3D Image Optimization Using Adam Algorithm

A PROJECT REPORT

Submitted by

Venkatesh T (910619205068)
Vinod T H (910619205069)

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

K.L.N.COLLEGE OF ENGINEERING, POTTAPALAYAM
(An Autonomous Institution, Affiliated to University, Chennai)

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"3D Image Optimization Using Adam Algorithm"** is the bona-fide work of **"Mr. T. Venkatesh (Reg. No. 910619205068), Mr. T.H. Vinod (Reg.No.910619205069)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. P.Ganesh Kumar	**Dr.J.S.Kanchana,**
B.E(I&C),M.E(APPL.ELECS.),Ph.D.	**B.E(CSE), M.E(CSE), Ph.D.,**
HEAD OF THE DEPARTMENT	**PROFESSOR**
INFORMATION TECHNOLOGY	**INFORMATION TECHNOLOGY**
K.L.N.COLLEGE OF ENGINEERING	**K.L.N.COLLEGE OF ENGINEERING**
(An ISO 9001-2008 Certified Institution)	(An ISO 9001-2008 Certified Institution)
POTTAPALAYAM, SIVAGANGAI,	**POTTAPALAYAM, SIVAGANGAI,**
TAMIL NADU, INDIA.	**TAMIL NADU, INDIA.**

Submitted for the Project viva-voce conducted on ______________

Internal Examiner **External Examiner**

Acknowledgement

Any work would be unfulfilled without a word of thanks. We hereby take pleasure in acknowledging the persons who guided me throughout our work.

First and foremost, thanks are to the omnipotent for providing us with his abundant blessings all throughout. We all extend our heartfelt thanks to **Er.K.N.K.KARTHIK, B.E.,** President of our college and **Dr.A.V.RAMPRASAD, M.E., Ph.D.,** Principal for provisioning us with the all required.

We esteem our self to articulate our sincere thanks to **Dr.P.Ganesh Kumar, B.E(I&C),M.E(APPL.ELECS.),Ph.D.** Head of the Department for leading us towards the zenith of success.

We express our grateful thanks to our **Project Guide** and **Project Coordinator Dr.J.S.Kanchana, B.E(CSE), M.E(CSE), Ph.D.,** for their invaluable guidance and motivation. Their assistance and advices had been very helpful throughout our project. I would like to thank all teaching and non-teaching staffs of our department who have been the sources of encouragement and ideas. I thank them for lending their support whenever needed.

Abstract

Image segmentation is the process of breaking a picture into collections of pixel regions that are each represented by a mask or labelled image. Medical image segmentation involves the extraction of regions of interest (ROIs) from 3D image data, such as from Magnetic Resonance Imaging (MRI) or Computed Tomography (CT) scans. The main goal of segmenting this data is to identify areas of the anatomy required for a particular study.

U-net is an image segmentation architecture built using CNN. It has a simpler structure i.e., it's a repetition of basic building blocks such as convolutions, ReLu, max pooling. The biggest advantage of U-Net architecture is that it will work with little amount of training data. In medical field there won't be abundant data available so U-Net is the best fit for this purpose.

The primary disadvantage of U-net is the slow learning rate at the deeper model's intermediate layers. Adam Algorithm is an optimization algorithm that can be used to speed up the learning rate of the deep learning models when compared with of stochastic gradient descent. Adam creates an optimization algorithm that can manage sparse gradients on noisy problems by combining the best elements of the Momentum and RMSProp algorithms. The learning rate will be increased by incorporating the Adam algorithm into the U-net design.

The metrics used to find the performance of the Deep learning model built from the U-Net architecture is Dice Coefficient, loss value and as well as time required for each Epoch.

TABLE OF CONTENTS

Chapter No	Title	Page No
	ABSTRACT	**i**
	LIST OF TABLES	**v**
	LIST OF FIGURES	**vii**
1	**INTRODUCTION**	1
	1.1 Introduction	2
	1.2 Problem Statement	3
	1.3 Project Objective	3
	1.4 Scope of the Project	3
	1.4.1 Existing System	3
	1.4.2 Proposed System	3
	1.5 Software Life Cycle Models	4
	1.5.1 Types	5
	1.5.2 Agile Model	5
	1.5.3 Reason for choosing the model	6
	1.6 Project Plan	7
	1.7 Summary	8
2	**LITERATURE REVIEW**	9
	2.1 Introduction	10
	2.2 Literature Review	10
	2.3 Summary	11

3 SYSTEM ANALYSIS 12

3.1 Introduction 13

3.2 Requirement Analysis 13

3.2.1 Functional Requirements 13

3.2.2 Non-Functional Requirements 13

3.2.3 Hardware Requirements 14

3.2.4 Software Requirements 14

3.2.5 Module Specification 14

3.2.5.1 Data Pre-Processing 14

3.2.5.2 U-Net architecture 15

3.2.5.3 Adam Algorithm 16

3.2.5.4 Performance Evaluation 18

3.3 Summary 18

4 SYSTEM DESIGN 19

4.1 Introduction 20

4.2 Data Flow Diagram 20

4.3 Object Oriented Analysis and Design 21

4.3.1 Detailed Design 21

4.3.1.1 Use Case Diagram 21

4.3.1.2 Activity Diagram 22

4.3.1.3 Class Diagram 23

4.3.1.4 Sequence Diagram 23

4.3.1.5 Communication Diagram 24

4.4 Summary 24

5	**IMPLEMENTATION**	25
6	**TESTING**	55
	6.1 Introduction	56
	6.2 Software Testing	56
	6.3 Basic Types of Testing	56
	6.4 Testing Techniques	57
	6.4.1 Unit Testing	57
	6.4.2 Integration Testing	58
	6.4.3 System Testing	58
	6.4.4 Performance Testing	59
	6.5 Summary	60
7	**SCREENSHOTS**	61
8	**CONCLUSION**	66
	8.1 Conclusion	67
	8.2 Future Enhancement	67
9	**APPENDIX**	68
10	**REFERENCES**	76

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
1.6.1	Project Plan	7
7.1	Comparison of Existing and Proposed System	64
7.2	All the Value of Dice Coefficient with respect to the Epoch	65

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.5.1	Agile model	6
3.2 1	Dataset containing CT scan	15
3.2.2	Comparison of learning path between Gradient Descent and Adam Algorithm	18
4.2.1	Data Flow Diagram	20
4.3.1	Use Case diagram	21
4.3.2	Activity diagram	22
4.3.3	Class diagram	23
4.3.4	Sequence diagram	23
4.3.5	Communication diagram	24
7.1	Data Pre-Processing	62
7.2	Building U-Net architecture	62
7.3	Web app home page	63
7.4	Uploading the data via web application.	63
7.5	Output gif image is show	64
7.6	Important stages of the output	64
7.7	comparison between orginal image and predicted image	65
9.1.1	python website home page	70
9.1.2	software installation	70
9.1.3	python installation checking	71
9.2.1	Anaconda distribution home page	71
9.2.2	Anaconda navigation GUI	72

1. INTRODUCTION

1.1 INTRODUCTION CHAPTER 1

Image segmentation is the process of partitioning of digital images into various parts or regions of pixels reducing the complexities of understanding the images to machines. Image segmentation could also involve separating the foreground from the background or assembling of pixels based on various similarities in the colour or shape. Various image segmentation algorithms are used to split and group a certain set of pixels together from the image. It is actually the task of assigning the labels to pixels and the pixels with the same label fall under a category where they have some or the other thing common in them.

In medical field Image segmentation is one of the most used technologies for extracting the regions of interest (ROIs) from a 3D image data such as from Magnetic Resonance Imaging (MRI) or Computed Tomography (CT) scans. The main goal of segmenting this data is to identify areas of the anatomy required for a particular study. For processing an 3D image and making an effective image segmentation we need to go for a well-designed deep learning model. Our project is focused on segmenting the liver part from the CT scan.

U-Net architecture is one of the most used deep learning models for image segmentation, it can be used for both 2D and 3D images. U-Net architecture is built using convolution neural network and it has a simpler structure i.e., it's a repetition of basic building blocks such as convolutions, ReLu, max pooling. The biggest advantage of U-Net architecture is that it will work with little amount of training data. In medical field there won't be abundant data available so U-Net is the best fit for this purpose.

The primary disadvantage of U-net is the slow learning rate at the deeper model intermediate layers. In most of the deep learning model the learning rate towards the target data are in zig-zag pattern (learning path), which results in high learning rate i.e., more time for the learning process in deeper layers of the model. So, our project is mainly focused on making that learning path more inclined into the horizontal direction, for this purpose Adam Algorithm is used. Adam Algorithm is also an optimization algorithm built using the best properties of RMSprop and Momentum algorithm. The learning rate will be increased by incorporating the Adam algorithm into the U-net architecture.

Dice Coefficient, loss value, and the amount of time needed for each epoch are the metrics used to assess the success of the Deep Learning model created using the U-Net architecture incorporated by the Adam algorithm.

1.2 Problem Statement

U-Net is the best architecture to build an deep learning model for Image segmentation especially in medical field because there might not be an abundant data available to build the model required, where U-Net architecture requires only a minimum amount of training data. But the problem with the U-Net architecture is that the learning path towards the target value is in zig-zag pattern therefore it makes the learning rate slower in the deeper layers of the model. Our project is focused on making the learning path more inclined in horizontal direction in order to speed up the learning rate of the model.

1.3 Problem Objective

The main objective of the project is to incorporate the Adam Algorithm with the U-Net architecture to speed up the learning rate of the Deep learning model in its deeper layers. Therefore, it can be used in many medical industries without the worry about the minimal data available to them.

1.4 Scope of the Project

1.4.1 Existing System

The image segmentation is mostly done via the machine learning algorithm or with some deep learning model but it always results in a slower learning rate which indeed make them more uncomfortable to use them to solve or automate the real world problems especially in medical field, this problem can be solved using only the high end computers which may result in more capital investment.

1.4.2 Proposed System

To speed up the learning rate of the deep learning model our project is focused on incorporating the Adam Algorithm with the U-Net architecture and making the learning path more inclined in the horizontal direction therefore enhanced speed of the learning rate of the deep learning model especially in the deeper layers.

1.5 SOFTWARE LIFE CYCLE MODEL

The software life cycle is represented pictorially and diagrammatically by a software life cycle model, also known as a process model. A life cycle model depicts every procedure needed to move a software product through each stage of its life cycle. It also captures the organisational framework in which these techniques are to be used. In other words, a life cycle model depicts the different tasks carried out on a piece of software from conception until retirement. The necessary development activities may be scheduled according to phases in various life cycle models. Therefore, no matter whether life cycle model is used, all of the fundamental tasks are included, even if they may be carried out in different sequences depending on the life cycle model, basically it involves 7 stages.

Stage 1: Planning and requirement analysis

The level of the SDLC that is most crucial and essential is requirement analysis. With input from all the stakeholders, domain experts, and SMEs in the industry, the senior team members carry it out. At this point, planning is also done for the requirements for quality assurance and for the identification of project-related risks. A meeting is scheduled with the client by the business analyst and project manager to obtain all the necessary information, such as what the customer wants to construct, who will be the end user, and what the product's goal is. A fundamental knowledge or understanding of the product is crucial before constructing it.

Stage 2: Defining Requirements

The process of representing, documenting, and getting the project stakeholders to approve the software requirements follows the completion of the requirement analysis. This is done by using the "SRS" document, which contains all the product requirements that must be created and developed during the project life cycle.

Stage 3: Designing the Software

The knowledge of the software project's needs, analysis, and design will all be revealed in the upcoming phase. This phase is the result of the previous two, such as requirement collection and client input.

Stage 4: Developing the project

The actual development phase of the SDLC starts here, and programming is created. Coding represents the start of design implementation. Programming tools including compilers, interpreters, debuggers, and other similar tools are used to generate and implement the code, and developers must adhere to the coding standards outlined by their management.

Stage 5: Testing

Following the generation of the code, it is compared to the requirements to ensure that the solutions are satisfying the demands identified and acquired during the requirements stage. Unit testing, integration testing, system testing, and acceptability testing are carried out at this level.

Stage 6: Deployment

When the software has been certified and no defects or mistakes have been reported, it is put into use. The software may then be delivered as is or with proposed improvements in the object portion depending on the assessment. The maintenance of the software starts once it has been deployed.

Stage 7: Maintenance

When the customer begins utilising the technologies that have been designed, the true problems and ongoing needs become apparent. Maintenance is the process when the developed product is given attention.

1.5.1 Types

The common life cycles are

- ➢ Waterfall model
- ➢ Incremental model
- ➢ Spiral model
- ➢ Big bang model
- ➢ Prototyping model
- ➢ RAD model
- ➢ Agile model

1.5.2 Agile model

Agile methodology is a software development approach that emphasizes flexibility, collaboration, and continuous improvement throughout the development process. Unlike the traditional Waterfall model, which follows a linear and sequential approach, the Agile methodology involves working in short iterations and delivering working software at the end of each iteration.

The Agile methodology is based on four core values:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

Overall, the Agile methodology promotes flexibility, collaboration, and continuous improvement, which can result in faster delivery of working software, improved quality, and increased customer satisfaction.

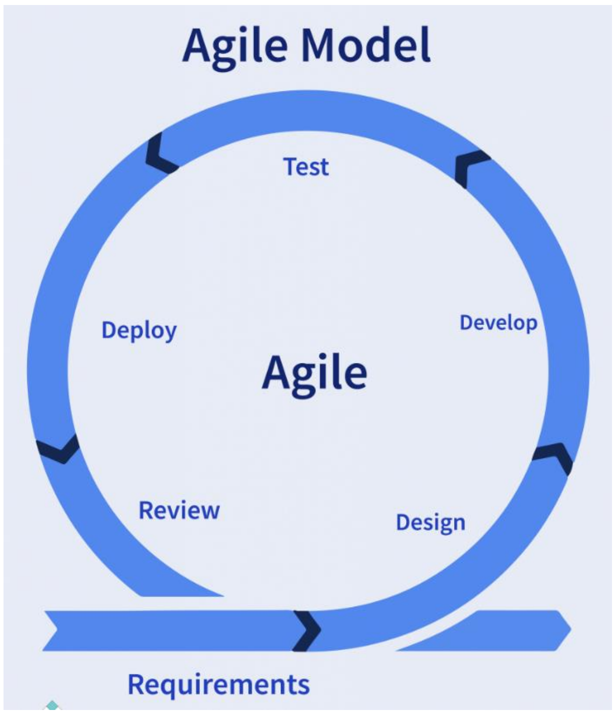


Figure 1.5.1: Agile model

1.5.3 Reason for choosing the Agile model.

Agile methodology is a practice which promotes continues interaction of development and testing during the SDLC process of any project. The entire project is divided into various small incremental builds which are way more convenient for us and all these builds are provided in iteration and each iteration lasts for 10 to 14 days.

