve state-of-the-art accuracy in liver segmentation, contributing significantly to the field of medical imaging and enhancing the diagnosis and treatment planning processes for liver cancer patients.

## 1.3 Aim and Objective

- i. **Accurate Tumor Localization:** The primary objective is to accurately identify the location of liver tumors within CT scans. U-Net aims to provide precise tumor segmentation maps that delineate tumor boundaries, helping radiologists pinpoint tumor locations.
- ii. **Scalability and Generalizability:** Our model can scale to handle large datasets and is capable of generalizing its learned knowledge to different healthcare institutions and regions.
- iii. **Interpretability and Trustworthiness:** Developed method makes the model's predictions interpretable and explainable to healthcare professionals, building trust in AI-assisted diagnosis and treatment planning.



- iv. **Population Health Insights:** Leverage the segmentation results to contribute insights into population-level health management, such as identifying trends in liver tumor prevalence, early detection strategies, and treatment outcomes.
- v. **Reduction of Manual Effort:** Reducing the manual effort required by radiologists, can save time and reduce the risk of human errors. Ultimately improving the efficiency of diagnosis and treatment.

## 1.4 Scope

The future scope for liver tumor segmentation using U-Net is dynamic and holds substantial potential for advancements in medical imaging and healthcare.

- i. **Clinical Adoption and Integration:** The future objectives primarily target a smooth transition from U-Net-based liver tumor segmentation models into standard clinical practice. For clinical trials, validation, and regulatory approvals, cooperation with healthcare facilities will be crucial.
- ii. **Multi-Modality Imaging:** Expand the use of liver tumor segmentation models to accommodate multiple imaging modalities, such as fusion imaging, Positron Emission Tomography (PET), and magnetic resonance imaging (MRI), enabling a more all-encompassing strategy for liver cancer diagnosis and treatment planning.
- iii. **Multi-Organ Segmentation:** Extended learning to the division of several organs, such as the kidney, bladder, brain, lungs, and bones. This facilitates simple integration in front-line medical care.
- iv. **Real-Time Decision Support:** Establish decision support tools in real time that instantly give radiologists feedback while they interpret images, improving the efficiency and accuracy of diagnosis.

# **CHAPTER-2**

## **2. LITERATURE SURVEY**

**Paper I - An automated slice sorting technique for multi-slice computed tomography liver cancer images using convolutional network. (2021)**

**Authors – Amandeep Kaur, Ajay Pal Singh Chahuan, Ashwani Kumar Aggarwal**

Our inspiration for liver tumor segmentation is based on this paper. The authors describe an automated technique that uses a convolutional neural network to classify several organs in 3D computed tomography pictures of patients with liver cancer. The aim is to assist radiologists in concentrating on the most pertinent CT slices to identify liver cancer and any potential metastases to the lungs and bones. The technique extracts features from the CT volume images using a CNN architecture with convolution and pooling layers. To increase the robustness of the model, data augmentation methods such as Gaussian filtering and histogram equalization are applied to the 63,503 CT slice dataset. To help determine if the cancer has spread, the slices are categorized based on whether they show signs of liver, lungs, or bones. The CNN model achieves high accuracy of 99.1% on augmented validation data. Key metrics like precision, recall, specificity and F1-score are computed to evaluate performance.

**Paper II - Automatic Liver Lesion Segmentation Using a Deep Convolutional Neural Network Method. (2017)**

**Author – Xiao Han**

We familiarized ourselves with the metrics for image segmentation, like the Dice Similarity Coefficient. The method for automatically segmenting liver lesions in CT images using a deep convolutional neural network is presented in this research. Long-range U-Net connections and short-range ResNet residual connections are combined to create a 32-layer 2.5D DCNN model. The model generates a lesion segmentation map for the central slice based on input from nearby axial slices. The LiTS dataset is used to train two models: one for detailed lesion segmentation in the liver region and one for coarse liver segmentation. The method ranks first in the challenge with a Dice score of 0.67 on the 70 LiTS test cases. According to the study, DCNNs can segment lesions even in the absence of manually created characteristics. Restrictions include limited precision and memory limitations that hinder 3D analysis. But the model shows good potential for accurate automatic lesion segmentation with further architecture enhancements and more training data.

### **Paper III - Neural Network-Based Automatic Liver Tumor Segmentation with Random Forest-Based Candidate Filtering. (2017)**

**Authors - Grzegorz Chlebus, Hans Meine, Jan Hendrik Moltz, Andrea Schenk (2017)**

The dataset we have taken into consideration for our project is the one employed in this paper. Using a cascaded deep learning methodology, the research provides an automated method for liver tumor segmentation in CT scans. First, 179 instances are used to train a 2D U-net to segment the hepatic region. In the liver mask, tumor candidates are then found using an additional 2D U-net that has been trained on 127 instances. Shape and texture features are extracted for each tumor candidate and filtered using a random forest classifier to minimize false positives. The pipeline ranks second with a Dice similarity coefficient of 0.65 after being tested on 70 test cases from the LiTS challenge. The work shows how CNN-based tumor segmentation performance can be enhanced by limiting the search space and candidate filtering. Some drawbacks include misclassified core tumor locations and missing malignancies due to imprecise liver masking. The technique has the potential to detect liver tumors completely automatically, but accuracy might be increased even further with improvements made to the design and training methodology.

### **Paper IV – Hierarchical Convolutional-Deconvolutional Neural Networks for Automatic Liver and Tumor Segmentation. (2017)**

**Author - Yading Yuan**

The study describes a hierarchical deep learning method for automatic liver and tumor segmentation in CT volumes using convolutional-deconvolutional neural networks (CDNs). Three CDNNs are trained in a sequential fashion: one for fine segmentation of the liver within the liver ROI, one for coarse localization of the liver, and one for tumor segmentation utilizing the liver mask as an extra input. The approach receives a mean liver dice of 0.963 (1st place) and a tumor dice of 0.657 (5th place) on the LiTS 2017 challenge dataset. Third place went to the tumor burden RMSE of 0.017. Robustness was enhanced by data augmentation and a Jaccard distance loss function. The framework operates at a speed of 33 seconds per volume. One of the limitations is the variable performance of tumor segmentation based on heterogeneity. However, the hierarchical CDNN method achieves the best results on the public LiTS leaderboard, suggesting promise for precise automatic liver tumor delineation.

## **Paper V - Liver Tumor Segmentation Based on Multi-Scale Candidate Generation and Fractal Residual Network. (2019)**

**Authors - Zhiqi Bai, Huiyan Jiang, Siqi Li<sup>1</sup>, and Yu-Dong Yao**

The technique for automatic liver tumor segmentation from CT volumes is presented in the paper. It makes use of active contour modelling, 3D fractal residual networks (FRN), and multi-scale candidate generation (MCG). First, 3D U-Net is used to segment the liver regions. Within the liver ROIs, tumor candidates are created utilizing neighbourhood data and multi-scale superpixels. The candidates are further divided into tumor and non-tumor categories using a 112-layer 3D FRN with fractal and residual connections. Intermediate supervision is provided by several auxiliary classifiers. Active contours refine the tumor borders after multi-scale fusion. The method achieves a dice per case of 0.67 for lesions on the 3D-IRCADb dataset. The sensitivity and accuracy are enhanced by the hierarchical MCG and FRN. Restrictions include missing tiny tumors and merging adjacent tumors.

