

# Automatic Segmentation of liver Tumor using Deep Learning

Mrinalini kakroo<sup>1</sup>, Vibhakar mansotra<sup>2</sup>

<sup>1</sup>Student, Department of Computer Science & IT, University of Jammu, Jammu, India  
kakroomrinalini@gmail.com

<sup>2</sup>Professor, Department of Computer Science & IT, University of Jammu, Jammu. India

---

## ABSTRACT

When it comes to medical imaging data like CT or MRI images, automatic segmentation of liver tumors is the process of precisely locating and isolating tumor spots without the need for human involvement. Liver tumor segmentation is crucial for accurate diagnosis and therapeutic planning of liver cancer. The purpose of this work is to provide a comprehensive summary of the state-of-the-art approaches to automatically segmenting liver cancers from medical imaging data. Here, we'll go through some of the more general and specialized approaches now in use in this field. By allowing for more precise tumor delineation, this technology may help doctors improve patient outcomes via prompt, individualized treatment. With increased research and collaboration between the medical and AI fields, deep learning-based liver tumor segmentation has the potential to become a vital weapon in the fight against liver cancer.

---

**Keywords:** Deep Learning, Segmentation, Tumors.

---

## I. INTRODUCTION

The right side of the body, just under the rib cage, is home to the massive, pyramid-shaped liver. The structure rests just underneath the right lung. It has a bilateral sagittal lobe division. The liver helps in nutrition storage and digestion. Carbohydrates, lipids, proteins, and starches all contribute to their make-up. Albumin is only one kind of protein it can make. This aids the maintenance of the body's fluid equilibrium. When someone is bleeding, the liver produces clotting factors to help the blood thicken and clot. The liver secretes bile, a chemical necessary for digestion and other bodily processes. The liver plays a crucial role in the body by filtering out harmful substances. Chemicals may accumulate and cause harm when the liver isn't functioning properly.

A computed tomography (CT) or magnetic resonance imaging (MRI) scan may detect liver cancer, unlike the vast majority of other malignancies. A CT scan of the abdomen produces high-resolution cross-sectional images. Additional scan processing is required for liver segmentation and to isolate tumorous areas from the rest of the CT picture. Tumors in the abdomen CT picture have an intensity that is similar to that of normal tissues, making segmentation difficult. Because of this, the pictures need amplification and processing to identify cancerous tissue.

In order to detect cancer early and treat it more effectively, recent advances in medical imaging have greatly enhanced diagnostic and radiation therapy delivery technologies. Many different imaging methods have been used, each with its own set of advantages and disadvantages. Among them are positron emission tomography (PET), ultrasonography (US), computed tomography (CT), and magnetic resonance imaging (MRI). Characterizing HCC in the liver often requires CT, MRI, or US. However, CT scans tend to be the go-to imaging method throughout the planning stages of a therapy. This research makes use of CT scan simulations, which are increasingly used in the planning of radiation therapy, to segment the liver and liver tumors.

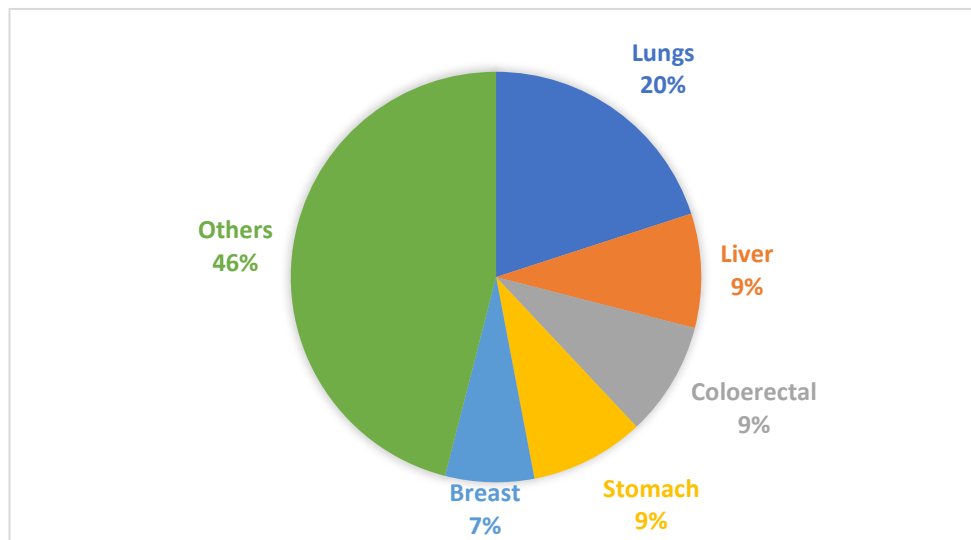


Figure 1: The worldwide percentage distribution of cancer types (2018 accessed data)

Automatic segmentation refers to the process of using software to locate tumors in medical images. It requires programming algorithms for computers to analyze photographic images and determine their meaning. Interest in deep learning, a subfield of AI, has skyrocketed in recent years because to its impressive performance in various image analysis tasks, such as medical image segmentation.

### 1.1 HYPOTHESIS

"Deep learning-based segmentation models will achieve greater accuracy and efficiency than standard image processing approaches," the authors write of using deep learning to isolate liver tumors from other medical imaging data.

### 1.2 EXPECTED CONTRIBUTION TO THE STUDY

The proposed study will help to segment the tumor using Deep learning with good accuracy.

### 1.3 JUSTIFICATION

- Clinical Significance
  - Diagnosis: - Accurate segmentation assists in the early identification and characterization of liver tumors, allowing for timely diagnosis and intervention. Correct tumour segmentation facilitates the measurement of tumour size, location, and multiplicity, all of which are crucial for staging and prognostic evaluation.
  - Treatment Planning: - Accurate segmentation is necessary for planning surgeries like resection, radiation therapy, and ablation. Target volumes for radiation therapy may be determined using precision segmentation, enabling careful dosing to achieve the desired therapeutic effect while sparing normal liver tissue.
- Manual Segmentation Limitations
 

Automated segmentation methods are able to overcome the limitations of manual segmentation when it comes to liver tumors. There are a number of major limitations to manual segmentation.

  - Time-Consuming: - When dealing with large datasets or complex tumor shapes, manual hepatic tumor segmentation may be a time-consuming and laborious process. Manually defining tumour boundaries for each image or slice may be time-consuming and stressful, particularly under pressure.
  - High Workload and Human Error: - The labor involved in manual segmentation increases the potential of human error, especially in busy hospital settings. Mistakes in delineation or misconceptions about patients' conditions may have serious consequences for their treatment.

Overcoming these limitations and using automated segmentation approaches, such as deep learning-based systems, may improve the accuracy, speed, and consistency of liver tumor segmentation. Automatic segmentation

techniques aim to overcome these challenges by providing more objective, repeatable, and successful solutions, which will ultimately benefit patients with liver cancer in terms of diagnosis, treatment planning, and monitoring.

#### 1.4 OBJECTIVES

- In order to measure the efficacy of the model, we will be looking at its F1-score, recall, precision, and accuracy.
- In order to learn, experiment with, and verify the segmentation method.

## II. REVIEW OF LITERATURE

When applied to medical images of the liver, deep learning-based automatic segmentation of tumors refers to the process of using cutting-edge computational methods to automatically detect and segment tumor regions. In this section, we provide a high-level overview of the literature on the topic of the work.

### 2.1 LITERATURE SURVEY RELATED TO CURRENT RESEARCH

The authors of this paper (**ChangYang Li et al.**) [1] provided a well-structured literature evaluation of probabilistic atlas-based fully automated liver segmentation for low and high-contrast CT volumes. The author of this work acknowledged that automated liver segmentation is challenging owing to the liver's size and shape variability and its similarity in density to other organs. They proposed a method that 1) employs iteratively created probabilistic atlases of the liver and rib cage, 2) employs the Gaussian distribution analysis to prevent the incorrect identification of the unnecessary surrounding tissues as "liver area" in the conventional probabilistic atlas-based method, and 3) returns the "missing sections" of the liver via deformable registration. Our automated technique can extract liver tissue from both high- and low-contrast CT volumes. Forty clinical CT images went towards making and verifying the atlas. Our method outperformed two other approaches that also relied on a probabilistic atlas to segment the liver.

**Weimin Huang et al.** [2] shown how to use 3D CT scans to identify liver tumors and isolate them from normal tissue. The process of autonomously detecting tumors may be seen as a two-class classification issue. In order for the method to be useful for tumor segmentation, each voxel must be properly labeled as belonging to a tumor class or a nontumor class. Each voxel has its own unique set of characteristics that are represented by a robust feature vector. An efficient learning algorithm In order to train a voxel classifier, the Extreme Learning Machine (ELM) is used. They assert that ELM can be trained as a one-class classifier for automated liver tumor diagnosis using only healthy liver samples as training data, and they provide proof of this. This leads to the development of a revolutionary approach to cancer diagnosis. They tested it using a binary ELM. To partially automate tumour boundary detection, we choose training data in 3D space inside a restricted area of interest (ROI) for classifiers. Our method is put to the test on a dataset of CT scans from real patients, and the results show promising identification and segmentation results. A major advantage of one-class ELM is its use as a preliminary detection approach, particularly if a two-class classifier is not well-trained or cannot adequately represent a novel or previously unrecognized tumor.