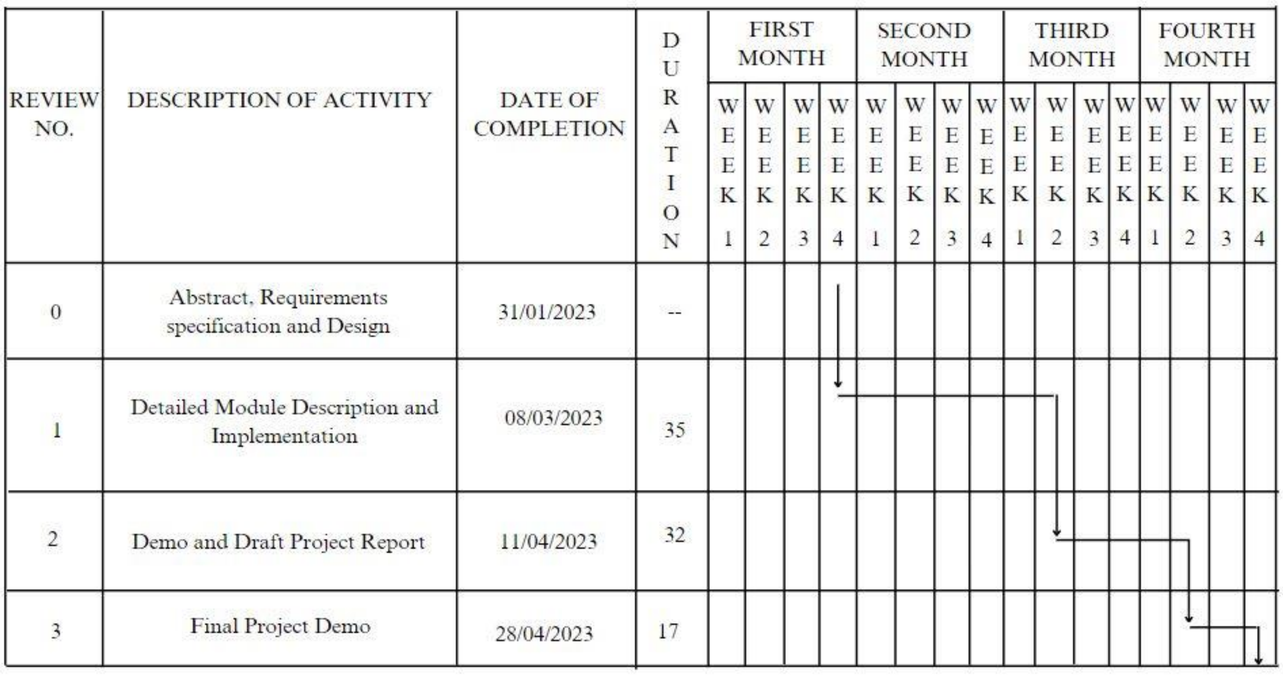
Another reason to use agile model is that it will respond to change over a following plan, It is difficult to think in advance which software requirements will persist and which will change. It is equally difficult to predict how priorities will change as the project proceeds. Analysis, design, development, and testing are not as predictable (from a planning point of view).

While building an software design and development are interleaved i.e., both activities should be performed in tandem so that design models are proven as they are created. It is difficult to think about how much design is necessary before construction is used to test the configuration and more importantly, agile model can accommodate even with fewer developers.

1.6 Project Plan

A project plan for a software project outlines the scope, timeline, budget, resources, and milestones for completing the project. It includes a detailed breakdown of the tasks and activities required to develop and deliver the software. The plan also includes a risk management strategy and quality assurance processes to ensure that the final product meets the client's requirements and specifications.

Table 1.6.1: Project Plan



REVIEW NO.	DESCRIPTION OF ACTIVITY	DATE OF COMPLETION	DURATION	FIRST MONTH				SECOND MONTH				THIRD MONTH				FOURTH MONTH			
				WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4
0	Abstract, Requirements specification and Design	31/01/2023	--																
1	Detailed Module Description and Implementation	08/03/2023	35																
2	Demo and Draft Project Report	11/04/2023	32																
3	Final Project Demo	28/04/2023	17																

1.7 Summary

In this section we have briefly mentioned about the overview of the project, objective and the impact it makes in the medical field and even more than that a clear picture of the project step by step is given from scratch.

We have used the agile model for SDLC (Software Development Life Cycle) as it accommodate fewer developers and can be able to accept many changes and works dynamically and also it requires minimal prior planning.

2. LITERATURE REVIEW

2.1 Introduction CHAPTER 2

In this chapter contains all the literature survey that was the motivation for creating such project and has to make this project into more efficient and effective manner so that a large number of industries can be benefitted from it.

2.2 Literature Review

[1] Kai Han, Lu Liu , Yuqing Song, Yi Liu, Chengjian Qiu, Yangyang Tang, Qiaoying Teng , and Zhe Liu, "An Effective Semi-Supervised Approach for Liver CT Image Segmentation", IEEE Journal of Biomedical and Health informatics, vol. 26, no. 8, August 2022.

Deep networks have made significant progress in the field of medical picture segmentation, but they often need enough pixel-level labelled data for training. Semi-supervised learning is a useful method for reducing reliance on labelled data. To broaden the training dataset, the majority of semi-supervised picture segmentation techniques now in use typically do not produce high-quality pseudo labels. The distances between unlabeled feature vectors and each class representation are then calculated to provide pseudo labels for the unlabeled images. For unlabeled photos in the training process, a novel random patch based on earlier places is also introduced. Thorough testing demonstrates that our method outperforms other semi-supervised algorithms when there are less labelled slices of the LiTS dataset available.

[2] W. Bai et al., "Semi-supervised learning for network-based cardiac MR image segmentation," in Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Interv., 2017..

Training a fully convolutional network for pixel-wise (or voxel-wise) image segmentation normally requires a large number of training images with corresponding ground truth label maps. However, it is a challenge to obtain such a large training set in the medical imaging domain, where expert annotations are time-consuming and difficult to obtain. In this paper, we propose a semi-supervised learning approach, in which a segmentation network is trained from both labelled and unlabelled data. The network parameters and the segmentations for the unlabelled data are alternately updated. We evaluate the method for short-axis cardiac MR image segmentation and it has demonstrated a high performance, outperforming a baseline supervised method. It also outperforms a state-of-the-art multi-atlas segmentation method by a large margin and the speed is substantially faster.

[3] D.-P. Fan et al., "Inf-Net: Automatic COVID-19 lung infection segmentation from CT images," IEEE Trans. Med. Imag., vol. 39, no. 8, Aug. 2020.

Early in 2020, the global spread of Coronavirus Disease 2019 (COVID-19) triggered an existential health crisis. Automatic lung infection diagnosis using computed tomography (CT) images has the potential to significantly improve the current healthcare approach to combat COVID-19. A unique COVID-19 Lung Infection Segmentation Deep Network (Inf-Net) is presented to automatically identify infected regions from chest CT slices in order to overcome these difficulties. In our Multi-Net, the high-level characteristics are combined to create a world map using a parallel partial decoder. Additionally, in order to address the lack of labelled data, we provide a semi-supervised segmentation framework built on a randomly chosen propagation approach. Our framework only needs a small number of labelled photos and mostly uses unlabeled data. Several tests using our COVID-SemiSeg and actual CT volumes show that the suggested Multi-Net surpasses the majority of cutting-edge segmentation models and improves performance.

[4] Y. Zhou et al., "Semi-supervised 3D abdominal multi-organ segmentation via deep multi-planar co-training," in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2019.

In multi-organ segmentation of abdominal CT scans, most existing fully supervised deep learning algorithms require lots of voxel-wise annotations, which are usually difficult, expensive, and slow to obtain. By contrast, deep network based semi-supervised learning methods have not drawn much attention in this field. In this work, we propose (DMPCT), whose contributions can be divided into two folds: 1) The deep model is learned in a co-training style which can mine consensus information from multiple planes; 2) Multi-planar fusion is applied to generate more reliable pseudo-labels, which alleviates the errors occurring in the pseudo-labels and thus can help to train better segmentation networks. Experiments are done on our newly collected large dataset with 100 unlabelled cases, 210 labelled cases where 16 anatomical structures are manually annotated by four radiologists and confirmed by a senior expert. The results suggest that DMPCT significantly outperforms the fully supervised method by more than 4% especially when only a small set of annotations is used.

2.3 Summary

This chapter explains the literature survey that has been carried out in order to make the project more effective.

3. SYSTEM ANALYSIS

CHAPTER 3

3.1 Introduction

System analysis is the process of studying a system to identify its components, their interrelationships, and how they work together to achieve specific goals or objectives. It is the first step in system development, and it involves gathering information about the current system, defining the requirements of the new system, and proposing solutions to any problems or inefficiencies that are identified.

The system analysis process is critical to the success of any system development project. It helps to ensure that the new system meets the needs of the stakeholders, is efficient and effective, and is aligned with the goals and objectives of the organization.

3.2 Requirement Analysis

Requirement analysis is the process of identifying, documenting, and prioritizing the needs and expectations of stakeholders for a software development project. It involves gathering, analysing, and defining the requirements that the software must meet to satisfy stakeholders and achieve the project's goals.

3.2.1 Functional Requirements

1. 3D image is only acceptable and read by niababel library only.
2. Under sampling must be done by making an axial cut along the Z axis.
3. U-Net architecture must be developed with convolution neural network.
4. In-case data’s of other data type given then it must inform it to the user.
5. If the model encounters any error it must inform that to user before termination
6. create_train_data and load_train_data must create and load the training data as required and same must also be implemented for testing also.

3.2.2 Non-Functional Requirements

1. U-Net architecture must be done from 32 channels then 64 channels upto 512 then concatenation of convolution layer will begin from 512 to 32.
2. The total time to perform the operation must be less than 8 seconds.
3. The site must be loaded within 3 seconds.
4. Output must be shown as a gif image within 2 seconds after the operation performed.
5. Intensity must be rescaled while presenting the output.

3.2.3 Hardware Requirements

1. 8GB ram or above
2. Intel or AMD 64bit architecture
3. NVIDIA graphics card
4. OS: windows 10 or 11 or Linux (kernal 3.10 or higher, gilbc 2.17 or higher)

3.2.4 Software Requirements

1. Anaconda Distribution
2. Jupyter notebook
3. VS code (2017 or above)
4. Python (3.8 or above but less than 3.10)
5. Libraries required:
   - i. Sckit Image
   - ii. Keras
   - iii. Os
   - iv. Numpy
   - v. Nibabel
   - vi. tensorflow

3.2.5 Module Specification

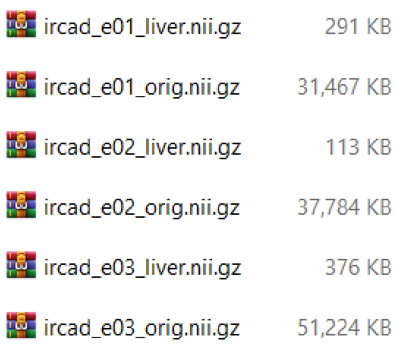
Our project consists of four modules

1. Data Pre-Processing
2. U-Net architecture
3. Adam Algorithm
4. Performance Evaluation

3.2.5.1 Data Pre-Processing

Data collected are the Computed Tomography Scan (CT scan) containing liver part, all the data must be collected in the form of NifTi (3D images). Then the images are read by the nibabel library and then under sampling must be done. The under sampling is done via making an axial cut along the Z-axis and the 2D section containing the liver is recovered.

The recovered section (mask images) will also undergo a testing whether it contains only zeros or not, if so the case then it means that it doesn't contain any liver part then that data must be dropped. At last all the images are saved as .npy (numpy) files for ease of access and fast computation. Total of 70% of data are used for training and reaming 30% is used for testing.



ircad_e01_liver.nii.gz	291 KB
ircad_e01_orig.nii.gz	31,467 KB
ircad_e02_liver.nii.gz	113 KB
ircad_e02_orig.nii.gz	37,784 KB
ircad_e03_liver.nii.gz	376 KB
ircad_e03_orig.nii.gz	51,224 KB

Figure 3.2.1: Dataset containing CT scan

ircad_e01_orig.nii.gz is the CT scan image and ircad_e01_liver.nii.gz is the mask image of the liver

3.2.5.2 U-Net architecture

U-Net comprises of an expanded path and a contracting path. The contracting path adheres to the standard convolutional network architecture. Two 3x3 convolutions (unpadded convolutions) are applied repeatedly, and after each one, a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 are applied for downsampling. We quadruple the number of feature channels with each downsampling step.

An upsampling of the feature map is followed by a 2x2 convolution ("up-convolution") that cuts the number of feature channels in half, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU, at each stage of the expansive path. The cropping is necessary due to the loss of border pixels in every convolution.

At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. The first layer will have 32 channels followed by 64 channels till 512 channels. After 512 channels layers have been made then concatenation of convolution layer will begin i.e. convolution transpose of channels will done from 256 to 32We will be using the activation as “ReLU” means rectified linear activation function ReLU is a piecewise linear function that will output the input directly if is positive otherwise it will output zero.

3.2.5.3 Adam Algorithm

Adam (Adaptive Moment Estimation) is a popular optimization algorithm used in deep learning. It is an extension of stochastic gradient descent (SGD) that computes adaptive learning rates for each parameter based on their past gradients. Adam uses a combination of exponentially decaying moving averages of past gradients and squared gradients to compute the adaptive learning rates and momentum. The algorithm dynamically adjusts the learning rate and momentum based on the first and second moments of the gradient. This approach helps the algorithm converge faster, reduce the variance of the parameter updates, and improve the optimization performance. Adam has become a widely used optimization algorithm in many deep learning applications, and it has demonstrated strong empirical results in practice.

Adam combines the benefits of both RMSprop and momentum by using a combination of moving averages of the gradient and squared gradients to adaptively adjust the learning rate and momentum for each parameter. This helps to reduce the variance of the parameter updates and accelerate the convergence of the optimization process. In summary, Adam uses the adaptive learning rates of RMSprop and the momentum of the past gradients to efficiently optimize the neural network parameters.

Proof of Adam Algorithm combines the best properties of both RMSprop and Momentum

Formula for Momentum,

$$V_{dw} = \beta_1 V_{dw_{prev}} + (1 - \beta_1)\, dw$$

$$V_{dB} = \beta_1 V_{dB_{prev}} + (1 - \beta_1)\, dB$$

$$W = W - \alpha V_{dw}$$

$$B = B - \alpha V_{dB}$$

Formula for RMSProp,

$$S_{dw} = \beta_2 \,.\, S_{dw_{prev}} + (1 - \beta_2)\,(dw)^2$$

$$S_{dB} = \beta_2 \,.\, S_{dB_{prev}} + (1 - \beta_2)\,(dB)^2$$

$$W = W - \alpha \left(dW \Big/ \sqrt{S_{dW} + \varepsilon} \right)$$

$$B = B - \alpha \left(dB \Big/ \sqrt{S_{dB} + \varepsilon} \right)$$

The formula for Adam Algorithm is generated by combining the numerator of the Momentum formula (V_{dw}) and denominator of the RMSProp ($\sqrt{S_{dW} + \varepsilon}$)

Formula for Adam Algorithm

$$W = W - \alpha \,.\, \frac{V_{dw}}{\sqrt{S_{dW} + \varepsilon}}$$

$$B = B - \alpha \,.\, \frac{V_{dB}}{\sqrt{S_{dB} + \varepsilon}}$$

Step 1: Write the formula for weight via a coding language.

$$W = W - \alpha \,.\, \frac{V_{dw}}{\sqrt{S_{dW} + \varepsilon}}$$

Step 2: Then do the same for the Bias

$$B = B - \alpha \,.\, \frac{V_{dB}}{\sqrt{S_{dB} + \varepsilon}}$$

Step 3: Now assign value for beta1 as 0.9, beta2 as 0.999, epsilon as 1e-8 and lastly to eta as 0.01

$$\beta_1 = 0\,.\,9$$

$$\beta_2 = 0\,.\,999$$

$$\epsilon = 10^{-8}$$

$$eta = 0.01$$

Step 4: Now find the dw and db for updation.

m_dw = beta1*m_dw + (1-beta1)*dw

m_db = beta1*m_db + (1-beta1)*db

Step 5: Now do the same for beta2.