### **Summary:**

Research on liver tumor segmentation from medical imaging is ongoing and has applications in cancer diagnosis, therapy planning, and screening. CNN-based deep learning techniques have recently demonstrated encouraging results on this challenge. The evaluated publications show how to use CNN design variations to precisely identify liver cancers from CT and MRI data. Using CNNs to classify image slices and identify metastases is one of the key concepts explored. Other ideas include generating tumor candidates using techniques like multi-scale superpixels, combining coarse segmentation of the entire liver with detailed segmentation of tumors, segmenting livers and lesions sequentially using cascaded networks, using multi-modality datasets, and incorporating post-processing steps like conditional random fields.

Preprocessing methods that enhance model robustness and generalization include rotations, flips, and elastic deformations of the data. Pre-extracted features are not as good as end-to-end training using composite loss functions that optimize both accuracy and overlap. The efficacy of deep learning techniques is demonstrated through quantitative evaluation on datasets such as LiTS Challenge and 3D-IRCADb, using metrics like dice coefficient, intersection over union, sensitivity, and specificity. The best techniques achieve dice scores of 0.93-0.96 for livers and 0.65-0.67 for tumors.

Although encouraging, issues with diffuse small tumor handling, computational effectiveness, and clinical practice adoption must be resolved for dependable real-world application. However, the quick development makes the automatic segmentation of liver tumors an interesting field for deep convolutional networks to be used in medical imaging. After careful consideration, we determined to use the U-Net architecture in our deep learning system to segment liver tumors. We are analyzing 130 CT scans as part of the Liver Tumor Segmentation Benchmark (LiTS) dataset.

# **CHAPTER-3**

## 3. DESIGN

### 3.1 Hardware Requirements

Deep learning models require massive processing power, gigabytes of training data, and exceptional performance to train over an extended period. To train U-Net for liver tumor segmentation, we needed a multicore Intel Core i5 10<sup>th</sup> generation processor, 16 GB of RAM, 15 GB of disk space for storing the dataset, and a 16 GB RAM graphics processing unit P100. The hardware requirements are the only exception to utilizing a trained model.

- Multi-core processor – Intel i5 10<sup>th</sup> generation or equivalent/higher
- Memory – 8 GB+ DDR4 (Recommended)
- Disk space – 2 GB+ ROM
- A Graphical Processing Unit (GPU)

### 3.2 Software Requirements

- **Python 3.x:** Python is the primary programming language used for deep learning and machine learning advancements. Python features numerous deep learning modules; it includes Tensorflow, Keras, SciPy, Sickit Learn, and more. Python 3 is the preferred version given to its integration capabilities and version support.
- **Anaconda Navigator:** With the help of the graphical user interface Anaconda Navigator, we can interact with environments and packages without having to execute conda instructions in a terminal window. When working with different environments with large bundles of packages, this will be quite beneficial.
- **Jupyter Notebook:** Jupyter Notebook is an open-source web application that allows machine learning engineers to create and share documents that include live code, equations, and other multimedia resources. This will be useful for debugging program at every stage, from data preparation to model deployment.

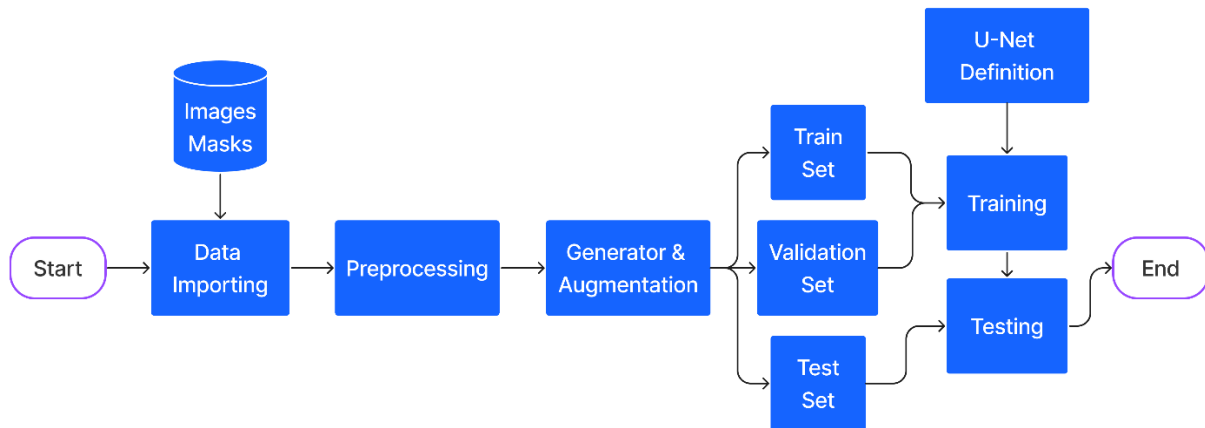
### 3.3 System Architecture

The presented liver tumor segmentation framework utilizes a deep convolutional neural network approach based on the U-Net architecture. The dataset is composed of up of aligned



mask slices and CT scan image slices that were taken from the LiTS challenge dataset. Following data loading, volume and slice number collected from filenames are used to order the image and mask paths. To diagnose cases of empty masks, liver only, or liver with tumors, the mask pixels are studied. From the total 2631 images the sorted data is then divided into training (2000), validation (200), and test (431) sets.

To expand the training data, generous image augmentation is applied including rotations, shifts, shears, zooms, and flips. The augmented image and mask pairs are generated in batches during training. The U-Net model comprises convolutional encoding layers to capture context, and transposed convolutional decoding layers to enable precise localization. It is trained end-to-end using a composite loss function based on binary cross-entropy and dice coefficient. Additional techniques including reduced learning rate scheduling, early stopping, and model checkpointing are employed to improve convergence and generalization. The trained model is evaluated on the test set using segmentation metrics like intersection-over-union and dice score. The approach demonstrates the viability of using deep convolutional neural networks for accurate liver tumor delineation from CT scans.



**Fig 1: System Architecture**

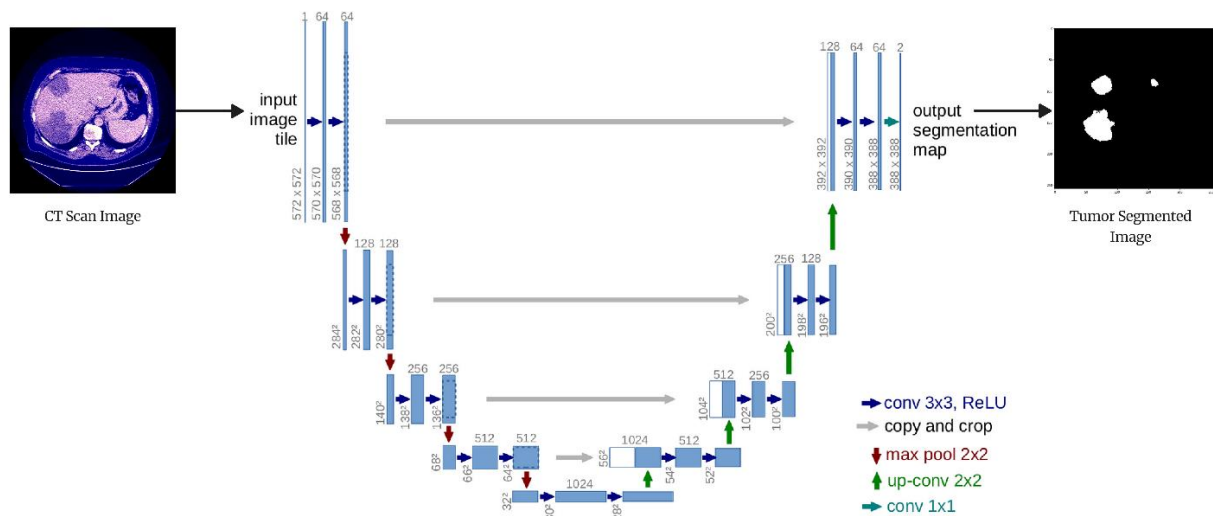
### 3.4 Model Architecture

The U-Net model follows an encoder-decoder structure, which is characterized by a contracting path (encoder) and an expansive path (decoder). In order to encode context and get hierarchical feature representations, the encoder route repeatedly applies 3x3 convolutions, ReLU (Rectified Linear Unit) activations, and 2x2 max pooling. This is a conventional convolutional neural

network architecture. There are five downsampling blocks in total, with 64, 128, 256, 512, and 1024 filters in each block.