**Belgherbi, A et al.** [3] shown the two stages required for segmentation. Remove the liver as early as possible with the help of morphological restoration. Second-stage liver lesions may be identified using the watershed change. Since CT contrast between lesions and liver intensity was low. Image-based liver lesion segmentation is notoriously difficult. Therefore, anatomical segmentation for hepatic tumors employs mathematical morphology techniques in an effort to address this problem. For identifying and segmenting focal liver lesions, the provided method can correctly segment lesions from the patient database. This method allows for the pre-testing of CT scan images. They put their proposed method to the test on a variety of images, and were pleased with the results. The detection rate for segments was 92%, with specificity of 99%. There's potential for improved segmentation in future by taking more nuanced approach to deal with wide variety of lesions & liver's boundaries.

**R. Rajagopal, P. Subbiah** [4], suggested an innovative and accurate method for segmenting liver tumors using CT scans. Noise reduction and contrast enhancement are the initial steps in pre-processing a CT scan of the liver. A support vector machine (SVM) classifier, trained on the user-provided image sets, is then used to label the tumor region in the liver image. Sequentially applying morphological procedures and feature extractions to the segmented binary image helps improve the SVM classification's initial segmentation result. The results of the experiment show that the proposed algorithm outperforms conventional methods. In order to help in subsequent diagnoses, the study proposes a novel method for categorizing tumors as part of the tumour segmentation strategy. The main advantage of their method is its ability to quickly and accurately diagnose different forms of liver tumors

with little human interaction. Future work may include incorporating neural networks and fuzzy algorithms into the suggested technique to further improve it.

### III. METHODOLOGY AND EXPERIMENTAL SETUP

#### 3.1 RESEARCH METHODOLOGY

Acquiring patient data was the first stage of the process. The kind of sickness and imaging technique utilized informed the selection of relevant patient data. The technique then proceeded to standardize the cleaning of the data, choose a convolutional neural network architecture appropriate for the task at hand, and evaluate the model's efficacy.

#### 3.2 PATIENT DATA

For this study, we utilized data from the subsets of the 3Dircadb dataset known as 3Dircadb1, 3Dircadb2, and so on, all of which were taken from the Liver segmentation 3D-IRCADb (3D Image Reconstruction for Comparison of Algorithm library) database. Each cluster represents an individual patient profile in the larger 3Dircadb dataset. Because of this, we may consider 3Dircadb1 to be a special instance or subset of the larger 3Dircadb dataset. Three-dimensional computed tomography images from 10 male and 10 female patients, 75% of whom had liver tumors, make up the 3D-IRCADb-01 database. Each patient is represented by their own folder, which may be downloaded independently or all at once. Information regarding the image, such as its width, depth, and height as well as where the tumors were segmented by Couninaud, is provided in the table below. It also draws attention to the serious difficulties that liver segmentation algorithms may encounter due to interaction with neighboring organs, an irregular liver form or density, or even image artifacts.





















Aperçu	N°	Sex	Year of birth	Voxel size (mm)	Image size (pixels)	Liver size (cm)	Liver Average density	Liver Pathologies	Segmentation drawbacks
	1	F	1944	0,57 0,57 1,6	512 512 129	18,3 15,1 14,1	111	7 tumours in III, IV/V, VII, VIII	stomach, pancreas, duodenum
	2	F	1987	0,78 0,78 1,6	512 512 172	20,1 16,9 15,7	84	1 tumour in V	pancreas, duodenum
	3	M	1956	0,62 0,62 1,25	512 512 200	16,7 14,9 15,2	108	1 tumour in IV	Artefac due to metal
	4	M	1942	0,74 0,74 2	512 512 91	16,9 12,0 17,2	107	7 tumours	heart
	5	M	1957	0,78 0,78 1,6	512 512 139	19,8 18,6 19,1	69	0 tumour in Liver	diaphragm, duodenum
	6	M	1929	0,78 0,78 1,6	512 512 135	18,8 14,3 20,2	105	20 tumours in liver	Heart
	7	M	1946	0,78 0,78 1,6	512 512 151	24,9 15,2 16,6	115	0 tumour in Liver (1 adrenal)	spleen
	8	F	1970	0,56 0,56 1,6	512 512 124	23,5 17,1 12,5	159	3 tumours in I, II and IV	
	9	M	1949	0,87 0,87 2	512 512 111	20,6 17,0 18,1	90	1 tumour in V/VIII	stomach, colon
	10	F	1953	0,73 0,73 1,6	512 512 122	18,4 15,5 14,8	104	8 tumours	duodenum, pancreas
	11	M	1956	0,72 0,72 1,6	512 512 132	19,1 14,4 16,2	108	0 tumour in Liver	stomach, pancreas
	12	F	1973	0,68 0,68 1,0	512 512 260	19,3 17,7 11,0	118	1 tumour in VI	
	13	M	1951	0,67 0,67 1,6	512 512 122	20,0 12,9 18,1	111	20 tumours	duodenum, heart
	14	F	1970	0,72 0,72 1,6	512 512 113	22,4 15,4 13,4	109	0 tumour in Liver	spleen, pancreas
	15	F	1946	0,78 0,78 1,6	512 512 125	18,8 17,7 14,7	104	2 tumours in II and VIII	pancreas
	16	M	1950	0,70 0,70 1,6	512 512 155	20,2 17,7 20,2	40	1 tumour in V	muscle
	17	M	1942	0,74 0,74 1,6	512 512 119	19,8 17,4 18,9	99	2 tumours in II and VIII	stomach, heart
	18	F	1958	0,74 0,74 2,5	512 512 74	22,5 15,1 18,6	49	1 tumour in IV/V	oesophagus, muscles
	19	F	1970	0,70 0,70 4	512 512 124	19,5 16,5 14,2	135	46 tumours	stomach, duodenum, pancreas
	20	F	1949	0,81 0,81 2	512 512 225	20,0 16,6 16,8	59	0 tumour in Liver	stomach

Fig 6: Shows data characteristics used in this research

### 3.4 IMPLEMENTATION

To get our CNN model up and running, we must first import the necessary libraries. We're making use of these libraries:

**Keras:** Python's Keras provides a high-level interface for neural networks. TensorFlow, Theano, and CNTK are just few of the major deep learning frameworks that it is based on. Keras provides a simple and straightforward interface for building and training deep learning models like CNNs. It facilitates rapid model prototyping and experimentation by providing a modular and adaptable framework for constructing diverse neural networks. Keras is compatible with a wide variety of other Python libraries and may do computations using either the central processing unit (CPU) or the graphics processing unit (GPU).

**Scikit-learn:** Scikit-learn, sometimes known as sklearn, is a Python package for machine learning. It offers a wide variety of techniques and tools for applications including classification, regression, clustering, and dimensionality reduction. Even if deep learning isn't scikit-learn's main emphasis, it does provide a good basis for more conventional machine learning tasks by way of its data preparation, model selection, and assessment capabilities. Common applications include dealing with non-image datasets, doing data preprocessing, and extracting features from raw data.

**NumPy:** NumPy is a powerful open-source Python toolkit for numerical computation. It is short for "Numerical Python" and allows users to deal with massive multi-dimensional arrays and matrices, as well as a wide variety of high-level mathematical operations. NumPy is an essential module for scientific computing and the foundation for many other libraries in the Python environment for scientific and data analysis.

**OpenCV (cv2):** When it comes to computer vision tasks, many developers turn to OpenCV (Open-Source Computer Vision Library). It has a wide variety of tools and algorithms for working with images and videos, such as editing, detecting features, identifying objects, and calibrating cameras. In computer vision applications, OpenCV is especially helpful for data augmentation, visualization, and picture preparation. Many users of Python may import OpenCV's cv2 module to have access to the library's features.

**OS:** Python's OS module allows for communication with the OS. You can manipulate files, folders, and processes in a wide variety of ways. File and directory management, reading and writing environment variables, running system commands, and manipulating path structures are all examples of frequent OS module use. Loading datasets from disk, maintaining file directories, and planning the training process are all examples of where the OS module might be helpful in the context of deep learning and CNNs. Keras, scikit-learn, OpenCV, and the OS module are libraries that provide crucial features for many facets of deep learning and computer vision projects. Gaining familiarity with their features and putting them to good use may greatly improve your efficiency while developing and deploying CNN models.