Step 6: Now find the bias correction

m_dw_corr = self.m_dw/(1-self.beta1**t)

m_db_corr = self.m_db/(1-self.beta1**t)

v_dw_corr = self.v_dw/(1-self.beta2**t)

v_db_corr = self.v_db/(1-self.beta2**t) , Where t is the target value

Step 7: Then update the weight and bias accordingly.

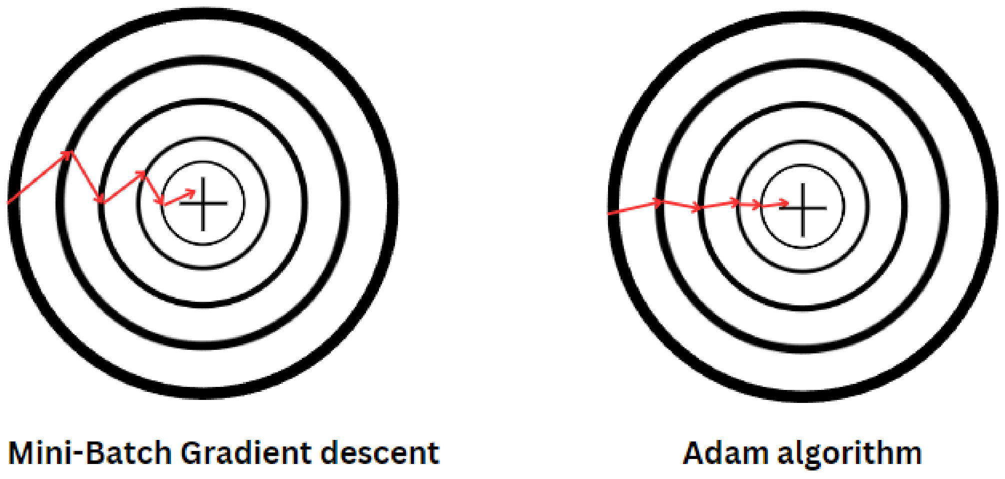


Figure 3.2.2: Comparison of learning path between Gradient Descent and Adam Algorithm

3.2.5.4 Performance Evaluation

The Dice coefficient is a commonly used similarity metric in image segmentation tasks. It measures the overlap between the predicted segmentation mask and the ground truth segmentation mask of an image. It is calculated as twice the intersection of the two masks divided by the sum of the sizes of the two masks. The Dice coefficient ranges from 0 to 1, where a value of 1 indicates perfect overlap between the predicted and ground truth segmentation masks.

The formula of dice coefficient are :

$$\frac{2*|X \cap Y|}{|X|+|Y|}$$

3.3 Summary

This chapter explains details of system analysis, function and non functional requirements along with hardware and software requirements. Apart from that a clear picture about each modules is explained clearly with the certain proof required for it.

4. SYSTEM DESIGN

4.1 Introduction CHAPTER 4

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specific requirements. It involves analysing the requirements, determining the system's objectives and constraints, and developing a high-level architecture for the system. System design includes defining the components that make up the system and the interactions between them. It also involves defining the data structures and algorithms used to process the data.

The system design phase is critical as it sets the foundation for the development, testing, and deployment of the system. It ensures that the system is designed to meet the desired requirements and that the architecture is flexible and scalable to accommodate future changes and enhancements.

4.2 Data Flow Diagram

Data Flow Diagram (DFD) is a graphical representation of how data flows through a system, showing the inputs, outputs, and processes involved. It is commonly used in software engineering and business analysis.

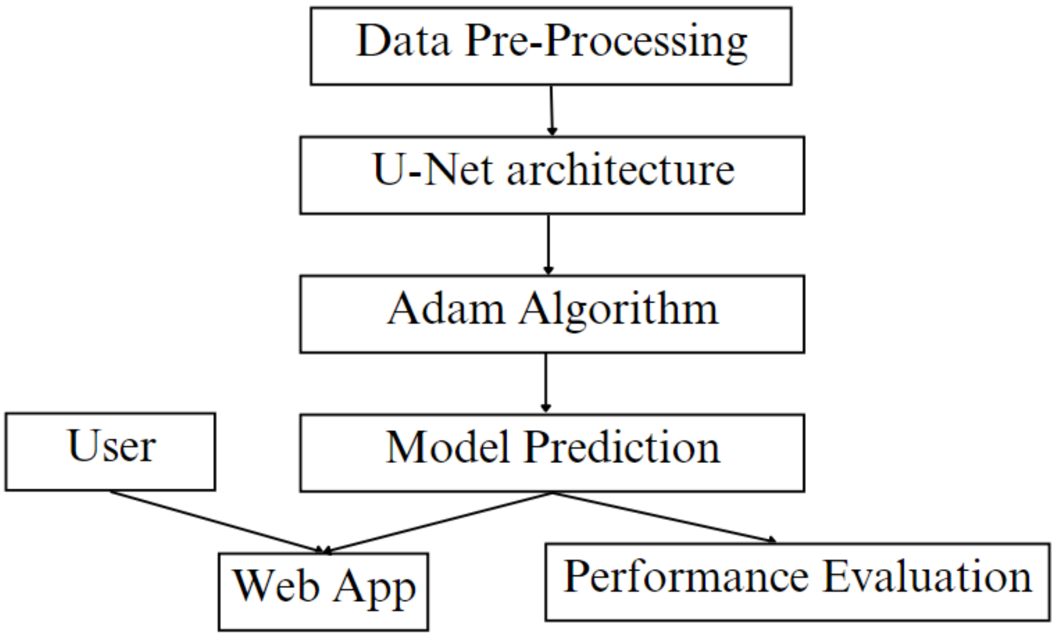


Figure 4.2.1: Data Flow Diagram

4.3 Object Oriented Analysis and Design

Object Oriented Analysis and Design (OOAD) is a software engineering approach that involves modelling a system as a collection of objects with their attributes and behaviours. It focuses on developing reusable and modular software systems.

4.3.1 Detailed Design

Detailed Design in UML is the process of creating a detailed blueprint for the software system based on the system design. It includes creating detailed class diagrams, sequence diagrams, state machine diagrams, and other UML diagrams to describe the behaviour and structure of the system. The detailed design phase also involves designing the user interface, database schema, and other technical details of the system. The output of this phase is a detailed design document that serves as a guide for the development team. Detailed design is crucial for developing high-quality, maintainable, and scalable software systems.

4.3.1.1 Use Case Diagram

A use case diagram is a type of UML diagram that provides a visual representation of the interactions between actors and the system. It is used to depict the system's functions, their relationships, and how they are related to actors who interact with the system. Use case diagrams are commonly used in software engineering to model and design systems.

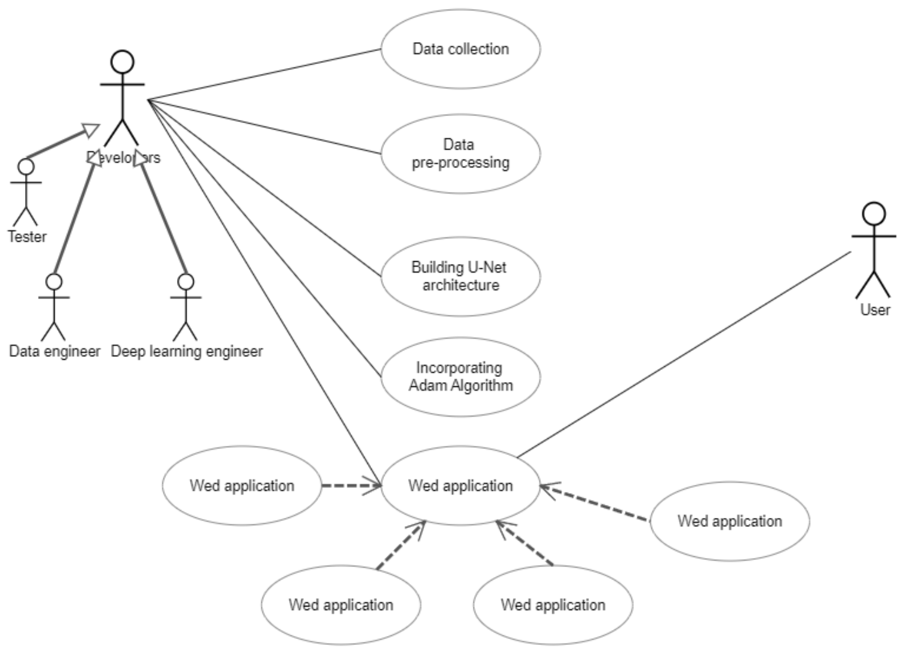


Figure 4.3.1: Use Case diagram

4.3.1.2 Activity Diagram

An activity diagram is a type of UML diagram that depicts the flow of activities or processes in a system. It is used to model business processes, software processes, and workflows. Activity diagrams show the activities, decisions, and branching paths involved in a process, making it easier to understand and analyse the system's behavior.

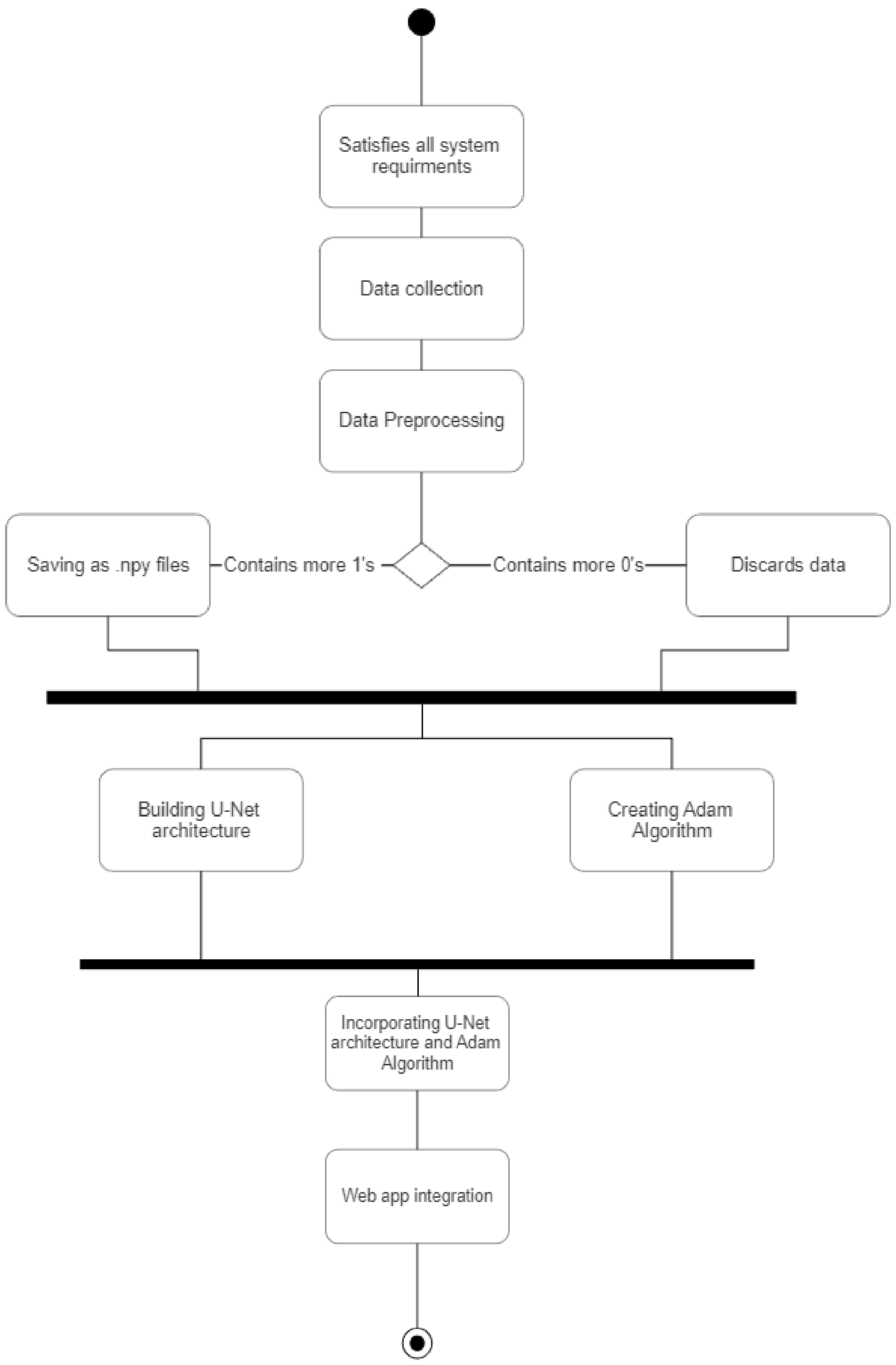


Figure 4.3.2: Activity diagram

4.3.1.3 Class Diagram

A class diagram is a type of UML diagram that describes the structure of a system by showing the classes, their attributes, methods, and relationships with other classes. It is a static representation that is commonly used in software engineering to model and design object-oriented systems.

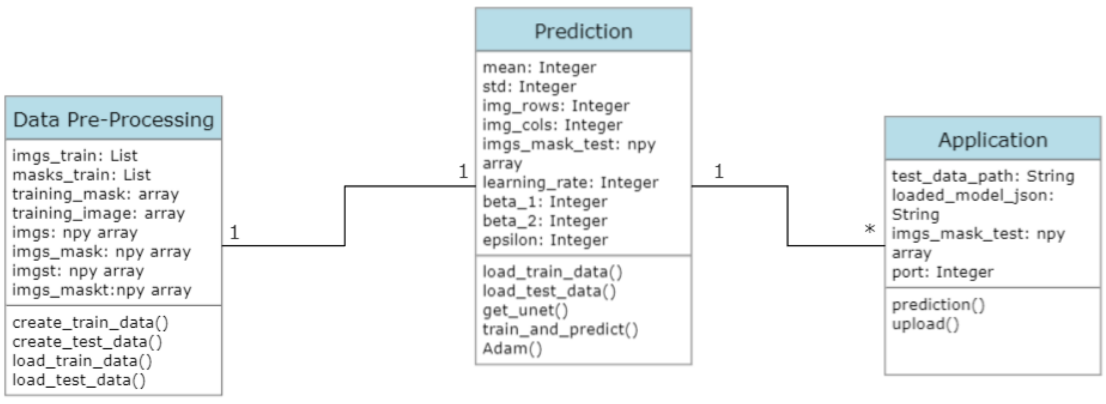


Figure 4.3.3: Class diagram

4.3.1.4 Sequence Diagram

A sequence diagram is a type of UML diagram that shows the interactions between objects or actors in a system, including the order in which messages are sent and received. It depicts the sequence of events in a system, making it easier to understand and analyse the system's behavior. Sequence diagrams are commonly used in software engineering to model and design systems that involve multiple objects and actors.

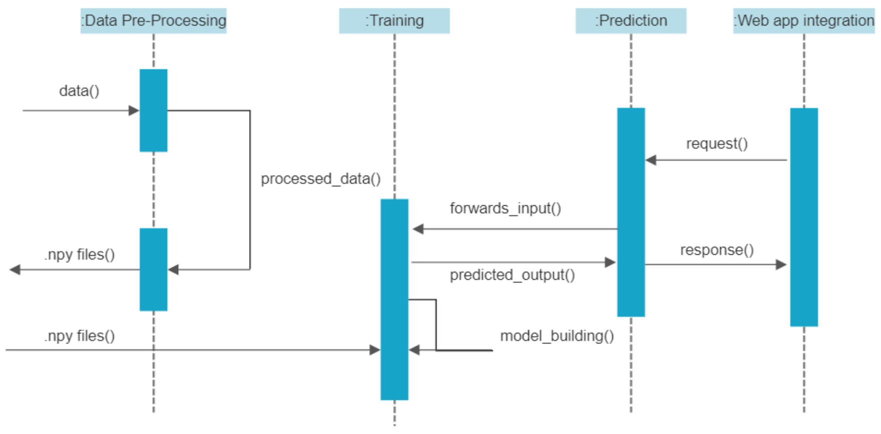


Figure 4.3.4: Sequence diagram

4.3.1.5 Communication Diagram

A communication diagram is a type of UML diagram that depicts the interactions between objects or actors in a system. It shows the flow of messages or information between objects, along with their lifelines and activation bars. Communication diagrams are commonly used in software engineering to model and design complex systems with multiple objects and actors.