The decoder pathway concatenates activations from the associated encoder block and recovers spatial information using 2x2 up-convolutions. By fusing localized data with encoded context, this allows for accurate localization. Mirroring the encoder components, two 3x3 convolutions and ReLU activations come after the upsampling. The decoder produces a segmentation map output with one filter and sigmoid activation after five upsample blocks with 512, 256, 128 and 64 filters.

Through combining localization and context information through skip connections between the encoder and decoder, the U-Net is able to precisely collect features for segmentation. The model uses additional regularization methods, such as batch normalization. Using a mixed loss function that includes dice coefficient terms and binary cross-entropy, the model is trained end-to-end on 2D CT slices. Rich feature representations are made possible by its 31 million trainable parameters, which allow for precise liver tumor delineation from CT scans.



**Fig 2: U-Net Model Architecture**

### 3.5 Algorithms

1. **Binary Cross-Entropy Dice loss:** The BCE Dice loss function is explicitly defined for better computation of images. We have implemented combination of Binary cross-entropy and Dice coefficient terms. Cross-entropy measures pixelwise differences while

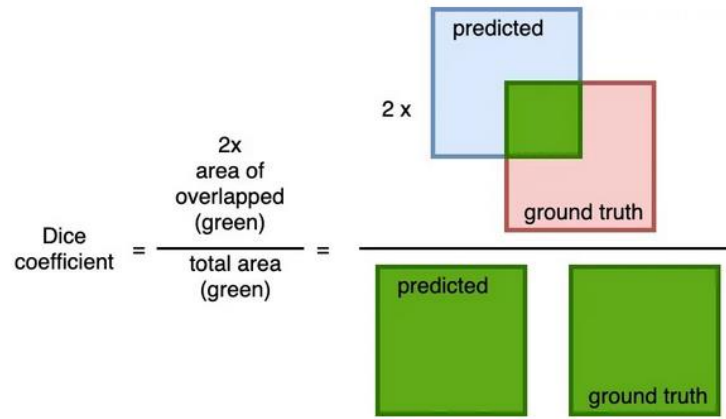
Dice coefficient evaluates region overlap. The combined loss balances pixelwise accuracy and region similarity. It is defined as:

$$\text{BCE\_dice\_loss} = 1 - \text{Dice}(\mathbf{y\_true}, \mathbf{y\_pred}) + \text{binary\_crossentropy}(\mathbf{y\_true}, \mathbf{y\_pred})$$

2. **Dice Coefficient:** The Dice coefficient measures the overlap between two segmentation masks and ranges from 0 to 1. It is defined as twice the intersection between the predicted and true masks divided by their combined areas:

$$\text{Dice}(\mathbf{X}, \mathbf{Y}) = 2 * |\mathbf{X} \cap \mathbf{Y}| / (|\mathbf{X}| + |\mathbf{Y}|)$$

Where X and Y are the predicted and true binary masks. A Dice of 1 indicates perfect overlap while 0 means no overlap. It provides a balanced metric accounting for both over and under-segmentation errors. Dice coefficient is commonly used to evaluate medical image segmentation models.



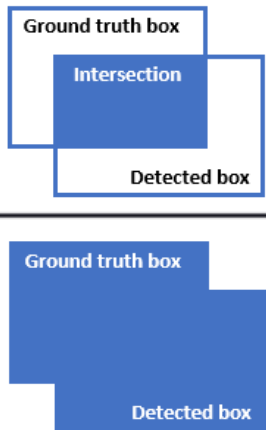
**Fig 3: Dice Coefficient**

3. **Binary Cross-Entropy:** Binary cross-entropy measures the pixel-wise difference between predicted and true binary masks. It is defined as:

$$\text{BCE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum(\mathbf{y} * \log(\hat{\mathbf{y}}) + (1-\mathbf{y}) * \log(1-\hat{\mathbf{y}}))$$

Where y is the true mask and  $\hat{y}$  is the predicted probability mask. A lower binary cross-entropy indicates fewer pixelwise errors in the predicted mask. It is often combined with regional losses like Dice coefficient to evaluate both pixel and region accuracy for segmentation. Binary cross-entropy punishes false predictions and rewards accurate ones during training.

- 4. Intersection over Union (IoU):** The IoU or Jaccard index measures the overlap between predicted and ground truth segmentation masks.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Intersection}}{\text{Union}}$$


**Fig 4: Intersection Over Union**

It is defined as the intersection area divided by the union area:

$$\text{IoU} = \text{Area of Intersection} / \text{Area of Union}$$

Where:

$$\text{Intersection} = y_{\text{true}} \cap y_{\text{pred}}$$

$$\text{Union} = y_{\text{true}} \cup y_{\text{pred}}$$

IoU scores range from 0 to 1, with 1 indicating perfect overlap. It is commonly used to evaluate semantic segmentation models.

- 5. Learning Rate Scheduling:** Learning rate scheduling reduces the learning rate over epochs according to a schedule. For example, the rate can be reduced by a factor every N epoch. This allows larger initial updates but finer updates later to enable convergence. We have defined a function that will return learning rate according to epoch. For 30 below epochs learning rate is defined as 1-e4 and after it will be decreased to 1-e5. This promoted the model to learn features more efficiently.
- 6. Adam Optimizer:** Adam is an adaptive gradient descent optimization algorithm suited for training deep neural networks. It maintains per-parameter learning rates that adapt based on exponential moving averages of the gradient (first moment) and squared gradient (second moment). The updates are calculated as:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$

$$\theta_t = \theta_{t-1} - \alpha * m_t / (\sqrt{v_t} + \epsilon)$$

Where  $m$  and  $v$  are first and second moment estimates,  $g$  is gradient,  $\alpha$  is global learning rate, and  $\beta_1$ ,  $\beta_2$  control decay.

### 3.5 Libraries

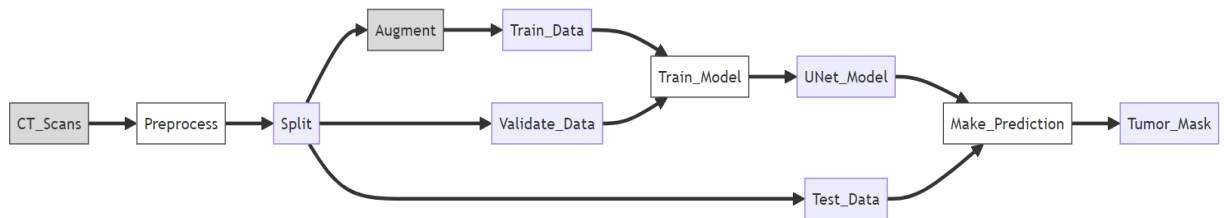
- **os:** The `os` module is used for interacting with the operating system. This is utilized to import names of files of images in dataset.
- **random:** The `random` module is used for generating random numbers and is often used for creating random data splits or augmentations.
- **tqdm:** This is a library for creating progress bars to monitor the progress of tasks. It's particularly useful when dealing with tasks that take some time to complete, such as loading or processing large datasets.
- **cv2 (OpenCV):** OpenCV is a popular computer vision library. It is used for image preprocessing and manipulation.
- **pandas:** The `pandas` library is used for data manipulation and analysis. It is used for working with structured data, such as CSV files, and organizing data for training and evaluation.
- **numpy:** NumPy is a fundamental package for scientific computing with Python. It is used for numerical operations and manipulation of arrays or tensors.
- **matplotlib:** Matplotlib is a popular library for creating static, animated, and interactive visualizations in Python. It's commonly used for data visualization, including plotting loss and accuracy curves during training.
- **tensorflow:** TensorFlow is an open-source deep learning framework developed by Google. It's one of the most widely used deep learning libraries for building and training neural networks.
  - **keras:** Keras is a high-level deep learning API that's integrated into TensorFlow. It simplifies the process of building, training, and evaluating neural networks.
  - **Input, Activation, BatchNormalization, Dropout, Lambda, Conv2D, Conv2DTranspose, MaxPooling2D, concatenate, AveragePooling2D, Dense, Flatten:** These are various layers and components available in TensorFlow/Keras for building neural network architectures.