#### Importing all the libraries

```
# Import the ImageDataGenerator class from the tensorflow.keras.preprocessing.image module
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Importing the numpy library and assigning an alias np
import numpy as np
# Importing the tensorflow library and assigning an alias tf
import tensorflow as tf
# Importing the pandas library and renaming it as 'pd'
import pandas as pd
# Import the tqdm library for progress bars
from tqdm import tqdm
# Import the os module
import os
# Importing the 'imread' and 'createCLAHE' functions from the 'cv2' module
from cv2 import imread, createCLAHE
# Importing the OpenCV library
import cv2
# Importing the glob module to retrieve file paths matching a pattern
from glob import glob
# This line enables the display of matplotlib plots inline in Jupyter notebooks
%matplotlib inline
# This line imports the pyplot module from the matplotlib library
import matplotlib.pyplot as plt
# Importing the clear_output function from the IPython.display module
from IPython.display import clear_output
# Importing the Adam optimizer from the Keras optimizers module
from tensorflow.keras.optimizers import Adam
# Import the train_test_split function from the scikit-learn library
from sklearn.model_selection import train_test_split
```

#### Loading the data

```

train_imgs_path = "../content/3dircadb/3DIRCADB/train/Images/"
train_masks_path = "../content/3dircadb/3DIRCADB/train/Masks"
test_imgs_path = "../content/3dircadb/3DIRCADB/test/Images"
test_masks_path = "../content/3dircadb/3DIRCADB/test/Masks"

```

- LIVER SEGMENTATION

```

model = ResUNet()
adam = keras.optimizers.Adam()
model.compile(optimizer=adam, loss=dice_coef_loss, metrics=["acc", dice_coef])

```

- TUMOR SEGMENTATION

```

model = ResUNet()
adam = keras.optimizers.Adam()
model.compile(optimizer=adam, loss=dice_coef_loss, metrics=["acc", dice_coef])

```

### MODEL FITTING (Liver Segmentation)

```

Epoch 1/20
2027/2027 [=====]2027/2027 [=====] - 996s 491ms/step - loss: 0.1306 - acc: 0.9736 - dice_coef: 0.8694 - val_loss: 0.1069 - val_acc: 0.9758 - val_dice_coef: 0.8931

Epoch 2/20
2027/2027 [=====]2027/2027 [=====] - 736s 363ms/step - loss: 0.0991 - acc: 0.9791 - dice_coef: 0.9009 - val_loss: 0.1570 - val_acc: 0.9700 - val_dice_coef: 0.8430

Epoch 3/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0900 - acc: 0.9791 - dice_coef: 0.9020 - val_loss: 0.0977 - val_acc: 0.9784 - val_dice_coef: 0.9023

Epoch 4/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0768 - acc: 0.9822 - dice_coef: 0.9232 - val_loss: 0.0947 - val_acc: 0.9783 - val_dice_coef: 0.9053

Epoch 5/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0733 - acc: 0.9826 - dice_coef: 0.9267 - val_loss: 0.1571 - val_acc: 0.9675 - val_dice_coef: 0.8429

Epoch 6/20
2027/2027 [=====]2027/2027 [=====] - 735s 362ms/step - loss: 0.0662 - acc: 0.9833 - dice_coef: 0.9338 - val_loss: 0.9792 - val_acc: 0.9092 - val_dice_coef: 0.8288

Epoch 7/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0626 - acc: 0.9835 - dice_coef: 0.9374 - val_loss: 0.1453 - val_acc: 0.9722 - val_dice_coef: 0.8547

Epoch 8/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0538 - acc: 0.9845 - dice_coef: 0.9462 - val_loss: 0.0602 - val_acc: 0.9834 - val_dice_coef: 0.9398

Epoch 9/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0499 - acc: 0.9849 - dice_coef: 0.9501 - val_loss: 0.0481 - val_acc: 0.9845 - val_dice_coef: 0.9519

Epoch 10/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0428 - acc: 0.9859 - dice_coef: 0.9572 - val_loss: 0.0832 - val_acc: 0.9790 - val_dice_coef: 0.9168

```

```

Epoch 11/20
2027/2027 [=====]2027/2027 [=====] - 733s 362ms/step - loss: 0.0394 - acc: 0.9862 - dice_coef: 0.9606 - val_loss: 0.1244 - val_acc: 0.9721 - val_dice_coef: 0.8756

Epoch 12/20
2027/2027 [=====]2027/2027 [=====] - 732s 361ms/step - loss: 0.0359 - acc: 0.9866 - dice_coef: 0.9641 - val_loss: 0.0668 - val_acc: 0.9818 - val_dice_coef: 0.9332

Epoch 13/20
2027/2027 [=====]2027/2027 [=====] - 732s 361ms/step - loss: 0.0305 - acc: 0.9871 - dice_coef: 0.9695 - val_loss: 0.0412 - val_acc: 0.9853 - val_dice_coef: 0.9588

Epoch 14/20
2027/2027 [=====]2027/2027 [=====] - 732s 361ms/step - loss: 0.0313 - acc: 0.9871 - dice_coef: 0.9687 - val_loss: 0.0451 - val_acc: 0.9848 - val_dice_coef: 0.9549

Epoch 15/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0292 - acc: 0.9873 - dice_coef: 0.9700 - val_loss: 0.2057 - val_acc: 0.9647 - val_dice_coef: 0.7943

Epoch 16/20
2027/2027 [=====]2027/2027 [=====] - 734s 362ms/step - loss: 0.0277 - acc: 0.9875 - dice_coef: 0.9723 - val_loss: 0.0263 - val_acc: 0.9873 - val_dice_coef: 0.9737

Epoch 17/20
2027/2027 [=====]2027/2027 [=====] - 732s 361ms/step - loss: 0.0282 - acc: 0.9874 - dice_coef: 0.9718 - val_loss: 0.0264 - val_acc: 0.9873 - val_dice_coef: 0.9736

Epoch 18/20
2027/2027 [=====]2027/2027 [=====] - 732s 361ms/step - loss: 0.0246 - acc: 0.9879 - dice_coef: 0.9754 - val_loss: 0.0248 - val_acc: 0.9875 - val_dice_coef: 0.9752

Epoch 19/20
2027/2027 [=====]2027/2027 [=====] - 732s 361ms/step - loss: 0.0240 - acc: 0.9879 - dice_coef: 0.9760 - val_loss: 0.0268 - val_acc: 0.9873 - val_dice_coef: 0.9732

Epoch 20/20
2027/2027 [=====]2027/2027 [=====] - 733s 362ms/step - loss: 0.0242 - acc: 0.9879 - dice_coef: 0.9758 - val_loss: 0.0224 - val_acc: 0.9878 - val_dice_coef: 0.9776

```

## MODEL FITTING (Tumor Segmentation)

Epoch 1/50 2027/2027	[=====]	2027/2027	[=====]	- 049s	419ms/step	- loss: 0.3623	- acc: 0.9933	- dice_coef: 0.6377	- val_loss: 0.7680	- val_acc: 0.9922	- val_dice_coef: 0.2120
Epoch 2/50 2027/2027	[=====]	2027/2027	[=====]	- 741s	365ms/step	- loss: 0.2845	- acc: 0.9965	- dice_coef: 0.7155	- val_loss: 0.2720	- val_acc: 0.9959	- val_dice_coef: 0.7280
Epoch 3/50 2027/2027	[=====]	2027/2027	[=====]	- 739s	365ms/step	- loss: 0.2535	- acc: 0.9969	- dice_coef: 0.7465	- val_loss: 0.4048	- val_acc: 0.9956	- val_dice_coef: 0.5952
Epoch 4/50 2027/2027	[=====]	2027/2027	[=====]	- 739s	365ms/step	- loss: 0.2428	- acc: 0.9971	- dice_coef: 0.7572	- val_loss: 0.2370	- val_acc: 0.9968	- val_dice_coef: 0.7630
Epoch 5/50 2027/2027	[=====]	2027/2027	[=====]	- 733s	361ms/step	- loss: 0.2381	- acc: 0.9972	- dice_coef: 0.7619	- val_loss: 0.2115	- val_acc: 0.9971	- val_dice_coef: 0.7885
Epoch 6/50 2027/2027	[=====]	2027/2027	[=====]	- 730s	360ms/step	- loss: 0.2251	- acc: 0.9973	- dice_coef: 0.7749	- val_loss: 0.4259	- val_acc: 0.9909	- val_dice_coef: 0.5741
Epoch 7/50 2027/2027	[=====]	2027/2027	[=====]	- 730s	360ms/step	- loss: 0.2210	- acc: 0.9973	- dice_coef: 0.7790	- val_loss: 0.2892	- val_acc: 0.9956	- val_dice_coef: 0.7188
Epoch 8/50 2027/2027	[=====]	2027/2027	[=====]	- 732s	361ms/step	- loss: 0.2114	- acc: 0.9974	- dice_coef: 0.7886	- val_loss: 0.1772	- val_acc: 0.9976	- val_dice_coef: 0.8228
Epoch 9/50 2027/2027	[=====]	2027/2027	[=====]	- 730s	360ms/step	- loss: 0.2076	- acc: 0.9975	- dice_coef: 0.7924	- val_loss: 0.2570	- val_acc: 0.9969	- val_dice_coef: 0.7430
Epoch 10/50 2027/2027	[=====]	2027/2027	[=====]	- 728s	359ms/step	- loss: 0.2103	- acc: 0.9975	- dice_coef: 0.7897	- val_loss: 0.2635	- val_acc: 0.9964	- val_dice_coef: 0.7365
Epoch 11/50 2027/2027	[=====]	2027/2027	[=====]	- 728s	359ms/step	- loss: 0.2001	- acc: 0.9976	- dice_coef: 0.7999	- val_loss: 0.1785	- val_acc: 0.9975	- val_dice_coef: 0.8215
Epoch 12/50 2027/2027	[=====]	2027/2027	[=====]	- 728s	359ms/step	- loss: 0.1959	- acc: 0.9976	- dice_coef: 0.8041	- val_loss: 0.6279	- val_acc: 0.9939	- val_dice_coef: 0.3721
Epoch 13/50 2027/2027	[=====]	2027/2027	[=====]	- 729s	359ms/step	- loss: 0.1920	- acc: 0.9977	- dice_coef: 0.8080	- val_loss: 0.1700	- val_acc: 0.9977	- val_dice_coef: 0.8300
Epoch 14/50 2027/2027	[=====]	2027/2027	[=====]	- 728s	359ms/step	- loss: 0.1826	- acc: 0.9978	- dice_coef: 0.8174	- val_loss: 0.1889	- val_acc: 0.9975	- val_dice_coef: 0.8111
Epoch 15/50 2027/2027	[=====]	2027/2027	[=====]	- 728s	359ms/step	- loss: 0.1869	- acc: 0.9978	- dice_coef: 0.8131	- val_loss: 0.1889	- val_acc: 0.9977	- val_dice_coef: 0.8191
Epoch 16/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1827	- acc: 0.9978	- dice_coef: 0.8173	- val_loss: 0.1948	- val_acc: 0.9971	- val_dice_coef: 0.8052
Epoch 17/50 2027/2027	[=====]	2027/2027	[=====]	- 722s	356ms/step	- loss: 0.1795	- acc: 0.9978	- dice_coef: 0.8205	- val_loss: 0.1634	- val_acc: 0.9978	- val_dice_coef: 0.8366
Epoch 18/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1852	- acc: 0.9978	- dice_coef: 0.8148	- val_loss: 0.2330	- val_acc: 0.9966	- val_dice_coef: 0.7670
Epoch 19/50 2027/2027	[=====]	2027/2027	[=====]	- 726s	358ms/step	- loss: 0.1724	- acc: 0.9979	- dice_coef: 0.8276	- val_loss: 0.1648	- val_acc: 0.9978	- val_dice_coef: 0.8360