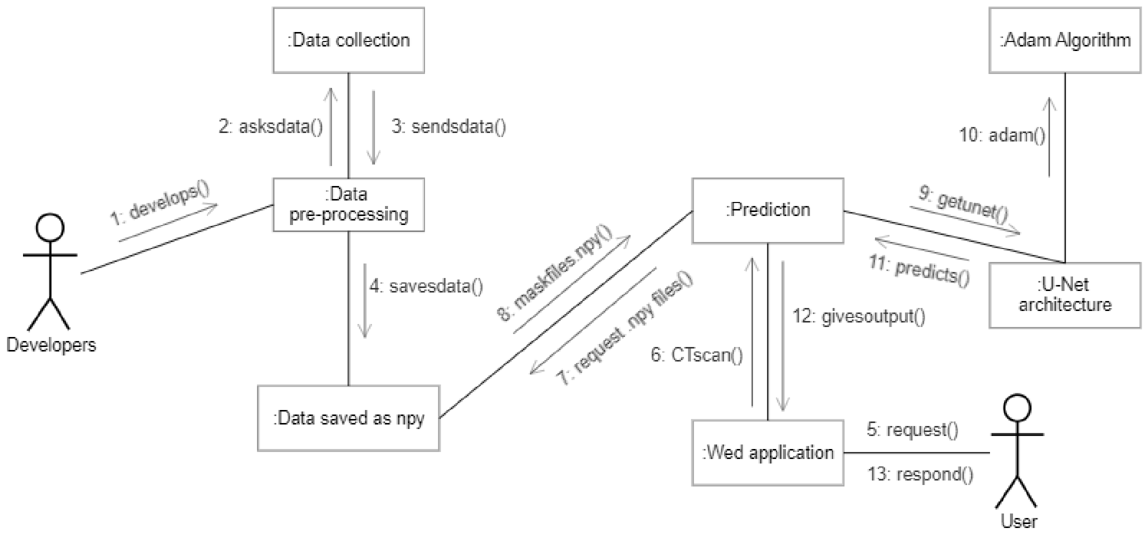


Figure 4.3.5: Communication diagram

4.4 Summary

This chapter explains in depth about the overall system design underlying the complete project and even explains all the scenarios and the system will react to it.

5. IMPLEMENTATION

5.1 data.py

CHAPTER 5

```
import os
import numpy as np

import nibabel

data_path = 'raw/'

image_rows = int(512/2)

image_cols = int(512/2)

def create_train_data():

    print('-'*30)

    print('Creating training data...')

    print('-'*30)

    train_data_path = os.path.join(data_path, 'train')

    images = os.listdir(train_data_path)

    imgs_train = []

    masks_train = []

    training_masks = images[::2]

    training_images = images[1::2]

    for liver, orig in zip(training_masks, training_images):

        training_mask = nibabel.load(os.path.join(train_data_path, liver))

        training_image = nibabel.load(os.path.join(train_data_path, orig))

        for k in range(training_mask.shape[2]):

            mask_2d = np.array(training_mask.get_data()[::2, ::2, k])

            image_2d = np.array(training_image.get_data()[::2, ::2, k])

            if len(np.unique(mask_2d)) != 1:

                masks_train.append(mask_2d)

                imgs_train.append(image_2d)
    imgs = np.ndarray(
            (len(imgs_train), image_rows, image_cols), dtype=np.uint8
            )
    imgs_mask = np.ndarray(
            (len(masks_train), image_rows, image_cols), dtype=np.uint8
            )
    for index, img in enumerate(imgs_train):
        imgs[index, :, :] = img
    for index, img in enumerate(masks_train):
        imgs_mask[index, :, :] = img

    np.save('imgs_train.npy', imgs)
    np.save('masks_train.npy', imgs_mask)
    print('Saving to .npy files done.')

def load_train_data():
    imgs_train = np.load('imgs_train.npy')
    masks_train = np.load('masks_train.npy')
    return imgs_train, masks_train

def create_test_data():
    print('-'*30)
    print('Creating test data...')
    print('-'*30)
```

```
test_data_path = os.path.join(data_path, 'test')
images = os.listdir(test_data_path)
imgs_test = []
masks_test = []
for image_name in images:
    print(image_name)
    img = nibabel.load(os.path.join(test_data_path, image_name))
    print(img.shape)
    for k in range(img.shape[2]):
        img_2d = np.array(img.get_data()[::2, ::2, k])
        if 'liver' in image_name:
            masks_test.append(img_2d)
        elif 'orig' in image_name:
            imgs_test.append(img_2d)
imgst = np.ndarray(
        (len(imgs_test), image_rows, image_cols), dtype=np.uint8
        )
imgs_maskt = np.ndarray(
        (len(masks_test), image_rows, image_cols), dtype=np.uint8
        )
for index, img in enumerate(imgs_test):
    imgst[index, :, :] = img
for index, img in enumerate(masks_test):
    imgs_maskt[index, :, :] = img
    np.save('imgs_test.npy', imgst)
    np.save('masks_test.npy', imgs_maskt)
    print('Saving to .npy files done.')


def load_test_data():
    imgs_test = np.load('imgs_test.npy')
    masks_test = np.load('masks_test.npy')
    return imgs_test, masks_test
if __name__ == '__main__':
    create_train_data()
    create_test_data()
```

5.2 train.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 24 22:01:31 2023
@author: vkedu
"""
from __future__ import print_function
import os
from skimage.transform import resize
import numpy as np
from skimage.segmentation import mark_boundaries
from keras.models import Model
from keras.layers import Input, concatenate, Conv2D, MaxPooling2D, Conv2DTranspose
from keras.callbacks import ModelCheckpoint

from keras.optimizers import Adam, SGD

from keras import backend as K

from skimage.exposure import rescale_intensity

from skimage import io

from data import load_train_data, load_test_data


K.set_image_data_format('channels_last')


img_rows = int(512/2)

img_cols = int(512/2)

smooth = 1.


def dice_coef(y_true, y_pred):

    y_true_f = K.flatten(y_true)

    y_pred_f = K.flatten(y_pred)

    intersection = K.sum(y_true_f * y_pred_f)

    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)


def dice_coef_loss(y_true, y_pred):

    return -dice_coef(y_true, y_pred)


def get_unet():

    inputs = Input((img_rows, img_cols, 1))
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

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

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

    up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5),
conv4], axis=3)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)
    up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6),
conv3], axis=3)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)

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

    up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(conv8),
conv1], axis=3)
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)
    conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)
    model = Model(inputs=[inputs], outputs=[conv10])
    model.compile(optimizer=Adam(lr=1e-3), loss=dice_coef_loss, metrics=[dice_coef])
    return model

def preprocess(imgs):
    imgs_p = np.ndarray((imgs.shape[0], img_rows, img_cols), dtype=np.uint8)
    for i in range(imgs.shape[0]):
        imgs_p[i] = resize(imgs[i], (img_cols, img_rows), preserve_range=True)

    imgs_p = imgs_p[..., np.newaxis]
    return imgs_p
import matplotlib.pyplot as plt
def train_and_predict():
    print('-'*30)
    print('Loading and preprocessing train data...')
    print('-'*30)
    imgs_train, imgs_mask_train = load_train_data()
    imgs_train = preprocess(imgs_train)
    imgs_mask_train = preprocess(imgs_mask_train)
    imgs_train = imgs_train.astype('float32')
    mean = np.mean(imgs_train)
    std = np.std(imgs_train)
    imgs_train -= mean
    imgs_train /= std
    imgs_mask_train = imgs_mask_train.astype('float32')
    print('Creating and compiling model...')
    model = get_unet()
    model_checkpoint = ModelCheckpoint('weights.h5', monitor='val_loss',
save_best_only=True)
    print('Fitting model...')
    history=model.fit(imgs_train, imgs_mask_train, batch_size=10, epochs=50, verbose=1,
shuffle=True,
              validation_split=0.2,
              callbacks=[model_checkpoint])
```

```
print('-'*30)

print('Loading and preprocessing test data...')

print('-'*30)

imgs_test, imgs_maskt = load_test_data()

imgs_test = preprocess(imgs_test)

imgs_test = imgs_test.astype('float32')

imgs_test -= mean

imgs_test /= std

print('-'*30)

print('Loading saved weights...')

print('-'*30)

model.load_weights('weights.h5')

print('-'*30)

print('Predicting masks on test data...')

print('-'*30)

imgs_mask_test = model.predict(imgs_test, verbose=1)

np.save('imgs_mask_test.npy', imgs_mask_test)

print('-' * 30)

print('Saving predicted masks to files...')

print('-' * 30)

pred_dir = 'preds'

if not os.path.exists(pred_dir):

    os.mkdir(pred_dir)
    for k in range(len(imgs_mask_test)):
        a=rescale_intensity(imgs_test[k][:,:,0],out_range=(-1,1))
        b=(imgs_mask_test[k][:,:,0]).astype('uint8')
        io.imsave(os.path.join(pred_dir, str(k) + '_pred.png'),mark_boundaries(a,b))

    plt.plot(history.history['dice_coef'])
    plt.plot(history.history['val_dice_coef'])
    plt.title('Model dice coeff')
    plt.ylabel('Dice coeff')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
if __name__ == '__main__':
    train_and_predict()
```

5.3 adam.py

```
import tensorflow.compat.v2 as tf
from keras import backend_config
from keras.optimizers.optimizer_v2 import optimizer_v2
# isort: off
from tensorflow.python.util.tf_export import keras_export
@keras_export(
    "keras.optimizers.legacy.Adam",
    v1=["keras.optimizers.Adam", "keras.optimizers.legacy.Adam"],
)
class Adam(optimizer_v2.OptimizerV2):
    _HAS_AGGREGATE_GRAD = True
    def __init__(
        self,
        learning_rate=0.001,
        beta_1=0.9,
        beta_2=0.999,
        epsilon=1e-7,
        amsgrad=False,
        name="Adam",
        **kwargs
    ):
        super().__init__(name, **kwargs)
        self._set_hyper("learning_rate", kwargs.get("lr", learning_rate))
        self._set_hyper("decay", self._initial_decay)
        self._set_hyper("beta_1", beta_1)
        self._set_hyper("beta_2", beta_2)
        self.epsilon = epsilon or backend_config.epsilon()
        self.amsgrad = amsgrad
    def _create_slots(self, var_list):
        # Create slots for the first and second moments.
        # Separate for-loops to respect the ordering of slot variables from v1.
        for var in var_list:
            self.add_slot(var, "m")
        for var in var_list:
            self.add_slot(var, "v")
        if self.amsgrad:
            for var in var_list:
                self.add_slot(var, "vhat")
                beta_2_t=beta_2_t,
                beta_2_power=beta_2_power,
                one_minus_beta_2_t=1 - beta_2_t,
            )
        )
    def set_weights(self, weights):
        params = self.weights
        num_vars = int((len(params) - 1) / 2)
        if len(weights) == 3 * num_vars + 1:
            weights = weights[: len(params)]
        super().set_weights(weights)
                beta1_power=coefficients["beta_1_power"],
                beta2_power=coefficients["beta_2_power"],
                lr=coefficients["lr_t"],
                beta1=coefficients["beta_1_t"],
                beta2=coefficients["beta_2_t"],
                epsilon=coefficients["epsilon"],
                grad=grad,
            use_locking=self._use_locking,
        )
    else:
        vhat = self.get_slot(var, "vhat")
        return tf.raw_ops.ResourceApplyAdamWithAmsgrad(
            var=var.handle,
            m=m.handle,
            v=v.handle,
            vhat=vhat.handle,
            beta1_power=coefficients["beta_1_power"],
            beta2_power=coefficients["beta_2_power"],
            lr=coefficients["lr_t"],
            beta1=coefficients["beta_1_t"],
            beta2=coefficients["beta_2_t"],
            epsilon=coefficients["epsilon"],
            grad=grad,
            use_locking=self._use_locking,
        )
def _resource_apply_sparse(self, grad, var, indices, apply_state=None):
    var_device, var_dtype = var.device, var.dtype.base_dtype
    coefficients = (apply_state or {}).get(
        (var_device, var_dtype)
    ) or self._fallback_apply_state(var_device, var_dtype)
```

```
# m_t = beta1 * m + (1 - beta1) * g_t
m = self.get_slot(var, "m")
m_scaled_g_values = grad * coefficients["one_minus_beta_1_t"]
m_t = tf.compat.v1.assign(
    m, m * coefficients["beta_1_t"], use_locking=self._use_locking
)
with tf.control_dependencies([m_t]):
    m_t = self._resource_scatter_add(m, indices, m_scaled_g_values)

# v_t = beta2 * v + (1 - beta2) * (g_t * g_t)
v = self.get_slot(var, "v")
v_scaled_g_values = (grad * grad) * coefficients["one_minus_beta_2_t"]
v_t = tf.compat.v1.assign(
    v, v * coefficients["beta_2_t"], use_locking=self._use_locking
)
with tf.control_dependencies([v_t]):
    v_t = self._resource_scatter_add(v, indices, v_scaled_g_values)

if not self.amsgrad:
    v_sqrt = tf.sqrt(v_t)
    var_update = tf.compat.v1.assign_sub(
        var,
        coefficients["lr"] * m_t / (v_sqrt + coefficients["epsilon"]),
        use_locking=self._use_locking,
            )
            return tf.group(*[var_update, m_t, v_t])
        else:
            v_hat = self.get_slot(var, "vhat")
            v_hat_t = tf.maximum(v_hat, v_t)
            with tf.control_dependencies([v_hat_t]):
                v_hat_t = tf.compat.v1.assign(
                    v_hat, v_hat_t, use_locking=self._use_locking
                )
            v_hat_sqrt = tf.sqrt(v_hat_t)
            var_update = tf.compat.v1.assign_sub(
                var,
                coefficients["lr"]
                * m_t
                / (v_hat_sqrt + coefficients["epsilon"]),
                use_locking=self._use_locking,
            )
            return tf.group(*[var_update, m_t, v_t, v_hat_t])

    def get_config(self):
        config = super().get_config()
        config.update(
            {
                "learning_rate": self._serialize_hyperparameter(
                "learning_rate"
            ),
            "decay": self._initial_decay,
            "beta_1": self._serialize_hyperparameter("beta_1"),
            "beta_2": self._serialize_hyperparameter("beta_2"),
            "epsilon": self.epsilon,
            "amsgrad": self.amsgrad,
          }
        )
        return config
class NonFusedAdam(optimizer_v2.OptimizerV2):
    _HAS_AGGREGATE_GRAD = True
    def __init__(
        self,
        learning_rate=0.001,
        beta_1=0.9,
        beta_2=0.999,
        epsilon=1e-7,
        amsgrad=False,
        name="Adam",
        **kwargs
    ):
        super().__init__(name, **kwargs)
        self._set_hyper("learning_rate", kwargs.get("lr", learning_rate))
        self._set_hyper("decay", self._initial_decay)
        self._set_hyper("beta_1", beta_1)
        self._set_hyper("beta_2", beta_2)
        self.epsilon = epsilon or backend_config.epsilon()
        self.amsgrad = amsgrad

    def _create_slots(self, var_list):
        for var in var_list:
            self.add_slot(var, "m")
        for var in var_list:
            self.add_slot(var, "v")
        if self.amsgrad:
            for var in var_list:
                self.add_slot(var, "vhat")

    def _prepare_local(self, var_device, var_dtype, apply_state):
        super()._prepare_local(var_device, var_dtype, apply_state)
        local_step = tf.cast(self.iterations + 1, var_dtype)
        beta_1_t = tf.identity(self._get_hyper("beta_1", var_dtype))
        beta_2_t = tf.identity(self._get_hyper("beta_2", var_dtype))
        beta_1_power = tf.pow(beta_1_t, local_step)
        beta_2_power = tf.pow(beta_2_t, local_step)
        lr = apply_state[(var_device, var_dtype)]["lr_t"] * (
            tf.sqrt(1 - beta_2_power) / (1 - beta_1_power)
    )
    apply_state[(var_device, var_dtype)].update(
        dict(
            lr=lr,
            epsilon=tf.convert_to_tensor(self.epsilon, var_dtype),
            beta_1_t=beta_1_t,
            beta_1_power=beta_1_power,
            one_minus_beta_1_t=1 - beta_1_t,
            beta_2_t=beta_2_t,
            beta_2_power=beta_2_power,
            one_minus_beta_2_t=1 - beta_2_t,
        )
    )
def set_weights(self, weights):
    params = self.weights
    num_vars = int((len(params) - 1) / 2)
    if len(weights) == 3 * num_vars + 1:
        weights = weights[: len(params)]
    super().set_weights(weights)