- **Adam:** This is the optimization algorithm used during the training of neural networks.
- **EarlyStopping, ModelCheckpoint, LearningRateScheduler:** These are callback functions used to control the training process, save models, and adjust learning rates.
- **ImageDataGenerator:** This is used for real-time data augmentation during training.
- **backend (tensorflow.keras.backend):** The backend is used for low-level operations and configurations of Keras.
- **warnings:** The warnings library is used to filter or suppress warnings generated by other libraries. It is useful to suppress non-critical warnings.

### 3.7 Dataset

The dataset we considered is derived from the Liver Tumor Segmentation Benchmark (LiTS), which was organized in conjunction with the IEEE International Symposium on Biomedical Imaging (ISBI) 2017 and the International Conferences on Medical Image Computing and Computer-Assisted Intervention (MICCAI) 2017 and 2018. The image dataset is diverse and contains primary and secondary tumors with varied sizes and appearances with various lesion-to-background levels (hyper-/hypo-dense), created in collaboration with seven hospitals and research institutions. LiTS remains an active or research, e.g., contributing the liver-related segmentation tasks in Medical Segmentation Decathlon.

### 3.8 Data Flow Diagram



**Fig 5: Data Flow Diagram**

The process flow for developing and accessing a U-Net-based liver tumor segmentation model is shown in the diagram. The initial step involves preprocessing the raw CT scan images to standardize and normalize the data. Next, the preprocessed data is divided into test, validation, and training sets. To increase the quantity of samples, augmentation is applied to the training data. The model training loop uses the test data, supplemented training data, and unaltered validation data at different phases. Model weights are updated using the training data, and after each epoch, the validation data offers an objective assessment of the model's performance. After training is complete, the test data is used to assess the final U-Net model's generalization accuracy. The output of the trained model is a tumor segmentation mask, which is utilized to generate predictions on the test scans. Overall, the flow diagram demonstrates how the CT scan data is cleaned up, enhanced, input into the training of the model, tested, and utilized to draw conclusions. It outlines the essential elements required to put in place a comprehensive deep learning pipeline for the task of tumor segmentation.

# CHAPTER-4



## 4. IMPLEMENTATION

This section provides a detailed implementation of liver tumor segmentation, including the approaches applied at each stage of model training and tumor segmentation. We chose to create and train the U-Net model using the TensorFlow/Keras deep learning framework and Python as our programming language.

### 4.1 Data Loading and Preprocessing

In data loading, the `os` library is used to read the filenames for the segmented masks and the CT scan images.

```
images = "/lits-image-extract-dataset/images/train_images"
masks = "/lits-image-extract-dataset/masks/train_masks"

# List all file names in the image folder
image_paths = []
for root, dirs, files in os.walk(images):
    for file in files:
        file_path = os.path.join(root, file)
        image_paths.append(file_path)

# Function for extracting volume and slice numbers for sorting
def get_volume_and_slice_numbers(file_path):
    # Extract volume and slice numbers from the file path
    parts = file_path.split('_')
    volume_number = int(parts[1].split('-')[1])
    slice_number = int(parts[-1].split('.')[0])
    return (volume_number, slice_number)

# Sort the image paths based on both volume and slice numbers
image_paths.sort(key=get_volume_and_slice_numbers)

# Creating mask files list
mask_paths = [path.replace('images', 'masks').replace('.jpg',
'_mask.jpg') for path in image_paths]
threshold_percentage = 0
```

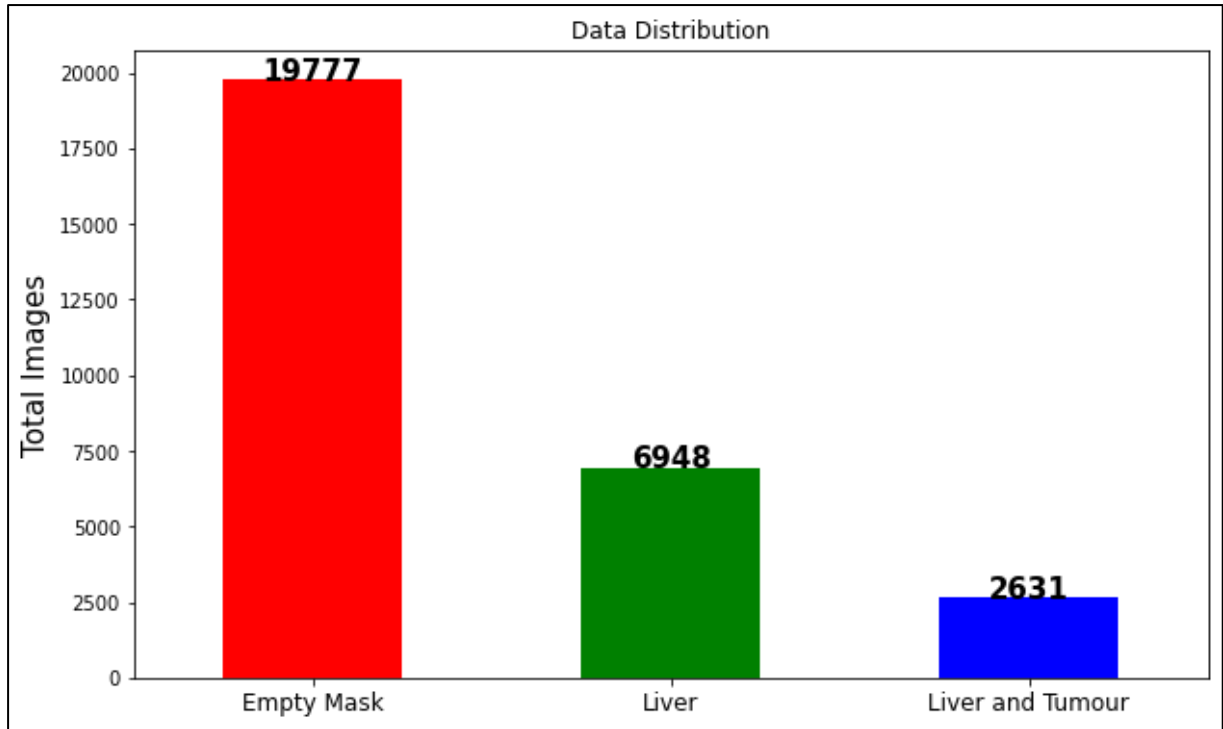
```

def check_tumor(mask_path):
    mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
    # Threshold the mask to identify tumor regions
    thresholded_mask = ((mask > 55) & (mask <
75)).astype(np.uint8) # Tumor regions will be 1, others 0
    # Calculate the percentage of tumor pixels in the binary mask
    tumor_pixel_percentage = (np.sum(thresholded_mask == 1) /
thresholded_mask.size) * 100
    # Check if the image contains tumor regions
    if (tumor_pixel_percentage > threshold_percentage):
        return '2'
    else:
        return '1'
def diagnosis(mask_path):
    # Check if the mask contains liver or tumor regions
    if (np.sum(imread(mask_path)) == 33554432):
        return '0'
    else:
        return (check_tumor(mask_path))
diagnosis_values = [] # To store the diagnosis values
for mask_path in tqdm(mask_paths, desc='Processing'):
    diagnosis_values.append(diagnosis(mask_path))

df = pd.DataFrame({"image_path": image_paths, "mask_path":
mask_paths, "diagnosis": diagnosis_values})