Epoch 20/50 2027/2027	[=====]	2027/2027	[=====]	- 726s	358ms/step	- loss: 0.1732	- acc: 0.9979	- dice_coef: 0.8268	- val_loss: 0.1685	- val_acc: 0.9978	- val_dice_coef: 0.8315
Epoch 21/50 2027/2027	[=====]	2027/2027	[=====]	- 726s	358ms/step	- loss: 0.1682	- acc: 0.9980	- dice_coef: 0.8318	- val_loss: 0.1567	- val_acc: 0.9978	- val_dice_coef: 0.8433
Epoch 22/50 2027/2027	[=====]	2027/2027	[=====]	- 726s	358ms/step	- loss: 0.1653	- acc: 0.9980	- dice_coef: 0.8347	- val_loss: 0.1579	- val_acc: 0.9979	- val_dice_coef: 0.8421
Epoch 23/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1722	- acc: 0.9979	- dice_coef: 0.8276	- val_loss: 0.3536	- val_acc: 0.9960	- val_dice_coef: 0.6464
Epoch 24/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1630	- acc: 0.9980	- dice_coef: 0.8370	- val_loss: 0.1598	- val_acc: 0.9979	- val_dice_coef: 0.8402
Epoch 25/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1563	- acc: 0.9981	- dice_coef: 0.8437	- val_loss: 0.1500	- val_acc: 0.9979	- val_dice_coef: 0.8500
Epoch 26/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	358ms/step	- loss: 0.1580	- acc: 0.9981	- dice_coef: 0.8420	- val_loss: 0.1561	- val_acc: 0.9979	- val_dice_coef: 0.8439
Epoch 27/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1571	- acc: 0.9981	- dice_coef: 0.8429	- val_loss: 0.1717	- val_acc: 0.9978	- val_dice_coef: 0.8283
Epoch 28/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1506	- acc: 0.9982	- dice_coef: 0.8494	- val_loss: 0.1507	- val_acc: 0.9980	- val_dice_coef: 0.8493
Epoch 29/50 2027/2027	[=====]	2027/2027	[=====]	- 726s	359ms/step	- loss: 0.1497	- acc: 0.9982	- dice_coef: 0.8503	- val_loss: 0.1570	- val_acc: 0.9978	- val_dice_coef: 0.8430
Epoch 30/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	358ms/step	- loss: 0.1477	- acc: 0.9982	- dice_coef: 0.8523	- val_loss: 0.1477	- val_acc: 0.9980	- val_dice_coef: 0.8523
Epoch 31/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1478	- acc: 0.9982	- dice_coef: 0.8522	- val_loss: 0.2773	- val_acc: 0.9965	- val_dice_coef: 0.7227
Epoch 32/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1456	- acc: 0.9982	- dice_coef: 0.8544	- val_loss: 0.1512	- val_acc: 0.9980	- val_dice_coef: 0.8488
Epoch 33/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	358ms/step	- loss: 0.1455	- acc: 0.9982	- dice_coef: 0.8545	- val_loss: 0.2388	- val_acc: 0.9966	- val_dice_coef: 0.7612
Epoch 34/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1420	- acc: 0.9983	- dice_coef: 0.8572	- val_loss: 0.1377	- val_acc: 0.9981	- val_dice_coef: 0.8623
Epoch 35/50 2027/2027	[=====]	2027/2027	[=====]	- 726s	358ms/step	- loss: 0.1393	- acc: 0.9983	- dice_coef: 0.8607	- val_loss: 0.2255	- val_acc: 0.9970	- val_dice_coef: 0.7745
Epoch 36/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	358ms/step	- loss: 0.1371	- acc: 0.9983	- dice_coef: 0.8629	- val_loss: 0.1780	- val_acc: 0.9977	- val_dice_coef: 0.8220
Epoch 37/50 2027/2027	[=====]	2027/2027	[=====]	- 728s	359ms/step	- loss: 0.1400	- acc: 0.9983	- dice_coef: 0.8600	- val_loss: 0.1500	- val_acc: 0.9979	- val_dice_coef: 0.8500
Epoch 38/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1364	- acc: 0.9984	- dice_coef: 0.8636	- val_loss: 0.1927	- val_acc: 0.9974	- val_dice_coef: 0.8073

Epoch 39/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1305	- acc: 0.9984	- dice_coef: 0.8695	- val_loss: 0.1332	- val_acc: 0.9982	- val_dice_coef: 0.8668
Epoch 40/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1317	- acc: 0.9984	- dice_coef: 0.8683	- val_loss: 0.2249	- val_acc: 0.9970	- val_dice_coef: 0.7751
Epoch 41/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1316	- acc: 0.9984	- dice_coef: 0.8684	- val_loss: 0.2441	- val_acc: 0.9966	- val_dice_coef: 0.7559
Epoch 42/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1238	- acc: 0.9985	- dice_coef: 0.8762	- val_loss: 0.1293	- val_acc: 0.9983	- val_dice_coef: 0.8707
Epoch 43/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1267	- acc: 0.9985	- dice_coef: 0.8733	- val_loss: 0.1293	- val_acc: 0.9983	- val_dice_coef: 0.8707
Epoch 44/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1263	- acc: 0.9985	- dice_coef: 0.8737	- val_loss: 0.1258	- val_acc: 0.9983	- val_dice_coef: 0.8750
Epoch 45/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1216	- acc: 0.9985	- dice_coef: 0.8784	- val_loss: 0.1171	- val_acc: 0.9984	- val_dice_coef: 0.8829
Epoch 46/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1248	- acc: 0.9985	- dice_coef: 0.8752	- val_loss: 0.1144	- val_acc: 0.9984	- val_dice_coef: 0.8856
Epoch 47/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1252	- acc: 0.9985	- dice_coef: 0.8748	- val_loss: 0.1699	- val_acc: 0.9979	- val_dice_coef: 0.8301
Epoch 48/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1227	- acc: 0.9985	- dice_coef: 0.8773	- val_loss: 0.1450	- val_acc: 0.9980	- val_dice_coef: 0.8550
Epoch 49/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1180	- acc: 0.9986	- dice_coef: 0.8820	- val_loss: 0.1664	- val_acc: 0.9979	- val_dice_coef: 0.8336
Epoch 50/50 2027/2027	[=====]	2027/2027	[=====]	- 727s	359ms/step	- loss: 0.1162	- acc: 0.9986	- dice_coef: 0.8838	- val_loss: 0.1185	- val_acc: 0.9983	- val_dice_coef: 0.8815