@tf.function(jit_compile=True)
def _resource_apply_dense(self, grad, var, apply_state=None):
    var_device, var_dtype = var.device, var.dtype.base_dtype
    coefficients = (apply_state or {}).get(
        (var_device, var_dtype)
    ) or self._fallback_apply_state(var_device, var_dtype)

    m = self.get_slot(var, "m")
    v = self.get_slot(var, "v")

    alpha = (
        coefficients["lr_t"]
        * tf.sqrt(1 - coefficients["beta_2_power"])
        / (1 - coefficients["beta_1_power"])
    )
    m.assign_add((grad - m) * (1 - coefficients["beta_1_t"]))
    v.assign_add((tf.square(grad) - v) * (1 - coefficients["beta_2_t"]))
    if self.amsgrad:
        vhat = self.get_slot(var, "vhat")
        vhat.assign(tf.maximum(vhat, v))
        v = vhat
    var.assign_sub((m * alpha) / (tf.sqrt(v) - coefficients["epsilon"]))


@tf.function(jit_compile=True)
def _resource_apply_sparse(self, grad, var, indices, apply_state=None):
    var_device, var_dtype = var.device, var.dtype.base_dtype
    coefficients = (apply_state or {}).get(
        (var_device, var_dtype)
```

```
) or self._fallback_apply_state(var_device, var_dtype)
# m_t = beta1 * m + (1 - beta1) * g_t
m = self.get_slot(var, "m")
m_scaled_g_values = grad * coefficients["one_minus_beta_1_t"]
m.assign(m * coefficients["beta_1_t"])
m.scatter_add(tf.IndexedSlices(m_scaled_g_values, indices))
# v_t = beta2 * v + (1 - beta2) * (g_t * g_t)
v = self.get_slot(var, "v")
v_scaled_g_values = (grad * grad) * coefficients["one_minus_beta_2_t"]
v.assign(v * coefficients["beta_2_t"])
v.scatter_add(tf.IndexedSlices(v_scaled_g_values, indices))

if not self.amsgrad:
    var.assign_sub(
        coefficients["lr"] * m / (tf.sqrt(v) + coefficients["epsilon"])
    )
else:
    v_hat = self.get_slot(var, "vhat")
    v_hat.assign(tf.maximum(v_hat, v))
    var.assign_sub(
        coefficients["lr"]
        * m
        / (tf.sqrt(v_hat) + coefficients["epsilon"])
    )
    def get_config(self):
        config = super().get_config()
        config.update(
            {
                "learning_rate": self._serialize_hyperparameter(
                    "learning_rate"
                ),
                "decay": self._initial_decay,
                "beta_1": self._serialize_hyperparameter("beta_1"),
                "beta_2": self._serialize_hyperparameter("beta_2"),
                "epsilon": self.epsilon,
                "amsgrad": self.amsgrad,
            }
        )
        return config
```

5.4 app.py

```
from flask import Flask, render_template, request
import numpy as np
import os
from skimage.segmentation import mark_boundaries
from skimage.exposure import rescale_intensity
from skimage import io
```

```
import nibabel
from tensorflow.keras.models import model_from_json
import imageio
app = Flask('_name_',template_folder='templates')
def prediction(test_data_path):
    pred_dir = '__pycache__/predsss'
    if not os.path.exists(pred_dir):
        os.mkdir(pred_dir)
    image_rows = int(512/2)
    image_cols = int(512/2)
    imgs_test = []
    img = nibabel.load(test_data_path)
    for k in range(img.shape[2]):
            img_2d = np.array(img.get_data()[::2, ::2, k])
            imgs_test.append(img_2d)
    imgst = np.ndarray(
            (len(imgs_test), image_rows, image_cols), dtype=np.uint8
            )
    for index, img in enumerate(imgs_test):
        imgst[index, :, :] = img
    np.save('test_1.npy', imgst)
    imgs_test = np.load('test_1.npy')
    json_file = open('model.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
    loaded_model.load_weights("weights.h5")
    imgs_mask_test = loaded_model.predict(imgs_test, verbose=1)
    np.save('test_2.npy', imgs_mask_test)
    for k in range(len(imgs_mask_test)):
        a=rescale_intensity(imgs_test[k][:,:],out_range=(-1,1))
        b=(imgs_mask_test[k][:,:,0]).astype('uint8')
        io.imsave(os.path.join(pred_dir, str(k) + '_pred.png'),mark_boundaries(a,b))
@app.route('/')
def index():
    return render_template('main.html')
@app.route('/result', methods=['GET', 'POST'])
def upload():
    f = request.files['file']
    f.save(f.filename)
    j=str(f.filename)
    test_data_path=j
    b=test_data_path.split('.')
    l=int(b[0])
    images = []
    d={1:[0,113],2:[113,238],3:[393,512]}
    for i in range(d[l][0],d[l][1]):
images.append(imageio.imread(os.path.join("__pycache__/preds",str(l),str(i)+"_pred.png")))
```

```
    imageio.mimsave('static/result.gif', images)
    imageio.mimsave('result/result.gif', images)
    print("All process completed and result is saved in result folder in your local PC")
    return render_template("preview.html")
port = int(os.environ.get('PORT', 5000))
app.run(host='0.0.0.0', port=port,debug=True)
```

5.5 main.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>3D-IMAGE OPTMIZATION</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="{{url_for('static',filename='styles.css')}}">
  <!-- <link rel="stylesheet" href="styles.css"> -->
</head>
<body>
  <div class="container-fluid">
```

```
    <div class="container align-items-center section">
      <div class="title text-light">3D IMAGE OPTMIZATION USING ADAM
ALAGORITHM</div>
      <div class="d-flex btn-section">
        <form action = "http://localhost:5000/result" method = "POST" enctype =
"multipart/form-data">
         <centre><br><input type = "file" name = "file" /></centre>
         <centre><input type = "submit" style="color:white;background-
color:#0e64ed;height:60px; width:100px;border: none"/></centre>
        <!-- <input type="file" id="myfile" class="w-100"> -->
        <!-- <button class="btn w-50"><a href="/preview.html" class="text-decoration-none
text-light">Upload</a></button> -->
      </div>
    </div>
  </div>
  <div class="img-container container p-5 my-5">
    <h2 class="text-center text-light">Project Details</h2>
    <div class="d-flex">
      <img src="{{url_for('static',filename='IMG1.jpeg')}}" alt="Deep Learning Image col-3">
      <p class="text-left text-light border-left-dashed p-4">
        Automated computing in medical field can be one of the greatest achievements of the
Man Kind,
        our project is focused on liver segmentation from CT scan. We will be using a U-Net
architecture to preform the liver segmentation.
        Firstly, the CT scan (3D) is converted into a 2D and then into a NumPy file for ease of
access and usage. Then these files are sent to
```

```
        the U-Net architecture to complete the liver segmentation process. The U-Net
architecture is built using the convolution neural network.

        It has one drawback i.e., is that its learning rate is slow, therefore, to increase the
learning rate we are using the Adam algorithm to

        make the learning path more inclined into horizontal direction therefore the learning rate
will be fast. Then the output images are combined

        into forming a gif image that will be displayed as an output.

      </p>

    </div>

  </div>

  <div class="footer">

     <div class="logo text-center">

      <div class="pt-3"><a
href="https://www.irjmets.com/paperdetail.php?paperId=d871897562f928adb635bb6b61fb0
0ed&title3D+IMAGE+OPTMIZATION+USING+ADAM+ALAGORITHM&authpr=J.S.Ka
nchana" target="_blank" class="text-decoration-none text-light"> Published
Journal</a></div>

      <i class="fa fa-graduation-cap bg-white text-dark p-2 m-2 mt-3" aria-hidden="true"></i>

      <i class="fa fa-graduation-cap bg-white text-dark p-2 m-2 mt-3" aria-hidden="true"></i>

      <i class="fa fa-graduation-cap bg-white text-dark p-2 m-2 mt-3" aria-hidden="true"></i>

     </div>

     <div class="info-section d-flex contaier justify-content-center">

      <div class="m-2 text-light">J.S.Kanchana</div>

        <div class="m-2 text-light">Venkatesh T</div>

        <div class="m-2 text-light">Vinod T.H</div>

     </div>
  </div>
</body>
</html>
```

5.6 preview.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image-Preview</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
  <link rel="stylesheet" href="{{url_for('static',filename='styles.css')}}">
</head>
<body>
  <div class="container text-center">
    <h3 class="m-5">Result</h3>
    <img src="{{url_for('static',filename='result.gif')}}" alt="image" class="result-image">
  </div>
</body>
</html>
```

5.6 style.css