```

We have built functions that use cv2 to determine what is in the masks and categorize them into three groups. Masks are empty, with liver only, and with tumor and liver. Using the diagnosis and check\_tumor functions all masks are labelled '0' for empty, '1' for only liver and '2' for masks with liver and tumor. There are 19777 empty masks out of 29356 photos, 6948 masks with only liver, and 2631 masks with both liver and tumor.



**Fig 6: Data Classification**

```
df = df[(df['diagnosis'] != '0') & (df['diagnosis'] != '1')]
df.reset_index(drop = True, inplace=True)
```

```
df = df.sample(frac=1, random_state = 32)
```

```
df_train = df[:2000]
df_test = df[2000:2200]
df_val = df[2200:]
```

Using the sample and slicing functions in pandas, we have divided the dataset into train, test, and validation, with 2000, 200, and 431 pairs, respectively, taking only the liver and tumor into consideration for tumor segmentation. The actual preprocessing is taken care of by the image generator function, which normalizes the data and transforms it to prepare it for the training process.

## 4.2 Data Generator and Augmentation

We opted to use ImageDataGenerator from Keras for data generation and argumentation. With the use of this function, we are producing images in batches.

```

def train_generator(data_frame, batch_size, aug_dict,
                    image_color_mode="rgb",
                    mask_color_mode="grayscale",
                    image_save_prefix="image",
                    mask_save_prefix="mask",
                    save_to_dir=None,
                    target_size=(256,256),
                    seed=1):

    image_datagen = ImageDataGenerator(**aug_dict)
    mask_datagen = ImageDataGenerator(**aug_dict)

    image_generator = image_datagen.flow_from_dataframe(
        data_frame,
        x_col = "image_path",
        class_mode = None,
        color_mode = image_color_mode,
        target_size = target_size,
        batch_size = batch_size,
        save_to_dir = save_to_dir,
        save_prefix = image_save_prefix,
        seed = seed)
    mask_generator = mask_datagen.flow_from_dataframe(
        data_frame,
        x_col = "mask_path",
        class_mode = None,
        color_mode = mask_color_mode,
        target_size = target_size,
        batch_size = batch_size,
        save_to_dir = save_to_dir,
        save_prefix = mask_save_prefix,
        seed = seed)

    train_gen = zip(image_generator, mask_generator)

    for (img, mask) in train_gen:
        img, mask = adjust_data(img, mask)
        yield (img,mask)

```

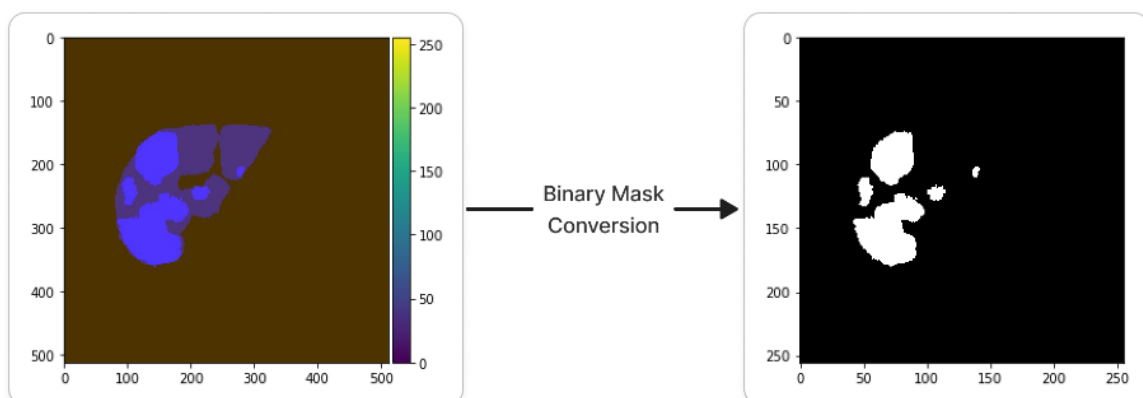
```

# adjust_data function makes the masks with liver and tumor into
only tumor,
# this converts the multi-class segmentation into binary
segmentation, it also normalizes the images
def adjust_data(img,mask):
    img = img / 255.
    mask = mask / 255.
    new_mask=np.zeros(mask.shape)
    new_mask[mask > 0.30] = 1
    new_mask[mask <= 0.30] = 0
    mask = new_mask

    return (img, mask)

```

For binary masks, we use the `adjust_data` function to change masks that contain both a tumor and liver to ones that just contain the tumor after normalizing the image and mask. This makes tumor segmentation, a binary segmentation task that is simpler than multi-class segmentation, the only task the model focuses on. Binary segmentation of tumors is sufficiently efficient since our objective is to segment tumors that will aid in the diagnosis stage. A wide range of data augmentation methods were used to increase the amount of training data. Rotations, shifts, flips, zooms, and elastic deformations were among them. During training, the Keras ImageDataGenerator class offered a practical framework for the augmentation of data in real time.



**Fig 7: Ground Truths to Binary Mask Conversion**

### 4.3 U-Net

The encoder and decoder components of the model architecture were described by the U-Net class. To allow context encoding and accurate localisation, it used skip connections, max pooling, upsampling, batch normalization and stacked convolutional blocks. Model hyperparameters like number of filters and depth were adjustable.

```
def unet(input_size=(256,256,3)):
    inputs = Input(input_size)

    conv1 = Conv2D(64, (3, 3), padding='same')(inputs)
    bn1 = Activation('relu')(conv1)
    conv1 = Conv2D(64, (3, 3), padding='same')(bn1)
    bn1 = BatchNormalization(axis=3)(conv1)
    bn1 = Activation('relu')(bn1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(bn1)

    conv2 = Conv2D(128, (3, 3), padding='same')(pool1)
    bn2 = Activation('relu')(conv2)
    conv2 = Conv2D(128, (3, 3), padding='same')(bn2)
    bn2 = BatchNormalization(axis=3)(conv2)
    bn2 = Activation('relu')(bn2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(bn2)

    conv3 = Conv2D(256, (3, 3), padding='same')(pool2)
    bn3 = Activation('relu')(conv3)
    conv3 = Conv2D(256, (3, 3), padding='same')(bn3)
    bn3 = BatchNormalization(axis=3)(conv3)
    bn3 = Activation('relu')(bn3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(bn3)

    conv4 = Conv2D(512, (3, 3), padding='same')(pool3)
    bn4 = Activation('relu')(conv4)
    conv4 = Conv2D(512, (3, 3), padding='same')(bn4)
    bn4 = BatchNormalization(axis=3)(conv4)
    bn4 = Activation('relu')(bn4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(bn4)
```

```

conv5 = Conv2D(1024, (3, 3), padding='same')(pool4)
bn5 = Activation('relu')(conv5)
conv5 = Conv2D(1024, (3, 3), padding='same')(bn5)
bn5 = BatchNormalization(axis=3)(conv5)
bn5 = Activation('relu')(bn5)

up6 = concatenate([Conv2DTranspose(512, (2, 2), strides=(2, 2),
padding='same')(bn5), conv4], axis=3)
conv6 = Conv2D(512, (3, 3), padding='same')(up6)
bn6 = Activation('relu')(conv6)
conv6 = Conv2D(512, (3, 3), padding='same')(bn6)
bn6 = BatchNormalization(axis=3)(conv6)
bn6 = Activation('relu')(bn6)

up7 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2),
padding='same')(bn6), conv3], axis=3)
conv7 = Conv2D(256, (3, 3), padding='same')(up7)
bn7 = Activation('relu')(conv7)
conv7 = Conv2D(256, (3, 3), padding='same')(bn7)
bn7 = BatchNormalization(axis=3)(conv7)
bn7 = Activation('relu')(bn7)

up8 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2),
padding='same')(bn7), conv2], axis=3)
conv8 = Conv2D(128, (3, 3), padding='same')(up8)
bn8 = Activation('relu')(conv8)
conv8 = Conv2D(128, (3, 3), padding='same')(bn8)
bn8 = BatchNormalization(axis=3)(conv8)
bn8 = Activation('relu')(bn8)

up9 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2),
padding='same')(bn8), conv1], axis=3)
conv9 = Conv2D(64, (3, 3), padding='same')(up9)
bn9 = Activation('relu')(conv9)
conv9 = Conv2D(64, (3, 3), padding='same')(bn9)
bn9 = BatchNormalization(axis=3)(conv9)
bn9 = Activation('relu')(bn9)