**Model summary:**

• **Liver Segmentation**

Layer (Type)	Output Shape	Param #	Connected to
Input_1 (InputLayer)	(None, 256, 256, 3)	0	
conv2d_1 (Conv2D)	(None, 256, 256, 16)	448	input_1[0][0]
batch_normalization_1 (BatchNorm)	(None, 256, 256, 16)	64	conv2d_1[0][0]
activation_1 (Activation)	(None, 256, 256, 16)	0	batch_normalization_1[0][0]
conv2d_3 (Conv2D)	(None, 256, 256, 16)	64	input_1[0][0]
conv2d_2 (Conv2D)	(None, 256, 256, 16)	2320	activation_1[0][0]
batch_normalization_2 (BatchNorm)	(None, 256, 256, 16)	64	conv2d_3[0][0]
add_1 (Add)	(None, 256, 256, 16)	0	conv2d_2[0][0] batch_normalization_2[0][0]
batch_normalization_3 (BatchNorm)	(None, 256, 256, 16)	64	add_1[0][0]
activation_2 (Activation)	(None, 256, 256, 16)	0	batch_normalization_3[0][0]
conv2d_4 (Conv2D)	(None, 128, 128, 32)	4640	activation_2[0][0]
batch_normalization_4 (BatchNorm)	(None, 128, 128, 32)	128	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 128, 128, 32)	544	add_1[0][0]
activation_3 (Activation)	(None, 128, 128, 32)	0	batch_normalization_4[0][0]
batch_normalization_5 (BatchNorm)	(None, 128, 128, 32)	128	conv2d_5[0][0]
conv2d_5 (Conv2D)	(None, 128, 128, 32)	9248	activation_3[0][0]
add_2 (Add)	(None, 128, 128, 32)	0	batch_normalization_5[0][0] conv2d_5[0][0]
batch_normalization_6 (BatchNorm)	(None, 128, 128, 32)	128	add_2[0][0]
activation_4 (Activation)	(None, 128, 128, 32)	0	batch_normalization_6[0][0]
conv2d_7 (Conv2D)	(None, 64, 64, 64)	18496	activation_4[0][0]
batch_normalization_7 (BatchNorm)	(None, 64, 64, 64)	256	conv2d_7[0][0]
conv2d_9 (Conv2D)	(None, 64, 64, 64)	2112	add_2[0][0]
activation_5 (Activation)	(None, 64, 64, 64)	0	batch_normalization_7[0][0]
batch_normalization_8 (BatchNorm)	(None, 64, 64, 64)	256	conv2d_9[0][0]
conv2d_8 (Conv2D)	(None, 64, 64, 64)	36928	activation_5[0][0]
add_3 (Add)	(None, 64, 64, 64)	0	batch_normalization_8[0][0] conv2d_8[0][0]
conv2d_18 (Conv2D)	(None, 32, 32, 256)	884992	activation_12[0][0]
batch_normalization_18 (BatchNorm)	(None, 32, 32, 256)	1024	conv2d_18[0][0]
conv2d_20 (Conv2D)	(None, 32, 32, 256)	90560	concatenate_1[0][0]
activation_13 (Activation)	(None, 32, 32, 256)	0	batch_normalization_18[0][0]
batch_normalization_19 (BatchNorm)	(None, 32, 32, 256)	1024	conv2d_20[0][0]
conv2d_19 (Conv2D)	(None, 32, 32, 256)	590080	activation_13[0][0]
add_6 (Add)	(None, 32, 32, 256)	0	batch_normalization_19[0][0] conv2d_19[0][0]
up_sampling2d_2 (Upsampling2D)	(None, 64, 64, 256)	0	add_6[0][0]
concatenate_2 (Concatenate)	(None, 64, 64, 320)	0	up_sampling2d_2[0][0] add_3[0][0]
batch_normalization_20 (BatchNorm)	(None, 64, 64, 320)	1280	concatenate_2[0][0]
activation_14 (Activation)	(None, 64, 64, 320)	0	batch_normalization_20[0][0]
conv2d_21 (Conv2D)	(None, 64, 64, 128)	368768	activation_14[0][0]
batch_normalization_21 (BatchNorm)	(None, 64, 64, 128)	512	conv2d_21[0][0]
conv2d_23 (Conv2D)	(None, 64, 64, 128)	41088	concatenate_2[0][0]
activation_15 (Activation)	(None, 64, 64, 128)	0	batch_normalization_21[0][0]
batch_normalization_22 (BatchNorm)	(None, 64, 64, 128)	512	conv2d_23[0][0]
conv2d_22 (Conv2D)	(None, 64, 64, 128)	147584	activation_15[0][0]
add_7 (Add)	(None, 64, 64, 128)	0	batch_normalization_22[0][0] conv2d_22[0][0]
up_sampling2d_3 (Upsampling2D)	(None, 128, 128, 128)	0	add_7[0][0]
concatenate_3 (Concatenate)	(None, 128, 128, 160)	0	up_sampling2d_3[0][0] add_2[0][0]
batch_normalization_23 (BatchNorm)	(None, 128, 128, 160)	640	concatenate_3[0][0]
activation_16 (Activation)	(None, 128, 128, 160)	0	batch_normalization_23[0][0]
conv2d_24 (Conv2D)	(None, 128, 128, 64)	92224	activation_16[0][0]
batch_normalization_24 (BatchNorm)	(None, 128, 128, 64)	256	conv2d_24[0][0]
conv2d_26 (Conv2D)	(None, 128, 128, 64)	10304	concatenate_3[0][0]
activation_17 (Activation)	(None, 128, 128, 64)	0	batch_normalization_24[0][0]
batch_normalization_25 (BatchNorm)	(None, 128, 128, 64)	256	conv2d_26[0][0]

batch_normalization_9 (BatchNorm)	(None, 64, 64, 64)	256	add_3[0][0]
activation_6 (Activation)	(None, 64, 64, 64)	0	batch_normalization_9[0][0]
conv2d_10 (Conv2D)	(None, 32, 32, 128)	73856	activation_6[0][0]
batch_normalization_10 (BatchNorm)	(None, 32, 32, 128)	512	conv2d_10[0][0]
conv2d_12 (Conv2D)	(None, 32, 32, 128)	8320	add_3[0][0]
activation_7 (Activation)	(None, 32, 32, 128)	0	batch_normalization_10[0][0]
batch_normalization_11 (BatchNorm)	(None, 32, 32, 128)	512	conv2d_12[0][0]
conv2d_11 (Conv2D)	(None, 32, 32, 128)	147584	activation_7[0][0]
add_4 (Add)	(None, 32, 32, 128)	0	batch_normalization_11[0][0] conv2d_11[0][0]
batch_normalization_12 (BatchNorm)	(None, 32, 32, 128)	512	add_4[0][0]
activation_8 (Activation)	(None, 32, 32, 128)	0	batch_normalization_12[0][0]
conv2d_13 (Conv2D)	(None, 16, 16, 256)	295168	activation_8[0][0]
batch_normalization_13 (BatchNorm)	(None, 16, 16, 256)	1024	conv2d_13[0][0]
conv2d_15 (Conv2D)	(None, 16, 16, 256)	33824	add_4[0][0]
activation_9 (Activation)	(None, 16, 16, 256)	0	batch_normalization_13[0][0]
batch_normalization_14 (BatchNorm)	(None, 16, 16, 256)	1024	conv2d_15[0][0]
conv2d_14 (Conv2D)	(None, 16, 16, 256)	590080	activation_9[0][0]
add_5 (Add)	(None, 16, 16, 256)	0	batch_normalization_14[0][0] conv2d_14[0][0]
batch_normalization_15 (BatchNorm)	(None, 16, 16, 256)	1024	add_5[0][0]
activation_10 (Activation)	(None, 16, 16, 256)	0	batch_normalization_15[0][0]
conv2d_16 (Conv2D)	(None, 16, 16, 256)	590080	activation_10[0][0]
batch_normalization_16 (BatchNorm)	(None, 16, 16, 256)	1024	conv2d_16[0][0]
activation_11 (Activation)	(None, 16, 16, 256)	0	batch_normalization_16[0][0]
conv2d_17 (Conv2D)	(None, 16, 16, 256)	590080	activation_11[0][0]
up_sampling2d_1 (Upsampling2D)	(None, 32, 32, 256)	0	conv2d_17[0][0]
concatenate_1 (Concatenate)	(None, 32, 32, 384)	0	up_sampling2d_1[0][0] add_4[0][0]
batch_normalization_17 (BatchNorm)	(None, 32, 32, 384)	1536	concatenate_1[0][0]
activation_12 (Activation)	(None, 32, 32, 384)	0	batch_normalization_17[0][0]
conv2d_25 (Conv2D)	(None, 128, 128, 64)	36928	activation_17[0][0]
add_8 (Add)	(None, 128, 128, 64)	0	batch_normalization_25[0][0] conv2d_25[0][0]
up_sampling2d_4 (Upsampling2D)	(None, 256, 256, 64)	0	add_8[0][0]
concatenate_4 (Concatenate)	(None, 256, 256, 80)	0	up_sampling2d_4[0][0] add_1[0][0]
batch_normalization_26 (BatchNorm)	(None, 256, 256, 80)	320	concatenate_4[0][0]
activation_18 (Activation)	(None, 256, 256, 80)	0	batch_normalization_26[0][0]
conv2d_27 (Conv2D)	(None, 256, 256, 32)	23072	activation_18[0][0]
batch_normalization_27 (BatchNorm)	(None, 256, 256, 32)	128	conv2d_27[0][0]
conv2d_29 (Conv2D)	(None, 256, 256, 32)	2592	concatenate_4[0][0]
activation_19 (Activation)	(None, 256, 256, 32)	0	batch_normalization_27[0][0]
batch_normalization_28 (BatchNorm)	(None, 256, 256, 32)	128	conv2d_29[0][0]
conv2d_28 (Conv2D)	(None, 256, 256, 32)	9248	activation_19[0][0]
add_9 (Add)	(None, 256, 256, 32)	0	batch_normalization_28[0][0] conv2d_28[0][0]
conv2d_30 (Conv2D)	(None, 256, 256, 1)	33	add_9[0][0]