```
body {margin: 0;
  background-color: #1b1e22;
  font-family: serif;
  color: white;}
.section {
  width: 800px;
  /* height: 1000px; */
  padding-top: 150px;
  text-align: center;}
.section .title {
  font-size: 50px;}
input[type='file'] {
  color: rgba(0, 0, 0, 0)}
input::file-selector-button {
  background-color: #1b1e22;
  background-size: 200%;
  border: 1px solid white;
  color: #fff;
  padding: 18px 30px;}
.btn {
  background-color: blue;
  padding: 1rem 1.25rem;
  border-radius: 0;}
.btn:hover {
  border: 1px solid white;}
.btn-section {
  padding: 30px 200px;}
.img-container img, .main-image img{
  border-radius: 30px;}
.img-container img {
  width: 500px;}
.border-left-dashed {
  padding: 0 60px;}
.footer {
  height: 150px;
  background-color: #15191f7a;}
.footer .logo i{
  border-radius: 50%;}
.result-image {
  border: 2px solid white;
  padding: 20px;
  border-radius: 10px;
  height: 700px;
  width: 1200px;}
```

6.Testing

6.1 Introduction

CHAPTER 6

This chapter gives a clear picture about the various testing performed for our project. Testing is one of the crucial part for a software development because it is the only way to find whether the project meets all the requirements without any error and checks that it can also be used out of the environment also.

6.2 Software Testing

Software testing is a process of evaluating a software system or application to detect any defects or errors in the system. It involves validating that the system meets its functional and non-functional requirements and performs as expected. The goal of software testing is to identify and eliminate any issues in the system before it is released to end-users. Testing is an essential part of the software development lifecycle and ensures the quality and reliability of the software.

The main objective of software testing are:

1. To ensure that the software meets its functional and non-functional requirements.
2. To detect and fix defects or errors in the software before it is released to end-users.
3. To improve the quality and reliability of the software.
4. To enhance the user experience by ensuring the software is easy to use and navigate.
5. To increase the efficiency of the software by identifying and removing any performance bottlenecks.
6. To ensure the software is secure and not vulnerable to attacks.
7. To reduce the overall cost of software development by identifying issues early in the development process.

6.3 Basic Types of Testing

There are several types of software testing, including:

1. **Unit Testing**: Testing individual components of the software to ensure they function as intended.
2. **Integration Testing**: Testing how multiple components of the software interact with each other.
3. **System Testing**: Testing the entire system to ensure it meets functional and non-functional requirements.
4. **Acceptance Testing**: Testing the software with real-world scenarios to ensure it meets user requirements.

5. **Regression Testing**: Testing previously working functionality after changes to ensure no new issues have been introduced.
6. **Performance Testing**: Testing the system's performance under varying load conditions.
7. **Security Testing**: Testing the system's security and vulnerability to attacks.

6.4 Testing Techniques

The testing technique used in our projects are:

- Unit testing
- Integration testing
- System testing
- Performance testing

6.4.1 Unit Testing

Unit testing is a type of software testing where individual units or components of the software are tested in isolation to ensure they function as expected. Each unit is tested separately from the rest of the system to identify and isolate any defects or errors. The goal of unit testing is to verify that each unit performs as intended and meets its functional requirements. This type of testing is typically performed by developers during the development phase and helps to ensure the quality and reliability of the software.

Reason to use unit testing:

1. Catching defects early: Unit testing helps identify defects or errors in the code early in the development process, reducing the overall cost of fixing them later.
2. Ensuring code quality: Unit testing helps ensure the code meets the expected requirements and follows coding standards, increasing the overall quality of the software.
3. Facilitating change: Unit testing helps identify the impact of changes to the codebase, making it easier to make changes without breaking the software.
4. Encouraging modularity: Unit testing encourages developers to create modular code that can be tested in isolation, making it easier to identify and fix issues.
5. Enabling automation: Unit testing can be automated, reducing the overall time and effort required to test the code and ensuring consistent testing.

6. Providing documentation: Unit tests serve as documentation of the expected behaviour of the code, making it easier for new developers to understand the codebase.

6.4.2 Integration Testing

Integration testing is a type of software testing where multiple components or modules of the software are combined and tested together as a group to ensure they work correctly and seamlessly. The goal of integration testing is to identify any defects or errors that occur when different components are combined, and to ensure that they interact with each other as intended. This type of testing is typically performed after unit testing and before system testing. Integration testing helps to identify issues early in the development process, reducing the overall cost and time required for testing and development.

Reason to use integration testing:

1. Verifying component interaction: Integration testing helps ensure that different components or modules of the software work together as intended and interact correctly.
2. Identifying defects early: Integration testing helps identify defects or errors that occur when different components are combined, reducing the overall cost of fixing them later.
3. Ensuring reliability: Integration testing helps ensure that the software is reliable and functions correctly as a whole.
4. Improving overall quality: Integration testing helps improve the overall quality of the software by identifying and fixing issues early in the development process.
5. Facilitating change: Integration testing helps identify the impact of changes to the codebase on the software as a whole, making it easier to make changes without breaking the software.
6. Meeting user expectations: Integration testing helps ensure that the software meets the functional and non-functional requirements and meets user expectations.

6.4.3 System Testing

System testing is a type of software testing that is performed on a complete, integrated system to evaluate its compliance with functional and non-functional requirements. It verifies that the software system meets its intended purpose, functionality, performance, and reliability.

System testing is usually conducted after integration testing is completed, and is performed on the entire system, as opposed to individual components or modules. The goal of system testing is to identify defects and errors that occur when the system is tested as a whole, and to ensure that the system meets the user's requirements and expectations.

Reason to use system testing:

1. Ensure that the system meets functional and non-functional requirements: System testing verifies that the software system meets the functional and non-functional requirements specified in the requirements document.
2. Identify defects early: System testing helps identify defects or issues that occur when the system is tested as a whole, reducing the overall cost of fixing them later.
3. Validate system functionality: System testing validates that the system works as intended and all its features are functioning correctly.
4. Ensure reliability and stability: System testing helps ensure that the software system is reliable, stable, and performs correctly under different load conditions.
5. Validate security and compatibility: System testing validates that the system is secure and compatible with other systems and platforms.
6. Meet user expectations: System testing helps ensure that the software system meets the user's requirements and expectations, and delivers the desired user experience.

6.4.4 Performance Testing

Performance testing is a type of software testing that is designed to evaluate the system's responsiveness, speed, stability, and scalability under various workloads. It measures the performance of the system and identifies performance bottlenecks and issues that can impact user experience. Performance testing can help determine how much load the system can handle before it becomes unstable, how fast it can perform under different load conditions, and how quickly it can recover from failures. This type of testing is typically conducted using specialized tools that simulate real-world usage scenarios and generate metrics and reports that can be used to optimize the system's performance.

Reason to use performance testing:

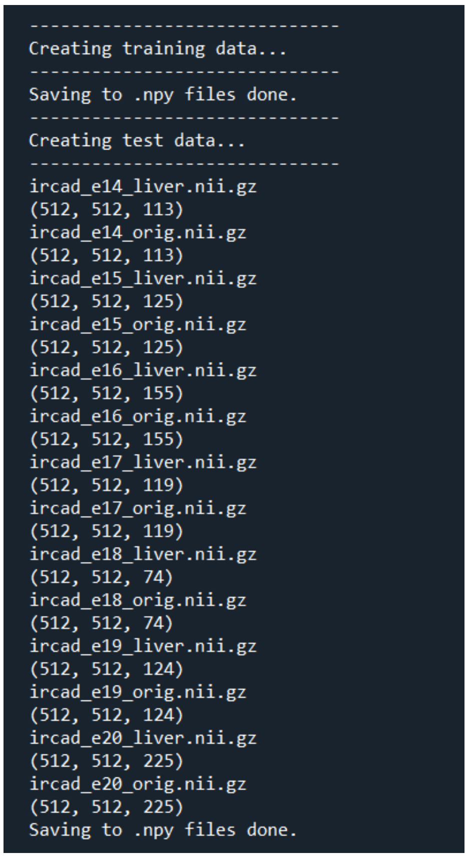
1. Ensure that the system meets functional and non-functional requirements: System testing verifies that the software system meets the functional and non-functional requirements specified in the requirements document.

2. Identify defects early: System testing helps identify defects or issues that occur when the system is tested as a whole, reducing the overall cost of fixing them later.
3. Validate system functionality: System testing validates that the system works as intended and all its features are functioning correctly.
4. Ensure reliability and stability: System testing helps ensure that the software system is reliable, stable, and performs correctly under different load conditions.
5. Validate security and compatibility: System testing validates that the system is secure and compatible with other systems and platforms.
6. Meet user expectations: System testing helps ensure that the software system meets the user's requirements and expectations and delivers the desired user experience.

6.5 Summary

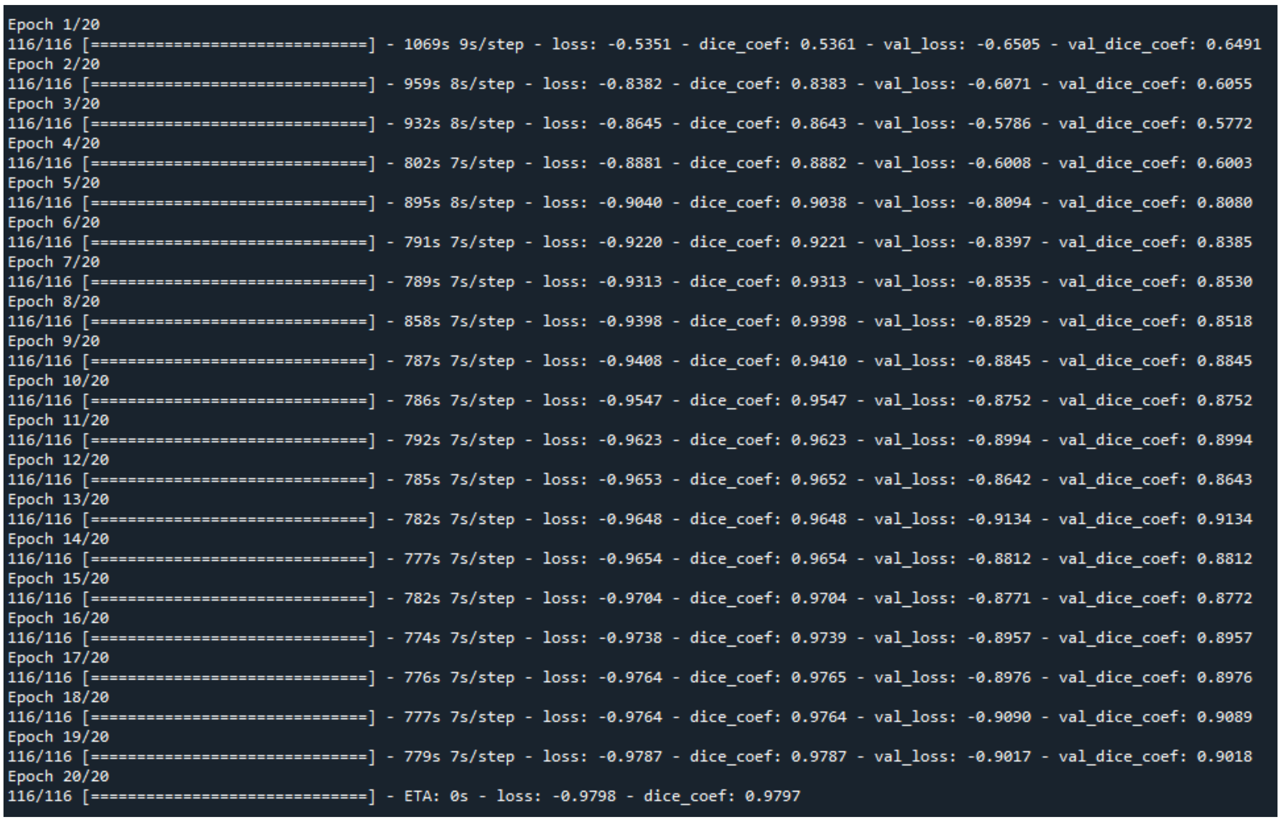
This chapter gave an immense idea about the software testing and various testing methods that are used in our software and reasons to choose some of the particular testing method.

7. SCREENSHOTS



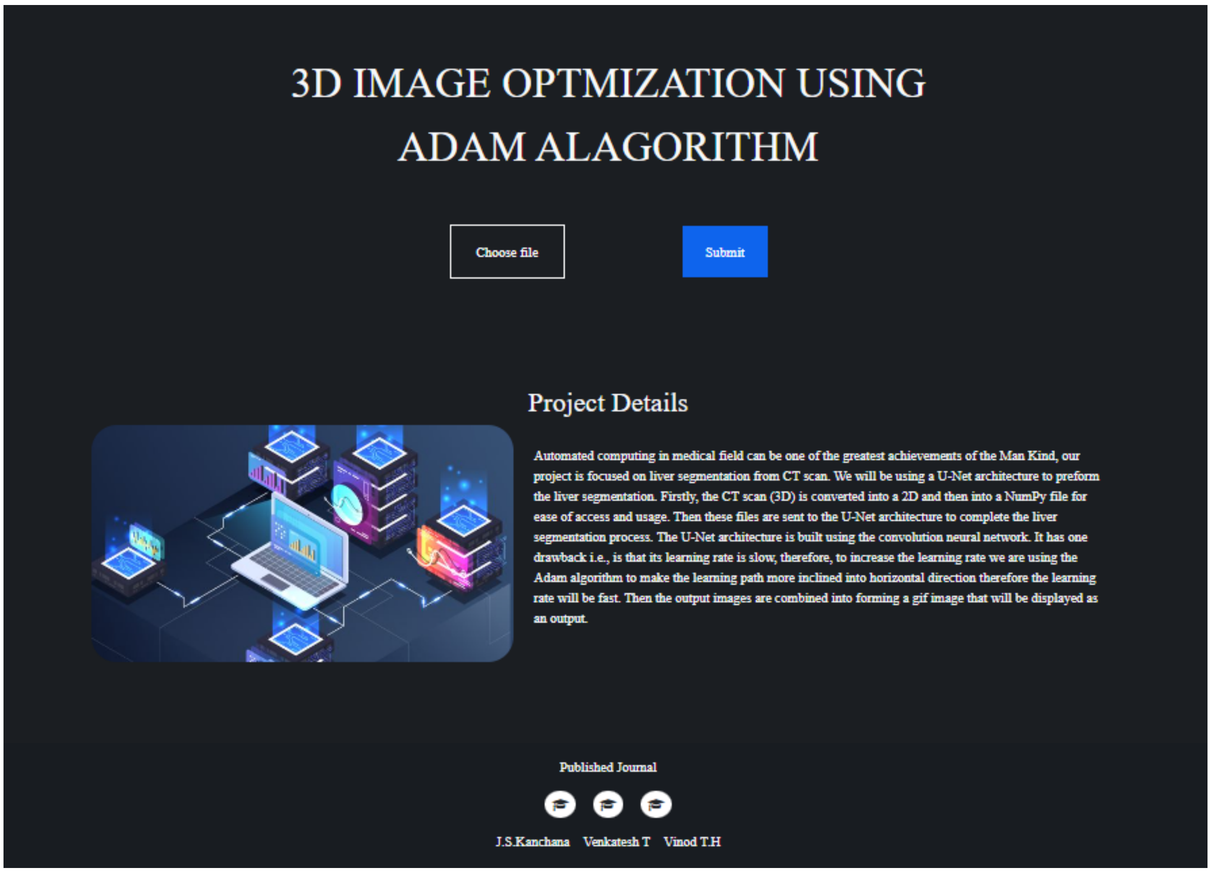
```
------------------------------
Creating training data...
------------------------------
Saving to .npy files done.
------------------------------
Creating test data...
------------------------------
ircad_e14_liver.nii.gz
(512, 512, 113)
ircad_e14_orig.nii.gz
(512, 512, 113)
ircad_e15_liver.nii.gz
(512, 512, 125)
ircad_e15_orig.nii.gz
(512, 512, 125)
ircad_e16_liver.nii.gz
(512, 512, 155)
ircad_e16_orig.nii.gz
(512, 512, 155)
ircad_e17_liver.nii.gz
(512, 512, 119)
ircad_e17_orig.nii.gz
(512, 512, 119)
ircad_e18_liver.nii.gz
(512, 512, 74)
ircad_e18_orig.nii.gz
(512, 512, 74)
ircad_e19_liver.nii.gz
(512, 512, 124)
ircad_e19_orig.nii.gz
(512, 512, 124)
ircad_e20_liver.nii.gz
(512, 512, 225)
ircad_e20_orig.nii.gz
(512, 512, 225)
Saving to .npy files done.
```

Figure 7.1: Data Pre-Processing