```

```
conv10 = Conv2D(1, (1, 1), activation='sigmoid')(bn9)
```

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

Using the Adam optimizer, the model was trained for 55 epochs with a batch size of 32. For every batch, combined binary cross-entropy and dice coefficient losses were computed. In the event that validation loss plateaued, learning rate scheduler lowered the learning rate. Every epoch, model weights were saved, and the best weights were retained. Model's performance is elaborated in results and discussion.

```
# Defining Loss function and metrics
```

```
smooth=1.
```

```
def dice_coef(y_true, y_pred):  
    y_true = K.flatten(y_true)  
    y_pred = K.flatten(y_pred)  
    intersection = K.sum(y_true * y_pred)  
    union = K.sum(y_true) + K.sum(y_pred)  
    return (2.0 * intersection + smooth) / (union + smooth)
```

```
def dice_coef_loss(y_true, y_pred):  
    return 1 - dice_coef(y_true, y_pred)
```

```
def bce_dice_loss(y_true, y_pred):  
    bce = tf.keras.losses.BinaryCrossentropy(from_logits=True)  
    return dice_coef_loss(y_true, y_pred) + bce(y_true, y_pred)
```

```
def iou(y_true, y_pred):  
    intersection = K.sum(y_true * y_pred)  
    sum_ = K.sum(y_true + y_pred)  
    jac = (intersection + smooth) / (sum_ - intersection + smooth)  
    return jac
```

```
# Set parameters
```

```
EPOCHS = 55
```

```
BATCH_SIZE = 32
```

```
learning_rate = 1e-4
```



```

train_generator_args = dict(rotation_range=0.1,
                             width_shift_range=0.05,
                             height_shift_range=0.05,
                             shear_range=0.05,
                             zoom_range=0.05,
                             horizontal_flip=True,
                             vertical_flip=True,
                             fill_mode='nearest')

train_gen = train_generator(df_train, BATCH_SIZE,
                           train_generator_args,
                           target_size=IMAGE_SIZE)

val_gen = train_generator(df_val, BATCH_SIZE,
                          dict(),
                          target_size=IMAGE_SIZE)

model = unet(input_size=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3))

# learning rate schedule
def lr_schedule(epoch):
    if epoch < 30:
        return 1e-4
    else:
        return 1e-5

opt = Adam(lr=learning_rate, beta_1=0.9, beta_2=0.999, epsilon=None,
amsgrad=False)
model.compile(optimizer=opt,
              loss=bce_dice_loss,
              metrics=['accuracy', iou, dice_coef])

callbacks = [ModelCheckpoint('unet_seg.hdf5', verbose=0,
                             save_best_only=True),
             ReduceLROnPlateau(monitor='val_loss',
                                factor=0.1,
                                patience=5, verbose=1, min_lr=1e-11),
             EarlyStopping(monitor='val_loss',

```

```
restore_best_weights=True, patience=5),  
LearningRateScheduler(lr_schedule)]
```

```
history = model.fit(train_gen,  
                    steps_per_epoch=len(df_train) / BATCH_SIZE,  
                    epochs=EPOCHS,  
                    callbacks=callbacks,  
                    validation_data = val_gen,  
                    validation_steps=len(df_val) / BATCH_SIZE)
```

## 4.4 Testing and Evaluation

The trained model was loaded and ran on the test set images. After the preprocessing of the test CT images, input was given to the model, and binary tumor predictions were produced by thresholding the probability masks produced by the model.

```
test_gen = train_generator(df_test, BATCH_SIZE,  
                          dict(),  
                          target_size=IMAGE_SIZE)  
results = model.evaluate(test_gen, steps=len(df_test) / BATCH_SIZE)  
print("Loss: ",results[0])  
print("Accuracy: ",results[1])  
print("Test IOU: ",results[2])  
print("Test Dice Coefficient: ",results[3])
```

The 'model.evaluate' function computed measures such as accuracy, IoU, dice score, and loss by comparing the predictions with ground truth masks. A few test images and predicted masks were visualized for deeper understanding and hyperparametric tuning.