• Tumor Segmentation

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 256, 256, 3)	0	
conv2d (Conv2D)	(None, 256, 256, 16)	448	input_1[0][0]
batch_normalization_v1 (Batch Normalization)	(None, 256, 256, 16)	64	conv2d[0][0]
activation (Activation)	(None, 256, 256, 16)	0	batch_normalization_v1[0][0]
conv2d_2 (Conv2D)	(None, 256, 256, 16)	64	input_1[0][0]
conv2d_1 (Conv2D)	(None, 256, 256, 16)	2320	activation[0][0]
batch_normalization_v1_1 (Batch Normalization)	(None, 256, 256, 16)	64	conv2d_2[0][0]
add (Add)	(None, 256, 256, 16)	0	conv2d_1[0][0] batch_normalization_v1_1[0][0]
batch_normalization_v1_2 (Batch Normalization)	(None, 256, 256, 16)	64	add[0][0]
activation_1 (Activation)	(None, 256, 256, 16)	0	batch_normalization_v1_2[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 32)	4640	activation_1[0][0]
batch_normalization_v1_3 (Batch Normalization)	(None, 128, 128, 32)	128	conv2d_3[0][0]
conv2d_5 (Conv2D)	(None, 128, 128, 32)	544	add[0][0]
activation_2 (Activation)	(None, 128, 128, 32)	0	batch_normalization_v1_3[0][0]
batch_normalization_v1_4 (Batch Normalization)	(None, 128, 128, 32)	128	conv2d_5[0][0]
conv2d_4 (Conv2D)	(None, 128, 128, 32)	9248	activation_2[0][0]
add_1 (Add)	(None, 128, 128, 32)	0	batch_normalization_v1_4[0][0] conv2d_4[0][0]
batch_normalization_v1_5 (Batch Normalization)	(None, 128, 128, 32)	128	add_1[0][0]
activation_3 (Activation)	(None, 128, 128, 32)	0	batch_normalization_v1_5[0][0]
conv2d_6 (Conv2D)	(None, 64, 64, 64)	18496	activation_3[0][0]
batch_normalization_v1_6 (Batch Normalization)	(None, 64, 64, 64)	256	conv2d_6[0][0]
conv2d_8 (Conv2D)	(None, 64, 64, 64)	2112	add_1[0][0]
activation_4 (Activation)	(None, 64, 64, 64)	0	batch_normalization_v1_6[0][0]
batch_normalization_v1_7 (Batch Normalization)	(None, 64, 64, 64)	256	conv2d_8[0][0]

conv2d_7 (Conv2D)	(None, 64, 64, 64)	36928	activation_4[0][0]
add_2 (Add)	(None, 64, 64, 64)	0	batch_normalization_v1_7[0][0] conv2d_7[0][0]
batch_normalization_v1_8 (Batch Normalization)	(None, 64, 64, 64)	256	add_2[0][0]
activation_5 (Activation)	(None, 64, 64, 64)	0	batch_normalization_v1_8[0][0]
conv2d_9 (Conv2D)	(None, 32, 32, 128)	73856	activation_5[0][0]
batch_normalization_v1_9 (Batch Normalization)	(None, 32, 32, 128)	512	conv2d_9[0][0]
conv2d_11 (Conv2D)	(None, 32, 32, 128)	8320	add_2[0][0]
activation_6 (Activation)	(None, 32, 32, 128)	0	batch_normalization_v1_9[0][0]
batch_normalization_v1_10 (Batch Normalization)	(None, 32, 32, 128)	512	conv2d_11[0][0]
conv2d_10 (Conv2D)	(None, 32, 32, 128)	147584	activation_6[0][0]
add_3 (Add)	(None, 32, 32, 128)	0	batch_normalization_v1_10[0][0] conv2d_10[0][0]
batch_normalization_v1_11 (Batch Normalization)	(None, 32, 32, 128)	512	add_3[0][0]
activation_7 (Activation)	(None, 32, 32, 128)	0	batch_normalization_v1_11[0][0]
conv2d_12 (Conv2D)	(None, 16, 16, 256)	295168	activation_7[0][0]
batch_normalization_v1_12 (Batch Normalization)	(None, 16, 16, 256)	1024	conv2d_12[0][0]
conv2d_14 (Conv2D)	(None, 16, 16, 256)	33024	add_3[0][0]
activation_8 (Activation)	(None, 16, 16, 256)	0	batch_normalization_v1_12[0][0]
batch_normalization_v1_13 (Batch Normalization)	(None, 16, 16, 256)	1024	conv2d_14[0][0]
conv2d_13 (Conv2D)	(None, 16, 16, 256)	590080	activation_8[0][0]
add_4 (Add)	(None, 16, 16, 256)	0	batch_normalization_v1_13[0][0] conv2d_13[0][0]
batch_normalization_v1_14 (Batch Normalization)	(None, 16, 16, 256)	1024	add_4[0][0]
activation_9 (Activation)	(None, 16, 16, 256)	0	batch_normalization_v1_14[0][0]
conv2d_15 (Conv2D)	(None, 16, 16, 256)	590080	activation_9[0][0]
batch_normalization_v1_15 (Batch Normalization)	(None, 16, 16, 256)	1024	conv2d_15[0][0]
activation_10 (Activation)	(None, 16, 16, 256)	0	batch_normalization_v1_15[0][0]

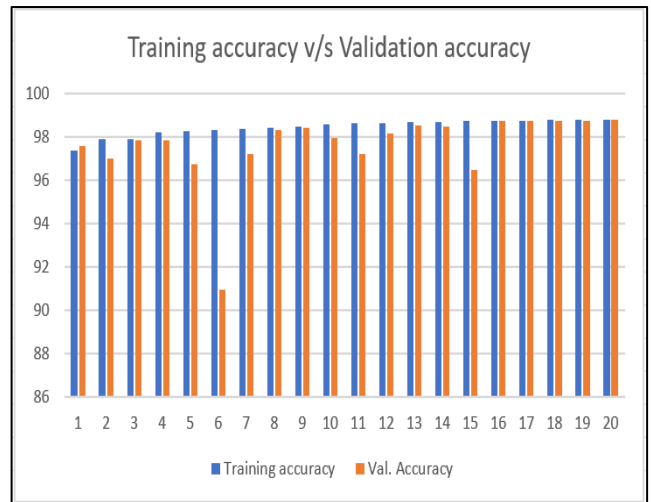
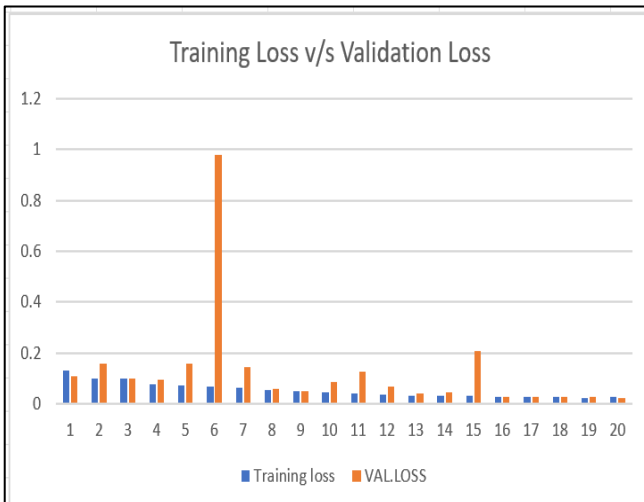
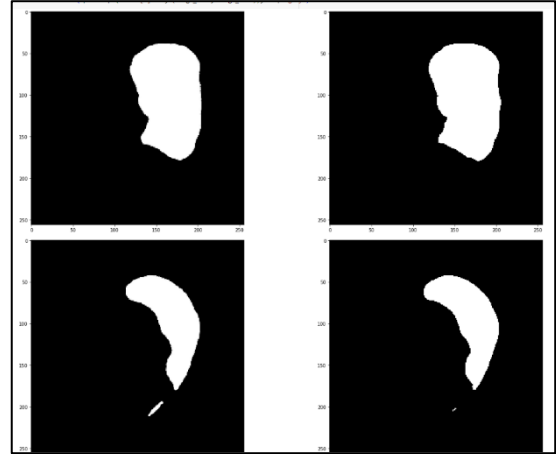
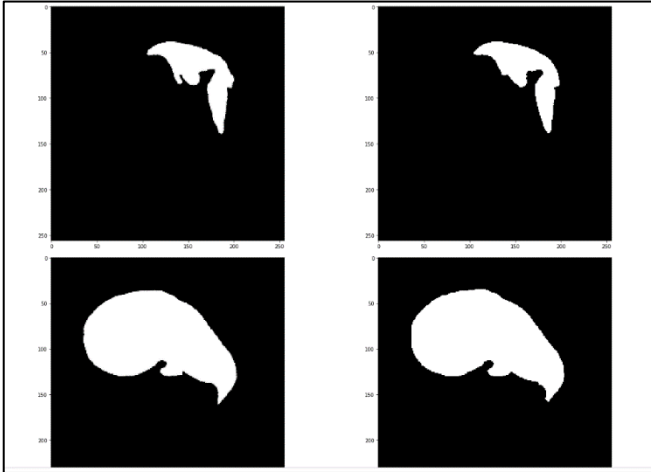
conv2d_16 (Conv2D)	(None, 16, 16, 256)	590080	activation_10[0][0]
up_sampling2d (UpSampling2D)	(None, 32, 32, 256)	0	conv2d_16[0][0]
concatenate (Concatenate)	(None, 32, 32, 384)	0	up_sampling2d[0][0] add_3[0][0]
batch_normalization_v1_16 (Batch Normalization)	(None, 32, 32, 384)	1536	concatenate[0][0]
activation_11 (Activation)	(None, 32, 32, 384)	0	batch_normalization_v1_16[0][0]
conv2d_17 (Conv2D)	(None, 32, 32, 256)	804992	activation_11[0][0]
batch_normalization_v1_17 (Batch Normalization)	(None, 32, 32, 256)	1024	conv2d_17[0][0]
conv2d_19 (Conv2D)	(None, 32, 32, 256)	98560	concatenate[0][0]
activation_12 (Activation)	(None, 32, 32, 256)	0	batch_normalization_v1_17[0][0]
batch_normalization_v1_18 (Batch Normalization)	(None, 32, 32, 256)	1024	conv2d_19[0][0]
conv2d_18 (Conv2D)	(None, 32, 32, 256)	590080	activation_12[0][0]
add_5 (Add)	(None, 32, 32, 256)	0	batch_normalization_v1_18[0][0] conv2d_18[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 64, 64, 256)	0	add_5[0][0]
concatenate_1 (Concatenate)	(None, 64, 64, 320)	0	up_sampling2d_1[0][0] add_2[0][0]
batch_normalization_v1_19 (Batch Normalization)	(None, 64, 64, 320)	1280	concatenate_1[0][0]
activation_13 (Activation)	(None, 64, 64, 320)	0	batch_normalization_v1_19[0][0]
conv2d_20 (Conv2D)	(None, 64, 64, 128)	368768	activation_13[0][0]
batch_normalization_v1_20 (Batch Normalization)	(None, 64, 64, 128)	512	conv2d_20[0][0]
conv2d_22 (Conv2D)	(None, 64, 64, 128)	41088	concatenate_1[0][0]
activation_14 (Activation)	(None, 64, 64, 128)	0	batch_normalization_v1_20[0][0]
batch_normalization_v1_21 (Batch Normalization)	(None, 64, 64, 128)	512	conv2d_22[0][0]
conv2d_21 (Conv2D)	(None, 64, 64, 128)	147584	activation_14[0][0]
add_6 (Add)	(None, 64, 64, 128)	0	batch_normalization_v1_21[0][0] conv2d_21[0][0]
up_sampling2d_2 (UpSampling2D)	(None, 128, 128, 128)	0	add_6[0][0]
concatenate_2 (Concatenate)	(None, 128, 128, 160)	0	up_sampling2d_2[0][0] add_1[0][0]