```
Epoch 1/20
116/116 [==============================] - 1069s 9s/step - loss: -0.5351 - dice_coef: 0.5361 - val_loss: -0.6505 - val_dice_coef: 0.6491
Epoch 2/20
116/116 [==============================] - 959s 8s/step - loss: -0.8382 - dice_coef: 0.8383 - val_loss: -0.6071 - val_dice_coef: 0.6055
Epoch 3/20
116/116 [==============================] - 932s 8s/step - loss: -0.8645 - dice_coef: 0.8643 - val_loss: -0.5786 - val_dice_coef: 0.5772
Epoch 4/20
116/116 [==============================] - 802s 7s/step - loss: -0.8881 - dice_coef: 0.8882 - val_loss: -0.6008 - val_dice_coef: 0.6003
Epoch 5/20
116/116 [==============================] - 895s 8s/step - loss: -0.9040 - dice_coef: 0.9038 - val_loss: -0.8094 - val_dice_coef: 0.8080
Epoch 6/20
116/116 [==============================] - 791s 7s/step - loss: -0.9220 - dice_coef: 0.9221 - val_loss: -0.8397 - val_dice_coef: 0.8385
Epoch 7/20
116/116 [==============================] - 789s 7s/step - loss: -0.9313 - dice_coef: 0.9313 - val_loss: -0.8535 - val_dice_coef: 0.8530
Epoch 8/20
116/116 [==============================] - 858s 7s/step - loss: -0.9398 - dice_coef: 0.9398 - val_loss: -0.8529 - val_dice_coef: 0.8518
Epoch 9/20
116/116 [==============================] - 787s 7s/step - loss: -0.9408 - dice_coef: 0.9410 - val_loss: -0.8845 - val_dice_coef: 0.8845
Epoch 10/20
116/116 [==============================] - 786s 7s/step - loss: -0.9547 - dice_coef: 0.9547 - val_loss: -0.8752 - val_dice_coef: 0.8752
Epoch 11/20
116/116 [==============================] - 792s 7s/step - loss: -0.9623 - dice_coef: 0.9623 - val_loss: -0.8994 - val_dice_coef: 0.8994
Epoch 12/20
116/116 [==============================] - 785s 7s/step - loss: -0.9653 - dice_coef: 0.9652 - val_loss: -0.8642 - val_dice_coef: 0.8643
Epoch 13/20
116/116 [==============================] - 782s 7s/step - loss: -0.9648 - dice_coef: 0.9648 - val_loss: -0.9134 - val_dice_coef: 0.9134
Epoch 14/20
116/116 [==============================] - 777s 7s/step - loss: -0.9654 - dice_coef: 0.9654 - val_loss: -0.8812 - val_dice_coef: 0.8812
Epoch 15/20
116/116 [==============================] - 782s 7s/step - loss: -0.9704 - dice_coef: 0.9704 - val_loss: -0.8771 - val_dice_coef: 0.8772
Epoch 16/20
116/116 [==============================] - 774s 7s/step - loss: -0.9738 - dice_coef: 0.9739 - val_loss: -0.8957 - val_dice_coef: 0.8957
Epoch 17/20
116/116 [==============================] - 776s 7s/step - loss: -0.9764 - dice_coef: 0.9765 - val_loss: -0.8976 - val_dice_coef: 0.8976
Epoch 18/20
116/116 [==============================] - 777s 7s/step - loss: -0.9764 - dice_coef: 0.9764 - val_loss: -0.9090 - val_dice_coef: 0.9089
Epoch 19/20
116/116 [==============================] - 779s 7s/step - loss: -0.9787 - dice_coef: 0.9787 - val_loss: -0.9017 - val_dice_coef: 0.9018
Epoch 20/20
116/116 [==============================] - ETA: 0s - loss: -0.9798 - dice_coef: 0.9797
```

Figure 7.2: Building U-Net architecture.



3D IMAGE OPTMIZATION USING ADAM ALAGORITHM

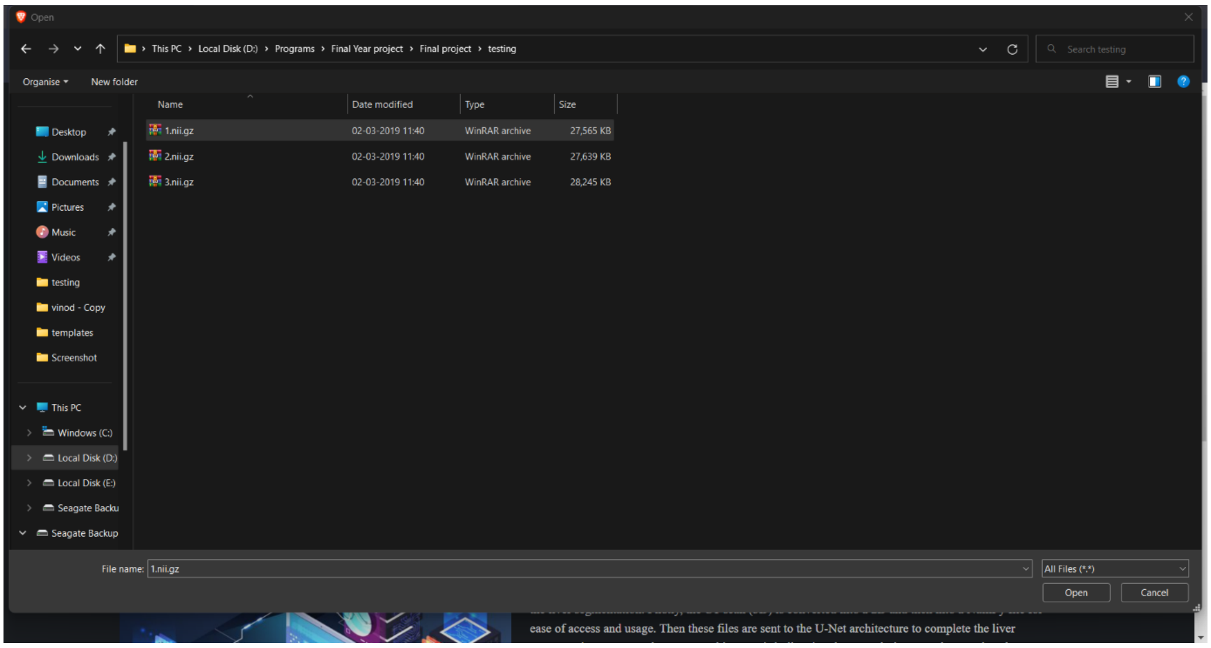
Project Details

Automated computing in medical field can be one of the greatest achievements of the Man Kind, our project is focused on liver segmentation from CT scan. We will be using a U-Net architecture to preform the liver segmentation. Firstly, the CT scan (3D) is converted into a 2D and then into a NumPy file for ease of access and usage. Then these files are sent to the U-Net architecture to complete the liver segmentation process. The U-Net architecture is built using the convolution neural network. It has one drawback i.e., is that its learning rate is slow, therefore, to increase the learning rate we are using the Adam algorithm to make the learning path more inclined into horizontal direction therefore the learning rate will be fast. Then the output images are combined into forming a gif image that will be displayed as an output.

Published Journal

J.S.Kanchana Venkatesh T Vinod T.H

Figure 7.3: Web app home page



Name	Date modified	Type	Size
1.nii.gz	02-03-2019 11:40	WinRAR archive	27,565 KB
2.nii.gz	02-03-2019 11:40	WinRAR archive	27,639 KB
3.nii.gz	02-03-2019 11:40	WinRAR archive	28,245 KB

File name: 1.nii.gz
All Files (*.*)

ease of access and usage. Then these files are sent to the U-Net architecture to complete the liver

Figure 7.4: Uploading the data via web application.

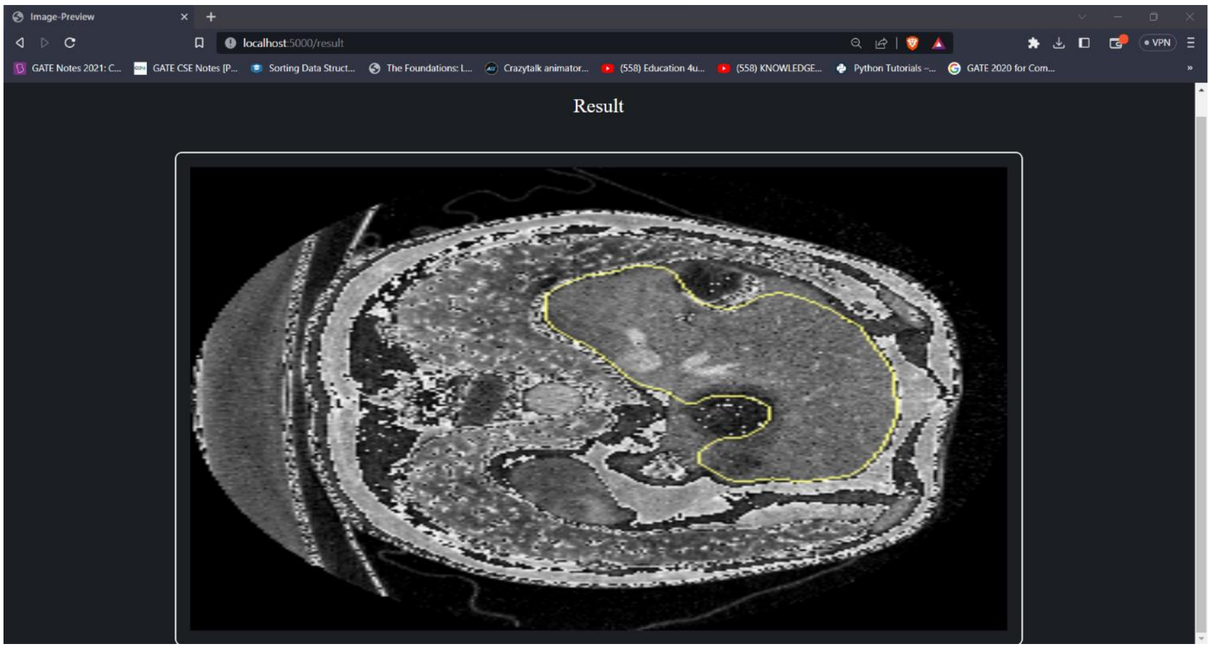


Figure 7.5: Output gif image is show.

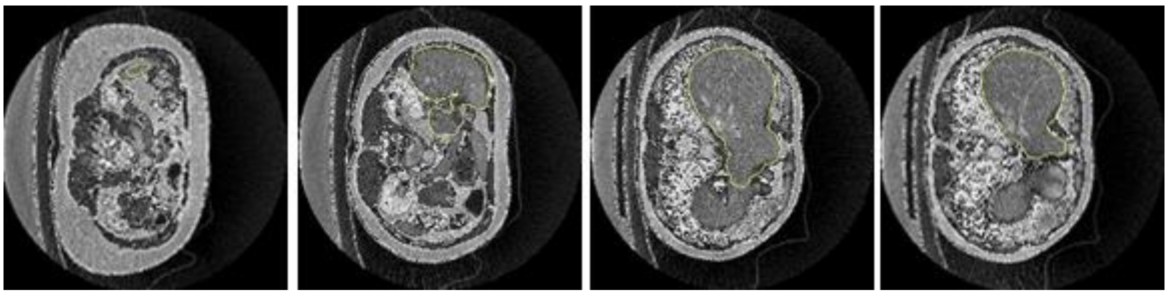


Figure 7.6: 4 important stages of the output

Table 7.1. Comparison of U-Net and Adam with U-Net

Method	U-Net	U-Net + Adam Algorithm
Dice – Coefficient	0.8485	0.9797
Val Dice – Coefficient	0.8427	0.9018
Loss	-0.7312	-0.9798
Val Loss	-0.7268	-0.9017

Table 7.2. All the value of Dice coefficient with respect to the epoch

SN.	Epoch	Total time	Time per step	Dice coefficient
1	Epoch 1/20	1069s	9s/step	0.5361
2	Epoch 2/20	959s	8s/step	0.8383
3	Epoch 3/20	932s	8s/step	0.8643
4	Epoch 4/20	802s	7s/step	0.8882
5	Epoch 5/20	895s	8s/step	0.9038
6	Epoch 6/20	791s	7s/step	0.9221
7	Epoch 7/20	789s	7s/step	0.9313
8	Epoch 8/20	858s	7s/step	0.9398
9	Epoch 9/20	787s	7s/step	0.9410
10	Epoch 10/20	786s	7s/step	0.9623
11	Epoch 11/20	792s	7s/step	0.9652
12	Epoch 12/20	785s	7s/step	0.9648
13	Epoch 13/20	782s	7s/step	0.9654
14	Epoch 14/20	777s	7s/step	0.9704
15	Epoch 15/20	782s	7s/step	0.9765
16	Epoch 16/20	774s	7s/step	0.9765
17	Epoch 17/20	776s	7s/step	0.9787
18	Epoch 18/20	777s	7s/step	0.9674
19	Epoch 19/20	779s	7s/step	0.9787
20	Epoch 20/20	0s	-	0.9797

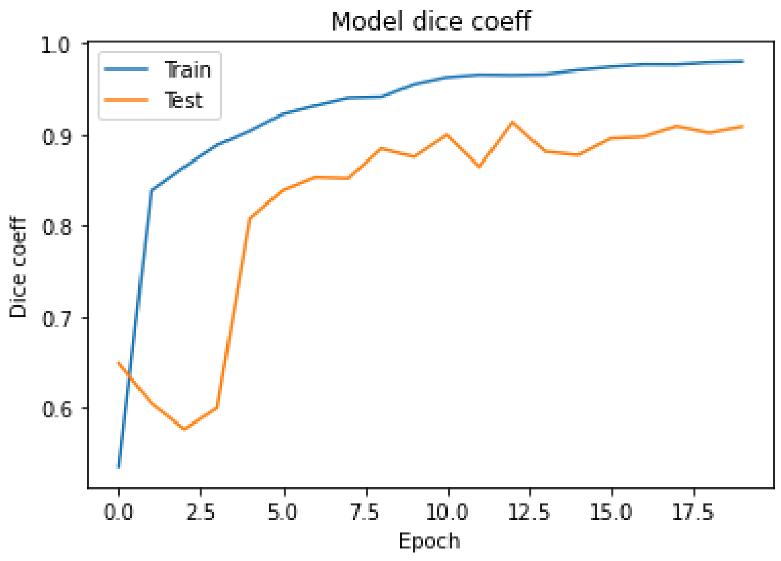


Figure 7.7: comparison between orginal image and predicted image

8. CONCLUSION

8.1 Conclusion CHAPTER 8

The U-Net architecture is a type of deep learning model that has gained popularity in the field of image segmentation. By optimizing this model with the Adam algorithm a fast-learning rate is achieved. The model can learn from data more efficiently and quickly. This optimized U-Net model has the potential to revolutionize various industries, such as healthcare and autonomous vehicles. To increase the usability of this model, a web application is implemented using Flask as the backend framework.

All the business logic has been scripted in Python, making it a versatile and flexible platform for future modifications. The web application can now be used to process images in real-time and provide accurate segmentation results to end-users. Overall, the combination of an optimized U-Net model and a user-friendly web application makes this a valuable tool for a wide range of applications.

8.2 Future Enhancement

To enhance the of the model using pseudo-labeled data. This is particularly important in the medical field, where obtaining sufficient datasets can be challenging. With relatively low amounts of data available, training the model can be difficult. Pseudo 68abelling offers a solution to this issue by utilizing unlabeled data and assigning labels to them. This approach can help increase the quantity and quality of the dataset and therefore improve model performance.

In addition, we plan to integrate the model with remote resources to increase the epoch size. By leveraging external computing resources, we can extend the training time and optimize the model more effectively. This integration will also help with processing larger datasets and enable the model to handle more complex data.

Overall, these enhancements will significantly enhance the performance, making it more reliable and accurate. This will be particularly valuable in the medical field, where the results of machine learning models can have a significant impact on patient outcomes.

9. APPENDIX

9.1 Python **CHAPTER 9**

Python is a high-level, interpreted programming language that is widely used for various applications such as web development, data science, artificial intelligence, machine learning, automation, and many more. It was created by Guido van Rossum in the late 1980s and released in 1991. Python is known for its simplicity, readability, and easy-to-learn syntax, which makes it a popular choice for beginners.

In advanced level Python programming, one can explore a wide range of features such as object-oriented programming, functional programming, decorators, generators, asynchronous programming, concurrency, and more. One can also dive deep into libraries and frameworks such as NumPy, Pandas, Scikit-learn, TensorFlow, Keras, Django, Flask, and many others.

Python’s object-oriented programming features allow developers to create and use classes, objects, and methods, making it easy to organize and reuse code. Functional programming is another popular paradigm in Python, which allows developers to write code using pure functions without side effects, enabling them to write cleaner, more efficient code.

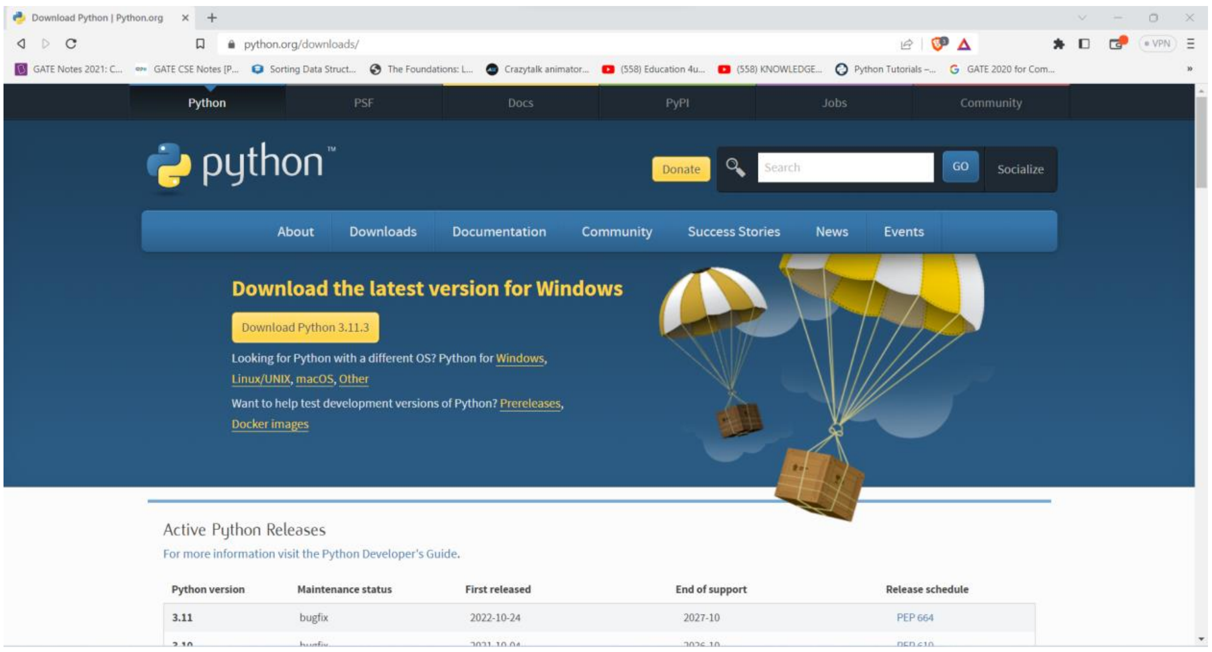
Decorators are another feature in Python that allow developers to modify the behavior of functions or classes at runtime. Generators enable developers to create iterators in a memory-efficient way, while asynchronous programming enables developers to write code that can handle multiple tasks simultaneously, making it ideal for high-performance applications.

Python’s standard library is extensive and includes modules for tasks such as file I/O, regular expressions, network programming, and more. Additionally, the open-source community has developed a vast ecosystem of libraries and frameworks that can be used to solve a wide range of problems.

Overall, Python is a versatile programming language that can be used for various applications, from web development to artificial intelligence. Its ease of use, vast community support, and growing ecosystem of libraries and frameworks make it an excellent choice for both beginners and advanced programmers.

Steps to Install Python

1. First, you need to download the Python installer from the official website. Go to https://www.python.org/downloads/ and choose the latest version of Python.



Download the latest version for Windows

Download Python 3.11.3

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python? Prereleases, Docker images

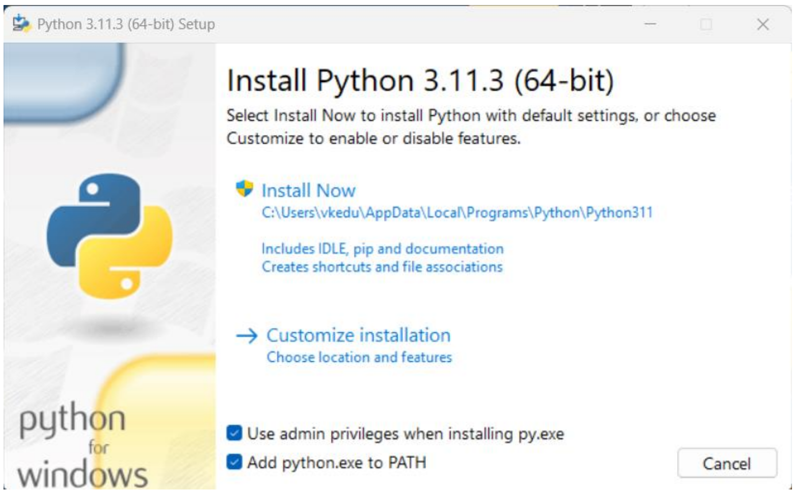
Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.11	bugfix	2022-10-24	2027-10	PEP 664

Figure 9.1.1: python website home page

2. Once the installer is downloaded, run the .exe file and click "Install Now".



Install Python 3.11.3 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\vkedu\AppData\Local\Programs\Python\Python311

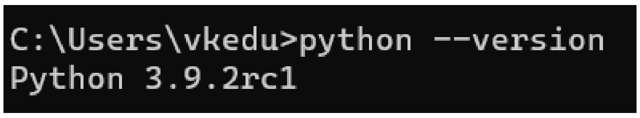
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

- [x] Use admin privileges when installing py.exe
- [x] Add python.exe to PATH

Figure 9.1.2: software installation

3. On the next screen, you can choose the components you want to install. You can choose to add Python to PATH to make it easier to access from the command line.
4. On the Advanced Options screen, you can customize the installation location and configure other settings.
5. After reviewing the settings, click “Install” to begin the installation process.
6. Once the installation is complete, you can launch Python from the Start Menu or by typing “python” in the command prompt.
7. To check if Python is installed correctly, open the command prompt and type “python –version”. This should display the version of Python you have installed.