## 4.5 Accuracy Calculation

Intersection over Union and Dice Coefficient are the true and reliable metrics for assessing an image segmentation model and procedure. However, we have utilized a function to determine the correct and erroneous anticipated masks in order to comprehend the accuracy of the model's performance. A prediction is deemed accurate if there is any overlap between the predicted mask and the ground truth, meaning that each one has at least one true value.

```

# adjust_mask adjusting masks to display with only tumor
def adjust_mask(mask_path):
    mask = cv2.imread(mask_path, 0)
    mask = cv2.resize(mask, IMAGE_SIZE, interpolation =
cv2.INTER_NEAREST)
    mask = mask / 255.
    new_mask=np.zeros(mask.shape)
    new_mask[mask > 0.30] = 1
    new_mask[mask <= 0.30] = 0
    return new_mask

cor , incor = 0 , 0
for i in range(len(df_test.index)):
    index=i
    img = cv2.imread(df_test['image_path'].iloc[index])
    img = cv2.resize(img ,IMAGE_SIZE)
    img = img / 255
    img = img[np.newaxis, :, :, :]
    pred = (np.squeeze(model.predict(img)) > 0.30)
    mask = np.squeeze(adjust_mask(df_test['mask_path'].iloc[index]))
    if pred.any() == mask.any():
        cor += 1
    else:
        incor += 1

print("Correct = " , cor , "Incorrect = " , incor )
print("Accuracy = " , (cor/(cor+incor)) * 100)

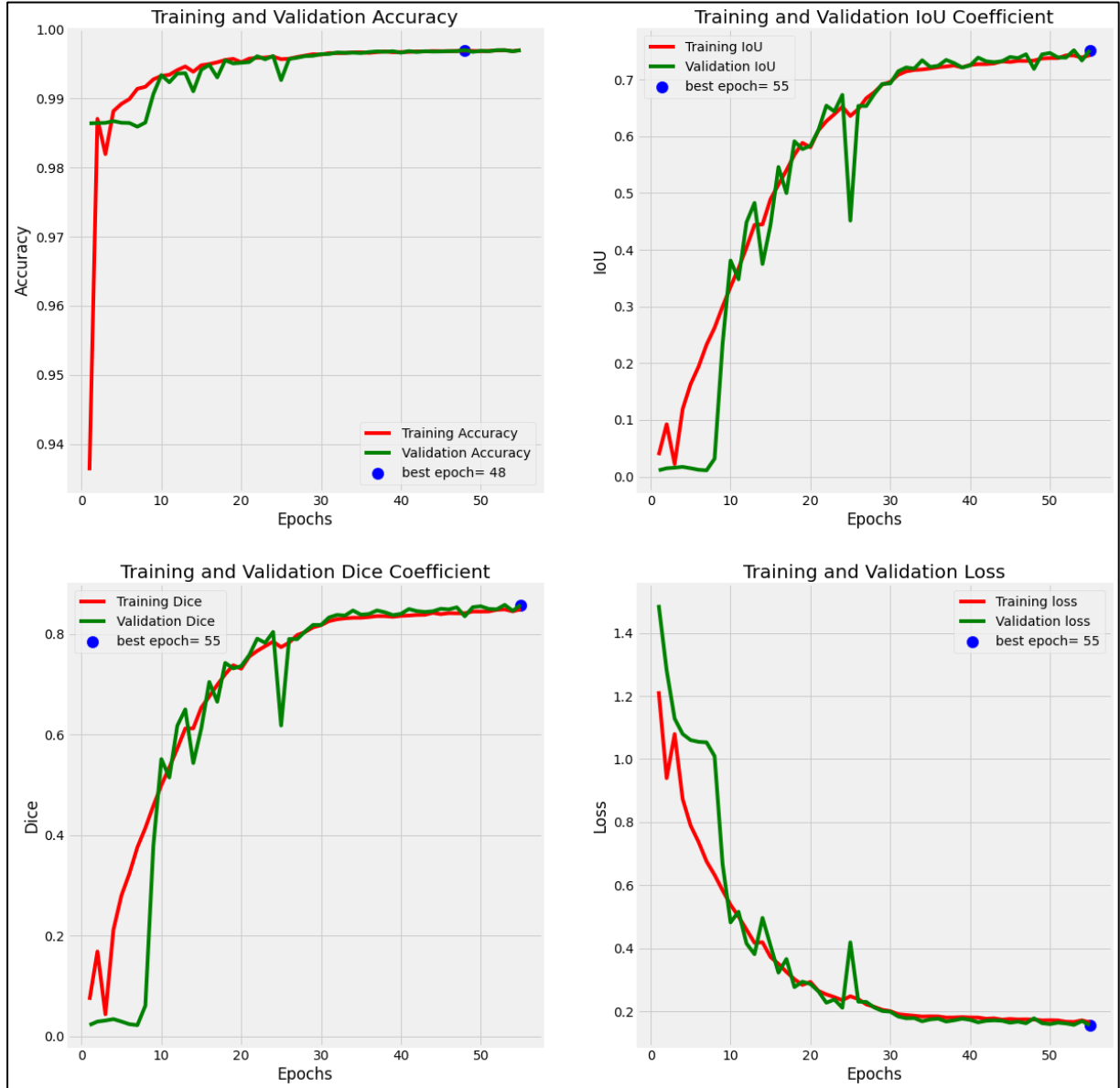
```

Accuracy was determined using a straightforward percentage method once the number of correct and incorrect predictions was determined. This enables us to determine the model's accuracy, or the image segmentation model's true accuracy.

# CHAPTER-5

## 5. RESULTS AND DISCUSSION

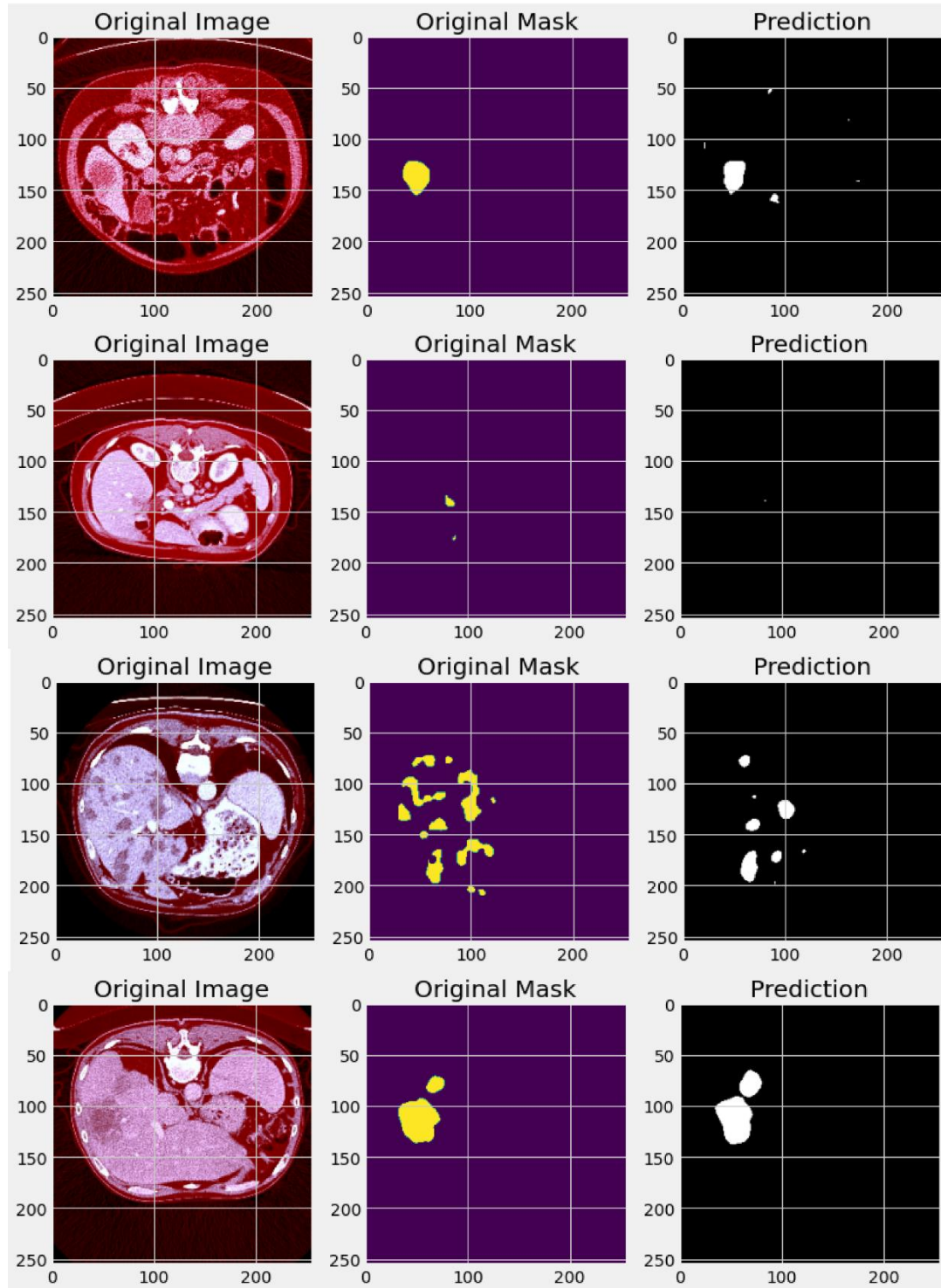
The liver tumor segmentation model was trained for 55 epochs on 2000 CT scan slices and validated on 431 slices after each epoch. The training loss decreased from 1.39 to 0.21 while the validation loss decreased from 1.49 to 0.16 over the course of training, indicating that the model was effectively learning to segment liver tumors.