add_1[0][0]			
batch_normalization_v1_22 (Batch Normalization)	(None, 128, 128, 160)	640	concatenate_2[0][0]
activation_15 (Activation)	(None, 128, 128, 160)	0	batch_normalization_v1_22[0][0]
conv2d_23 (Conv2D)	(None, 128, 128, 64)	92224	activation_15[0][0]
batch_normalization_v1_23 (Batch Normalization)	(None, 128, 128, 64)	256	conv2d_23[0][0]
conv2d_25 (Conv2D)	(None, 128, 128, 64)	10304	concatenate_2[0][0]
activation_16 (Activation)	(None, 128, 128, 64)	0	batch_normalization_v1_23[0][0]
batch_normalization_v1_24 (Batch Normalization)	(None, 128, 128, 64)	256	conv2d_25[0][0]
conv2d_24 (Conv2D)	(None, 128, 128, 64)	36928	activation_16[0][0]
add_7 (Add)	(None, 128, 128, 64)	0	batch_normalization_v1_24[0][0] conv2d_24[0][0]
up_sampling2d_3 (UpSampling2D)	(None, 256, 256, 64)	0	add_7[0][0]
concatenate_3 (Concatenate)	(None, 256, 256, 80)	0	up_sampling2d_3[0][0] add_0[0][0]
batch_normalization_v1_25 (Batch Normalization)	(None, 256, 256, 80)	320	concatenate_3[0][0]
activation_17 (Activation)	(None, 256, 256, 80)	0	batch_normalization_v1_25[0][0]
conv2d_26 (Conv2D)	(None, 256, 256, 32)	23072	activation_17[0][0]
batch_normalization_v1_26 (Batch Normalization)	(None, 256, 256, 32)	128	conv2d_26[0][0]
conv2d_28 (Conv2D)	(None, 256, 256, 32)	2592	concatenate_3[0][0]
activation_18 (Activation)	(None, 256, 256, 32)	0	batch_normalization_v1_26[0][0]
batch_normalization_v1_27 (Batch Normalization)	(None, 256, 256, 32)	128	conv2d_28[0][0]
conv2d_27 (Conv2D)	(None, 256, 256, 32)	9248	activation_18[0][0]
add_8 (Add)	(None, 256, 256, 32)	0	batch_normalization_v1_27[0][0] conv2d_27[0][0]
conv2d_29 (Conv2D)	(None, 256, 256, 1)	33	add_8[0][0]

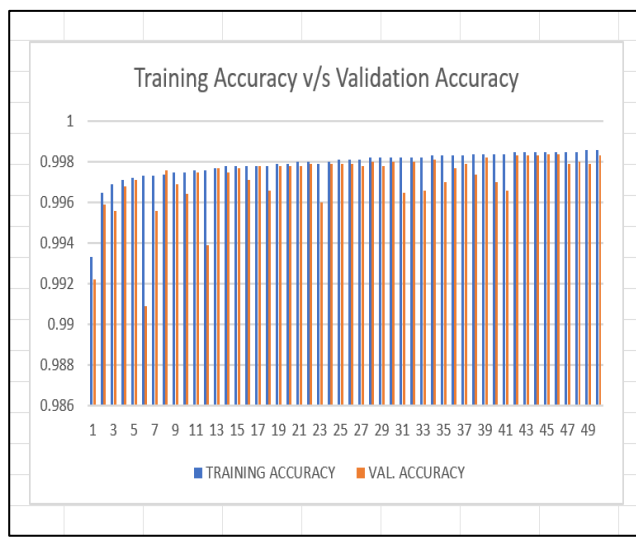
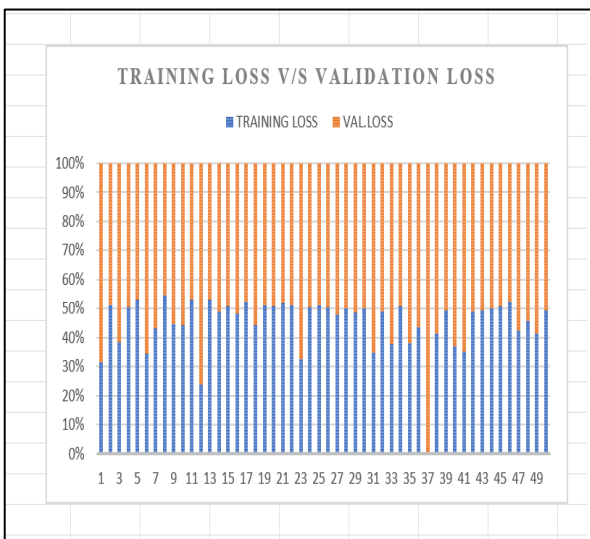
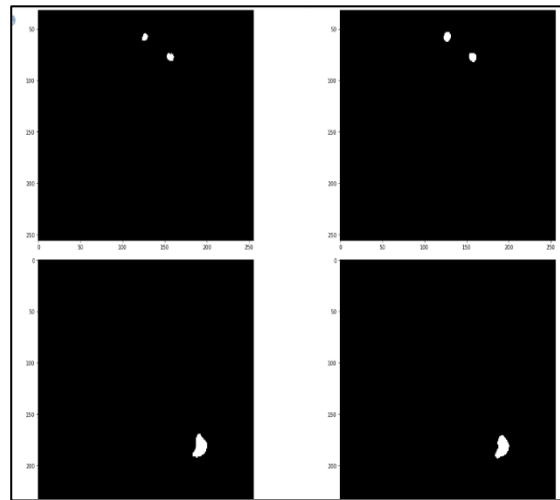
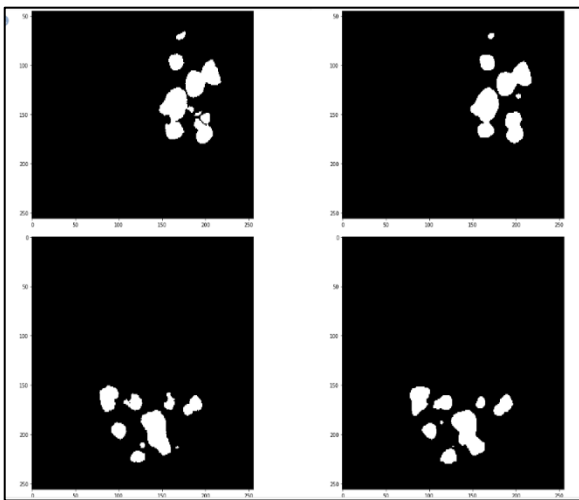
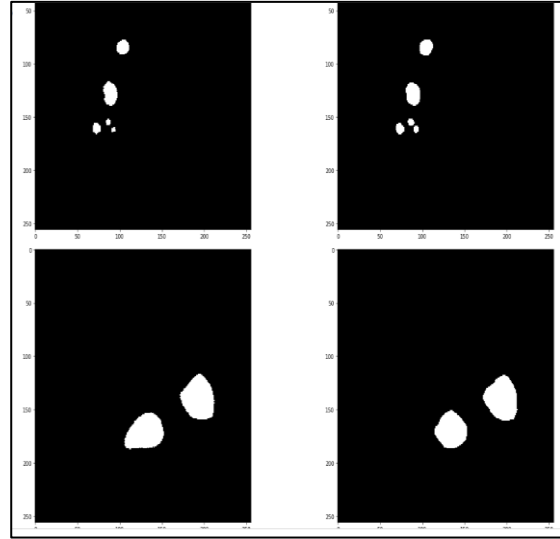
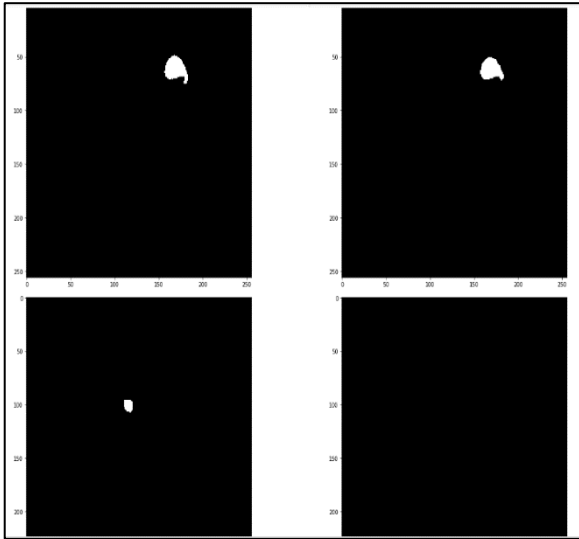
## IV. RESULTS