```
C:\Users\vkedu>python --version
Python 3.9.2rc1
```

Figure 9.1.3: python installation checking

9.2 Anaconda Distribution

Anaconda is a popular distribution of Python programming language that provides a platform for data science and machine learning. It includes a collection of open-source software packages for scientific computing, data analysis, and visualization. Anaconda comes with its own package manager called conda, which allows users to easily manage and install packages.

One of the main advantages of Anaconda is that it provides a comprehensive environment for data analysis and machine learning. It includes popular packages such as NumPy, Pandas, Scikit-learn, and TensorFlow, among others. Moreover, it also provides tools for creating and managing virtual environments, making it easy to switch between different projects and their dependencies. Overall, Anaconda simplifies the process of setting up a data science environment and provides a streamlined experience for data analysts and developers.

Steps to install anaconda are :

1. Visit the official Anaconda website (https://www.anaconda.com/products/individual) and download the appropriate version of Anaconda for your operating system.

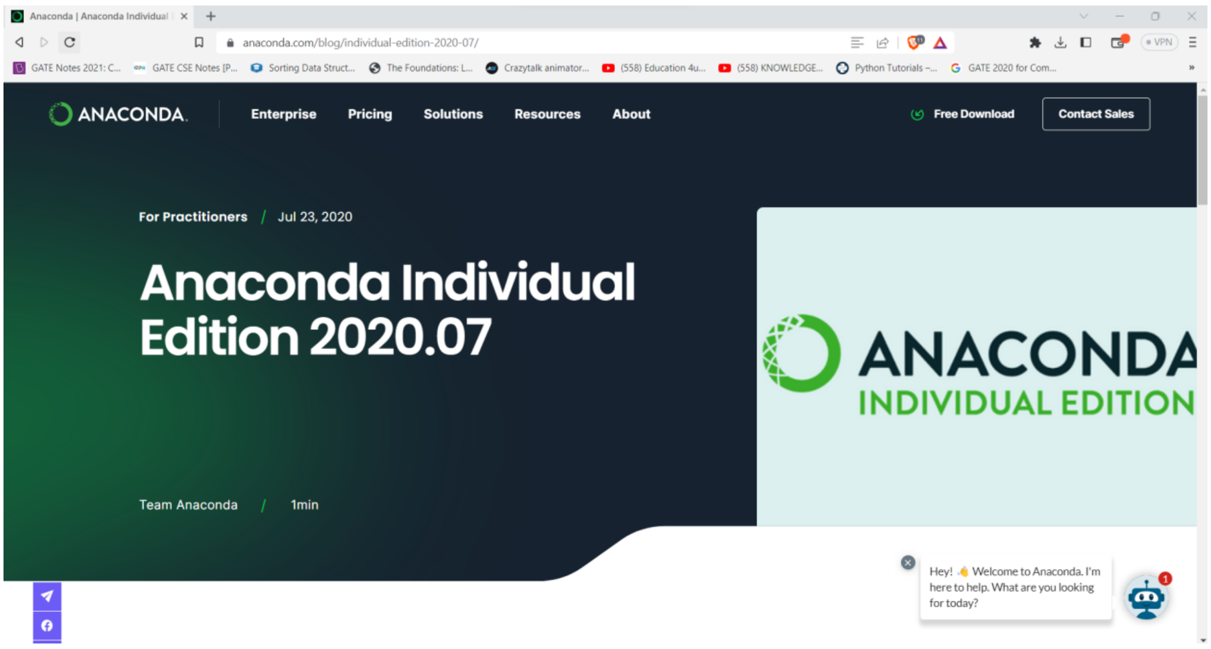
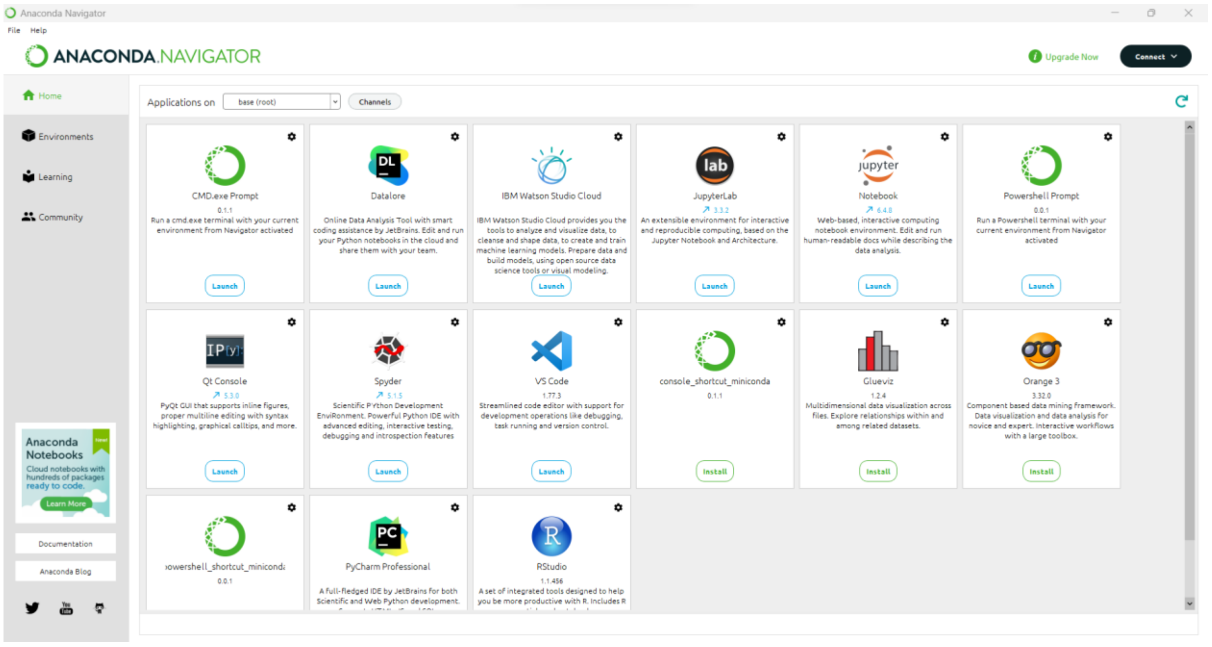

anaconda.com/blog/individual-edition-2020-07/
ANACONDA
Enterprise
Pricing
Solutions
Resources
About
Free Download
Contact Sales
For Practitioners / Jul 23, 2020
Anaconda Individual Edition 2020.07
Team Anaconda / 1min
ANACONDA
INDIVIDUAL EDITION
Hey! Welcome to Anaconda. I'm here to help. What are you looking for today?

Figure 9.2.1: Anaconda distribution home page

2. Once the download is complete, launch the Anaconda installer.
3. Follow the instructions provided by the installer, and choose the desired installation options, such as installation directory and whether to add Anaconda to your system PATH.
4. Once the installation is complete, you can launch Anaconda Navigator or Anaconda Prompt to start working with Anaconda.
5. In Anaconda Navigator, you can use the graphical user interface to manage packages, environments, and applications. In Anaconda Prompt, you can use the command line interface to manage packages and environments.



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

Datalore
Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab
3.3.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.4.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
5.3.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
5.1.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
1.77.3
Streamlined code editor with support for development operations like debugging, task running and version control.

console_shortcut_miniconda
0.1.1

Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.32.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

owershell_shortcut_minicond
0.0.1

PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R

Figure 9.2.2: Anaconda navigation GUI

6. To create a new environment, use the conda create command, followed by the name of the environment and the packages you want to include.
7. To activate an environment, use the conda activate command, followed by the name of the environment.

9.3 Libraries used and its installation.

The libraries used are:

- Scikit-image
- Imageio
- Flask

- Nibabel
- Tensorflow
- Keras
- Os
- Numpy

9.3.1 Scikit-image

scikit-image is a Python library for image processing and computer vision. It provides various algorithms for image enhancement, segmentation, feature extraction, and more.

To install scikit-image in Python, the following command can be used:

```
pip install scikit-image
```

9.3.2 Nibabel

Nibabel is a Python library for reading and writing various neuroimaging file formats. It provides easy and efficient access to volumetric imaging data in different formats. Nibabel is widely used in neuroimaging research and analysis.

To install nibabel using pip, open the command prompt or terminal and type:

```
pip install nibabel
```

9.3.3 Keras

Keras is a high-level neural networks API written in Python, designed to enable fast experimentation with deep learning models. It provides a user-friendly interface that allows you to build, train, and deploy deep learning models with ease. Keras is known for its simplicity, modularity, and extensibility, and is widely used in industry and academia.

To install Keras in Python, you can use the following command:

```
pip install keras
```

This will install the latest version of Keras along with its dependencies. If you prefer to install a specific version, you can specify the version number using the following command:

```
pip install keras==<version>
```

Replace <version> with the version number you want to install (e.g., 2.4.3).

9.3.4 Numpy

NumPy is a Python library for numerical computing, providing support for large, multi-dimensional arrays and matrices, as well as a range of mathematical functions to operate on these arrays. NumPy is widely used in scientific computing, data analysis, and machine learning.

To install NumPy in Python, you can use the following command:

```
pip install numpy
```

This will install the latest version of NumPy available on the Python Package Index (PyPI). Alternatively, you can install a specific version of NumPy by specifying the version number after the package name, like this:

```
pip install numpy==1.19.3
```

This will install NumPy version 1.19.3 specifically.

9.3.5 OS

The OS module is a Python built-in module that provides a way of using operating system dependent functionality. It allows Python programs to interact with the underlying operating system in various ways. This module provides a portable way of using operating system dependent functionality.

In Python, the OS module is available by default, so there is no need to install it separately. It can be imported in a Python program using the following command:

```
import os
```

9.3.6 TensorFlow

TensorFlow is an open-source software library developed by Google for building and training machine learning models such as deep neural networks. It can run on multiple platforms such as CPUs, GPUs, and even mobile devices. TensorFlow offers a wide range of tools and APIs for building and deploying machine learning models efficiently. To install TensorFlow in Python, you can use the following command:

```
!pip install tensorflow
```

This will install the latest version of TensorFlow available in the Python package index (PyPI).

9.3.7 Flask

Flask is a popular Python web framework used for building web applications, APIs, and microservices. It is a lightweight and flexible framework that provides various features such as routing, templating, request handling, and more. Flask is easy to use and requires minimal setup, making it a popular choice for small to medium-sized projects. To install Flask, you can use the following command in Python:

```
pip install Flask
```

9.3.8 Imageio

Imageio is a Python library for reading and writing image data. It is designed to be simple and easy to use and supports a wide range of image file formats. With imageio, you can easily load images into your Python programs, manipulate them, and save them back out to disk. To install Imageio, you can use the following command in your terminal or command prompt:

```
pip install imageio
```

This will download and install the latest version of Imageio from the Python Package Index (PyPI), along with any required dependencies.

10. REFERENCES

REFERENCES CHAPTER 10

[1] Kai Han, Lu Liu , Yuqing Song, Yi Liu, Chengjian Qiu, Yangyang Tang, Qiaoying Teng , and Zhe Liu, “An Effective Semi-Supervised Approach for Liver CT Image Segmentation”, IEEE journal of biomedical and health informatics, vol. 26, no. 8, august 2022

[2] Y. Xia et al., “3D semi-supervised learning with uncertainty-aware multiview co-training,” in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis., 2020, pp. 3646–3655.

[3] J. Peng, G. Estrada, M. Pedersoli, and C. Desrosiers, “Deep co-training for semi-supervised image segmentation,” Pattern Recognit., vol. 107, 2020, Art. no. 107269.

[4] D.-P. Fan et al., “Inf-Net: Automatic COVID-19 lung infection segmentation from CT images,” IEEE Trans. Med. Imag., vol. 39, no. 8, pp. 2626–2637, Aug. 2020.

[5] L. Yu, S. Wang, X. Li, C.-W. Fu, and P.-A. Heng, “Uncertainty-aware self-ensembling model for semi-supervised 3D left atrium segmentation,” in Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv., 2019, pp. 605–613.

[6] Y. Zhou et al., “Semi-supervised 3D abdominal multi-organ segmentation via deep multi-planar co-training,” in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2019, pp. 121–140.

[7]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. Imag., vol. 37, no. 12, pp. 2663–2674, Dec. 2018.

[8] Q. Yu, L. Xie, Y. Wang, Y. Zhou, E. K. Fishman, and A. L. Yuille, “Recurrent saliency transformation network: Incorporating multi-stage visual cues for small organ segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 8280–8289.

[9] W. Bai et al., “Semi-supervised learning for network-based cardiac MR image segmentation,” in Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Interv., 2017, pp. 253–260.

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[11] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 3431–3440.

[12] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Interv., 2015, pp. 234–241.