**Fig 8: Training Metrics Plot**

With regard to process observation, the model's optimal iteration with the highest IoU and Dice coefficient scores is the 55th epoch. In the 55th epoch, the loss was at its lowest, even with the highest evaluation score. By identifying the maximum values from the model's training history, this optimal epoch was determined.

Following training, the model was evaluated on a test set of 200 slices. It achieved a Dice coefficient of 0.85 and IoU of 0.74 on the test set. The pixel-level accuracy was 0.865 and the number of correct predictions was 173 and incorrect predictions were 27 out of 200 test scans. A few of the test images were visualized with their ground truths and predicted masks; below is the segment visualization part.



**Fig 9: Visualisation of Predictions**

Qualitatively, the model was often able to capture the full shape and extent of larger, well-defined tumors. For tumors with diffuse boundaries, the predictions tended to undershoot slightly compared to the ground truth. A portion of false-positive findings occurred due to similar-appearing structures adjacent to the liver being mistaken for tumors. The model also struggled to separate fused tumor masses correctly in some cases.

Overall, the quantitative metrics and qualitative observations indicate successful training of a U-Net model for liver tumor segmentation from CT scans. The model achieves good performance on clear, large tumors within the liver but has limitations in handling smaller, diffuse tumors or those extending beyond the liver borders. Further refinement of the training data and methodology could potentially improve results in these challenging cases.

In conclusion, the U-Net model performs well when segmenting liver tumors from CT scans attributable to its dice-based training methodology. There are chances to further enhance the management of difficult tumor form and features.

# **CHAPTER-6**



## 6. CONCLUSION AND FUTURE ENHANCEMENT

**In summary**, this study used a deep convolutional neural network technique to automate the segmentation of liver tumors. The LiTS challenge dataset's CT scans and ground truth masks were used to build and train a U-Net architecture. To increase the training data and strengthen the robustness of the model, extensive data augmentation was used. A composite loss function that included dice coefficient terms and binary cross-entropy was used to train the model from start to finish. This helped the model maximize region overlap similarity as well as accuracy at the pixel level. A test set of CT scan images that was kept out was used to assess the trained model. With a Dice score of 0.85 and an IoU of 0.74, it demonstrated a high degree of agreement between the true and projected tumor locations. Visual examination revealed precise tumor identification and segmentation with distinct boundaries. The model had some difficulties with small or diffuse tumors, occasionally under-segmenting the tumor extent in those cases. But overall, the quantitative and qualitative results demonstrate the feasibility of using deep learning for automated liver tumor segmentation from CT scans. Further refinement of the methodology could improve generalization and robustness.

**Additionally**, there are several strategies to improve the model and put it into clinical use. By incorporating multi-modality datasets like MRI, the model may become more resilient to changes in imaging procedures. Compared to 2D, 3D convolutions could more effectively take volumetric context into account. Further post-processing techniques, such as conditional random fields, could improve the segmentations even further. The capacity to generalize would continue to increase with training on larger and more varied samples. In order to include the model in workflows for clinical diagnosis and treatment planning, user-friendly interfaces and APIs for hospital system integration would need to be developed. Additionally, procedures for regulatory approval would need to be followed. Conducting comprehensive studies on safety, effectiveness, and resilience would be necessary to transition the technique from a prototype to a standard clinical procedure. However, the outcomes thus far offer a good starting point for the future creation of trustworthy automated methods for liver tumor segmentation.

## REFERENCES

- [1] Amandeep Kaur, Ajay Pal Singh Chahuan, Ashwani Kumar Aggarwal. (2021). An automated slice sorting technique for multi-slice computed tomography liver cancer images using convolutional network.
- [2] Xiao Han. (2017). Automatic Liver Lesion Segmentation Using a Deep Convolutional Neural Network Method.
- [3] Grzegorz Chlebus, Hans Meine, Jan Hendrik Moltz, Andrea Schenk. (2017). Neural Network-Based Automatic Liver Tumor Segmentation with Random Forest-Based Candidate Filtering.
- [4] Yading Yuan. (2017). Hierarchical Convolutional-Deconvolutional Neural Networks for Automatic Liver and Tumor Segmentation.
- [5] Zhiqi Bai, Huiyan Jiang, Siqi Li1, and Yu-Dong Yao. (2019). Liver Tumor Segmentation Based on Multi-Scale Candidate Generation and Fractal Residual Network.
- [6] Patrick Bilic, Patrick Christ, Hongwei Bran Li, Eugene Vorontsov, Avi Ben-Cohen etc. (2022). The Liver Tumor Segmentation Benchmark (LiTS).
- [7] Shervin Minaee, Shervin Minaee, Antonio Plaza, Nasser Kehtarnavaz, Demetri Terzopoulos. (2021). Image Segmentation Using Deep Learning: A Survey.
- [8] Mubashir Ahmad, Syed Furqan Qadri, Salman Qadri, Iftikhar Ahmed Saeed, Syeda Shamaaila Zareen, Zafar Iqbal, Amerah Alabrah, Hayat Mansoor Alaghbari, Sk. Md. Mizanur Rahman. (2022). A Lightweight Convolutional Neural Network Model for Liver Segmentation in Medical Diagnosis.
- [9] Wen Li1, Fucang Jia1, Qingmao Hu. (2015). Automatic Segmentation of Liver Tumor in CT Images with Deep Convolutional Neural Networks.
- [10] Wen Li1, Fucang Jia1, Qingmao Hu. (2015). Automatic Segmentation of Liver Tumor in CT Images with Deep Convolutional Neural Networks.
- [11] Patrick Ferdinand Christa, Florian Ettlinger, Felix Gr`una, Mohamed Ezzeldin. (2017). Automatic Liver and Tumor Segmentation of CT and MRI Volumes Using Cascaded Fully Convolutional Neural Networks.

- [12] Xin Dong, Yizhao Zhou, Lantian Wang, Jingfeng Peng, Yanbo Lou, Yiqun Fan. (2020). Liver Cancer Detection Using Hybridized Fully Convolutional Neural Network Based on Deep Learning Framework.
- [13] Feiniu Yuan, Zhengxiao Zhang, Zhijun Fang. (2022). An effective CNN and Transformer complementary network for medical image segmentation.
- [14] Sultan Almotairi, Ghada Kareem, Mohamed Aouf, Badr Almutairi, Mohammed A. M. Salem. (2019). Liver Tumor Segmentation in CT Scans Using Modified SegNet.
- [15] Agarwal, T. K., Tiwari, M., & Lamba, S. S. (2014). Modified histogram based contrast enhancement using homomorphic filtering for medical images. In 2014 IEEE international advance computing conference.
- [16] Anwar, S. M., Majid, M., Qayyum, A., Awais, M., Alnowami, M., & Khan, M. K. (2018). Medical image analysis using convolutional neural networks: A review. *Journal of Medical Systems*, 42, 226.
- [17] Ben-Cohen, A., Diamant, I., Klang, E., Amitai, M., & Greenspan, H. (2016). Fully convolutional network for liver segmentation and lesions detection. In *Deep learning and data labeling for medical applications* (pp. 77–85). Springer
- [18] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. *Cluster Computing*, 1–13.
- [19] [https://github.com/bnsreenu/python\\_for\\_microscopists](https://github.com/bnsreenu/python_for_microscopists)
- [20] [https://github.com/qubvel/segmentation\\_models](https://github.com/qubvel/segmentation_models)