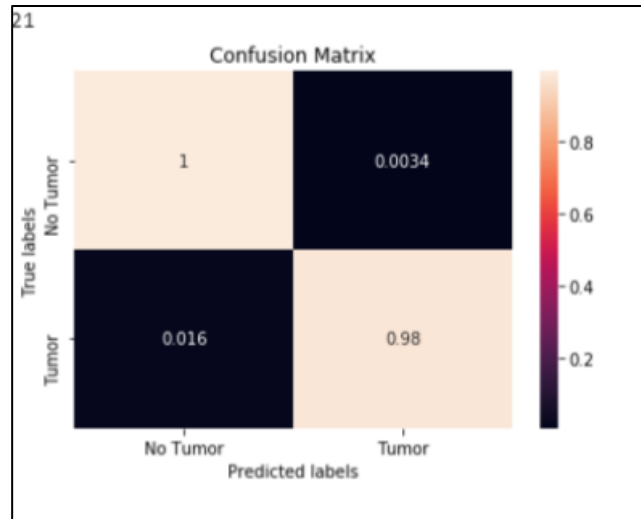
### 4.1 RESULTS

- Liver Segmentation



- Tumor Segmentation





## V. CONCLUSION AND FUTURE WORK

### 5.1 CONCLUSION

Automated liver tumor segmentation using deep learning has emerged as a powerful tool for medical picture analysis. In this last section, we summarize the significant results and contributions of the research on liver tumor segmentation using deep learning, emphasizing their clinical importance and potential effect on clinical practice. We also discuss challenges faced throughout the research and provide recommendations for future development of this technology.

The research demonstrates that automated segmentation algorithms based on deep learning can accurately identify liver tumors in medical images. In order to effectively segregate tumors and capture their complicated architecture, convolutional neural networks (CNNs) and encoder-decoder designs, such U-Net, have shown to be effective. Automatic segmentation based on deep learning is more efficient, reliable, and scalable than manual segmentation. As a result, large-scale clinical datasets are now possible without the need for laborious and time-consuming human annotation.

The liver and liver tumors have been the focus of numerous deep learning models developed for early detection. Automatic segmentation algorithms are necessary for developing a reliable radiation therapy treatment plan. In this study, we introduced the Liver Tumor Segmentation (LiTS) standard. This study details the use of a deep learning model to the problem of segmenting CT images of tumors and livers. To train and test the suggested model, we turned to the standard 3D-IRCADB1 dataset. This work provides a comprehensive literature assessment of 19 deep learning publications, all of which discuss automated liver tumor segmentation. This paper provides a summary of Neural networks (both Convolutional and U-Nets) and their applications to liver segmentation. The writers of the literature review reflected on and addressed a wide range of issues and ideas. The techniques used throughout the studies all stem from the realm of deep learning, yielding reliable outcomes.

In conclusion, automated liver tumor segmentation using deep learning has the potential to greatly improve both liver tumor diagnosis and treatment. By more precisely defining tumors, this technology may help doctors provide prompt, individualized treatments that ultimately lead to better patient outcomes. If the medical and AI communities can work together to learn more about deep learning-based liver tumour segmentation, it may become an indispensable tool in the fight against liver cancer.

### 5.2 FUTURE WORK

Researchers entering or already active in the same area might investigate the methods for the additional alterations that will improve outcomes. To make the model more stable and applicable, it may be necessary to use a larger training dataset. The precision of deep learning models may improve with time, allowing for more precise and consistent tumor segmentation in the liver. This might lead to more accurate diagnosis and improved therapy

tracking. Automatic segmentation has the potential to reduce health care disparities throughout the globe by providing accurate tumor analysis in regions with less medical expertise.

Automatic liver tumor segmentation based on deep learning has the potential to greatly improve diagnostics and therapy. The science of AI will advance and be able to influence how it impacts patient care if practitioners, researchers, and medical professionals work together more closely.

## REFERENCES

- [1] H. Kong, "FULLY AUTOMATED LIVER SEGMENTATION FOR LOW- AND HIGH- CONTRAST CT VOLUMES BASED ON PROBABILISTIC ATLASES Biomedical & Multimedia Information Technology Research Group , University of Sydney , NSW Department of PET and Nuclear Medicine , Royal Prince Alfred," *Methods*, pp. 1733–1736, 2010.
- [2]. Weimin Huang, Ning Li, Ziping Lin, Guang-Bin Huang, Weiwei Zong, Jiayin Zhou, Yuping Duan, "Liver Tumor Detection and Segmentation using Kernel-based Extreme Learning Machine" *white paper*.
- [3]. Belgherbi, A., Hadjidj, I., Bessaid, A, "Semi-automated Method for the Liver Lesion Extraction from a CT Images Based on Mathematical Morphology", *iMedPub Journals Journal Of Biomedical Sciences*, Vol. 2 No. 2:4, doi: 10.3823/1019, PP 1-9,2013.
- [4]. R. Rajagopal, P.Subbiah, "Computer Aided Detection of Liver Tumor using SVM Classifier", *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, ISSN (Print) : 2320 – 3765, ISSN (Online): 2278 – 8875, Vol. 3, Issue 6, PP 10170-10177, June 2014.
- [5]. A.Anuja Merlyn, A.Anuba Merlyn, "A Systematic Analysis System for CT Liver Image Classification and Image Segmentation by Local Entropy Method", *IJISSET - International Journal of Innovative Science, Engineering & Technology*, ISSN 2348 – 7968, Vol. 1 Issue 3, PP 479-483, May 2014.
- [6]. Dr. S. Vijayarani, Mr.S.Dhayanand, "Liver Disease Prediction using SVM and Naïve Bayes Algorithms", *International Journal of Science, Engineering and Technology Research (IJSETR)*, ISSN: 2278 – 7798, Volume 4, Issue 4, PP 816-820, April 2015.
- [7]. "Liver Tumor Detection using Artificial Neural Networks for Medical Images" *IJIRST International Journal for Innovative Research in Science & Technology*, Volume 2, Issue 03, ISSN (online): 2349-6010, PP 34-38, August 2015.
- [8] P. P. R and P. B. Murthy M, "Liver Cancer Analysis using Machine Learning Techniques? A Review," *Int. J. Eng. Res. Technol.*, vol. 5, no. 22, pp. 1–4, 2017, [Online]. Available: [www.who.int](http://www.who.int)
- [9] Y. Enokiya, Y. Iwamoto, Y.-W. Chen, and X.-H. Han, "Automatic Liver Segmentation Using U-Net with Wasserstein GANs," *J. Image Graph.*, vol. 6, no. 2, pp. 152–159, 2018, doi: 10.18178/joig.6.2.152-159
- [10] X. Li, H. Chen, X. Qi, Q. Dou, C. W. Fu, and P. A. Heng, "H-DenseUNet: Hybrid Densely Connected UNet for Liver and Tumor Segmentation from CT Volumes," *IEEE Trans. Med. Imaging*, vol. 37, no. 12, pp. 2663–2674, 2018, doi: 10.1109/TMI.2018.2845918.
- [11] L. Chen *et al.*, "Liver tumor segmentation in CT volumes using an adversarial densely connected network," *BMC Bioinformatics*, vol. 20, no. Suppl 16, pp. 1–13, 2019, doi: 10.1186/s12859-019-3069-x.
- [12] P. Bilic *et al.*, "The Liver Tumor Segmentation Benchmark (LiTS)," no. January, 2019, [Online]. Available: <http://arxiv.org/abs/1901.04056>
- [13] S. Naeem, A. Ali, S. Qadri, and W. K. Mashwani, "applied sciences Machine-Learning Based Hybrid-Feature Analysis for Liver Cancer Classification Using Fused (MR and CT) Images," 2020.

- 
- [14] T. Saba, "Recent advancement in cancer detection using machine learning: Systematic survey of decades, comparisons and challenges," *J. Infect. Public Health*, vol. 13, no. 9, pp. 1274–1289, 2020, doi: 10.1016/j.jiph.2020.06.033.
- [15] A. Kalsoom, A. Moin, M. Maqsood, I. Mehmood, and S. Rho, "An Efficient Liver Tumor Detection using Machine Learning," *Proc. - 2020 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2020*, pp. 706–711, 2020, doi: 10.1109/CSCSI51800.2020.00130.
- [16] M. Islam, K. N. Khan, and M. S. Khan, "Evaluation of Preprocessing Techniques for U-Net Based Automated Liver Segmentation," *2021 Int. Conf. Artif. Intell. ICAI 2021*, pp. 187–192, 2021, doi: 10.1109/ICAI52203.2021.9445204
- [17] M. Rela, S. N. Rao, and P. R. Reddy, "Performance analysis of liver tumor classification using machine learning algorithms," *Int. J. Adv. Technol. Eng. Explore.*, vol. 9, no. 86, pp. 143–154, 2022, doi: 10.19101/IJATEE.2021.87465.
- [18] R. A. Khan, Y. Luo, and F. X. Wu, "Machine learning based liver disease diagnosis: A systematic review," *Neurocomputing*, vol. 468, pp. 492–509, 2022, doi: 10.1016/j.neucom.2021.08.138.
- [19] S. Lysdahlgaard, "Comparing Radiomics features of tumour and healthy liver tissue in a limited CT dataset: A machine learning study," *Radiography*, vol. 28, no. 3, pp. 718–724, 2022, doi: 10.1016/j.radi.2022.03.015.