

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING

★ ★ ★



UNDERGRADUATE THESIS

BUILDING A SECURITY MODULE
FOR WEB FIREWALL

COMPUTER SCIENCE COMMITTEE

Supervisors: DR. NGUYEN AN KHUONG, PH.D.
MR. LE DINH THUAN, M.Sc.
MR. NGUYEN VAN HOA, B.ENG.
MR. LE NGUYEN MINH KHOI, B.ENG.
GVPB: THS. TRẦN GIANG SƠN

Students TRINH CONG VU 1852882
LE THIEN THANH 1814009

Ho Chi Minh City, 06/2023

COMMITMENT

We vouch for the fact that the work in this dissertation was completed in accordance with the guidelines established by the university and was not completed for submission to any other academic bodies. The works are our own, unless otherwise noted by a particular citation in the text.

HO CHI MINH CITY, NOVEMBER 2022

ACKNOWLEDGEMENT

First and foremost, we want to express our appreciation to our supervisors, professor Nguyen An Khuong and Le Dinh Thuan, for their patience, inspiration, and extensive expertise, all of which were crucial in assisting us with the thesis. Throughout the entire process of working and producing the thesis, their advice was helpful.

We would like to express our thankfulness to Nguyen Van Hoa and Le Nguyen Minh Khoi for their particular assistance and important information sharing. We cannot complete a fantastic thesis without their assistance.

Also, we want to thank the data support from *POLARIS Cybersecurity Joint Stock Company* for us to successfully complete this thesis.

In addition, we want to express our gratitude to our families and friends who have helped and supported us greatly during this thesis and our time at university.

THESIS GROUP MEMBERS.

PREFACE

With the increased exchange of information and other activity on the World Wide Web, the Web has become the primary platform for attackers to cause havoc. Effective methods for detecting Web threats are crucial for ensuring Web security. With the explosion of data, it is becoming more challenging to manually detect and prevent cyber threats; however, the use of machine learning-based ways to fight cybersecurity is on the rise, with broad uses of machine learning in cybersecurity providers.

In this thesis, we propose a machine learning-based strategy to cyber-attack defense. Typically, rule-based WAFs have been widely employed. They do, however, have a significant false-positive rate. Typically, rule-based WAFs have been widely employed. They do, however, have a significant false-positive rate. As a consequence, we are building a machine learning-based module to validate requests that have been labeled suspicious by WAFs in order to enhance the WAFs and provide better surveillance.

The malicious request validator is based on the assumption that genuine requests to a website generally fit into the same category. The module use a Convolutional Neural Network to classify the suspected request and evaluates whether it falls into the same category as the standard requests observed. The ultimate decision on whether to block a request is made using this result and combining it with the request content analyzer.

The dataset for phishing detection is collected from Polaris Infosec, a Web Application & API Protection (WAAP).

The false-positive rate, often referred as the false alarm rate, is the most crucial requirement for modules in experimentation. Precision and accuracy are other key criteria. We also accounted for processing time with the suspicious request detector, as WAFs are supposed to be instant.

LIST OF FIGURES

2.1	Normal users (top-left and bottom-left) are permitted access to the server with a WAF enabled, but attackers (middle-left) are prevented from doing so.	6
2.2	Phép toán Max-pooling với kích thước kernel 2×2 và stride 2.	8
2.3	Receptive field của phép toán Convolution có kích thước filter 3×3 , stride 2 và padding “same”.	8
2.4	Giải thuật Batchnorm áp dụng lên dữ liệu đầu vào của một mini-batch.	9
2.5	Đồ thị biểu diễn hàm sigmoid.	10
2.6	Đồ thị biểu diễn hàm ReLU.	11
2.7	Một số ví dụ về đầu vào và đầu ra của hàm Softmax.	11
2.8	Phép toán Dilation trên A áp dụng thành phần cấu trúc B	13
2.9	Phép toán Erosion trên A áp dụng thành phần cấu trúc B	13
2.10	Áp dụng phép toán Skeleton để rút trích khung xương đối tượng trong không gian 2D.	14
2.11	Áp dụng phép toán Skeleton để rút trích khung xương đối tượng trong không gian 3D.	15
4.1	Logo Viện nghiên cứu chống ung thư đường tiêu hoá ở Pháp.	20
4.2	Ảnh trực quan gói dữ liệu <i>3D-IRCADb-01</i>	20
4.3	Nhãn phân đoạn bị thiếu trên một số lớp ảnh ở bệnh nhân số 2.	22
4.4	Giá trị nhãn phân đoạn tĩnh mạch chủ trên bệnh nhân số 1	24
4.5	Giá trị nhãn phân đoạn tĩnh mạch chủ trên bệnh nhân số 3	24
5.1	Returned information	30
5.2	Compulsory query	30
5.3	Vulnerable PHP Code	31
5.4	The hacker’s query and result	32
5.5	The code is protected against SQL injections	33
5.6	Malicious request validator architecture	37
5.7	Decision model for the combination of CNN and the Regression model	37
6.1	Nội suy được sử dụng để lấp các lớp bị thiếu trong khối ảnh CT.	40
6.2	Mô phỏng quá trình trích xuất thành phần gan trong khối ảnh CT.	41
6.3	Phân phối mức sáng của các điểm ảnh thuộc nền cơ quan gan và mạch máu gan cùng ngưỡng giới hạn trái và phải.	42
6.4	Biến đổi cường độ sáng điểm ảnh để làm rõ mạch máu.	43
6.5	Làm giàu dữ liệu bằng phép trích xuất khối ngẫu nhiên.	43
6.6	Kiến trúc của bộ mã nguồn <i>Insight Deep Learning</i>	44
6.7	Trực quan hóa kết quả trên ứng dụng Slicer.	46
7.1	Kết quả hệ thống mạch máu của trường hợp tốt nhất trong thí nghiệm 2. .	54
7.2	Kết quả hệ thống mạch máu của trường hợp tốt nhất trong thí nghiệm 2 và nhãn phân đoạn.	55

7.3	Kết quả tìm đường chính giữa và điểm phân nhánh của trường hợp tốt nhất trong thí nghiệm 2.	56
7.4	Kết quả hệ thống mạch máu của trường hợp xấu nhất trong thí nghiệm 2.	57
7.5	Kết quả hệ thống mạch máu của trường hợp xấu nhất trong thí nghiệm 2 và nhãn phân đoạn.	58
7.6	Kết quả tìm đường chính giữa và điểm phân nhánh của trường hợp xấu nhất trong thí nghiệm 2.	59
7.7	Kết quả hệ thống mạch máu của trường hợp tốt nhất trong thí nghiệm 8.	60
7.8	Kết quả hệ thống mạch máu của trường hợp tốt nhất trong thí nghiệm 8 và nhãn phân đoạn.	61
7.9	Kết quả tìm đường chính giữa và điểm phân nhánh của trường hợp tốt nhất trong thí nghiệm 8.	62
7.10	Kết quả hệ thống mạch máu của trường hợp xấu nhất trong thí nghiệm 8.	63
7.11	Kết quả hệ thống mạch máu của trường hợp xấu nhất trong thí nghiệm 8 và nhãn phân đoạn.	64
7.12	Kết quả tìm đường chính giữa và điểm phân nhánh của trường hợp xấu nhất trong thí nghiệm 8.	65
B.1	Kế hoạch thực hiện luận văn.	73

LIST OF TABLES

4.1	Tình trạng nhãn phân đoạn của cơ quan gan và hệ thống mạch máu.	23
4.2	Giá trị nhãn phân đoạn của cơ quan gan và hệ thống mạch máu.	25
5.1	Table 4.1	37
6.1	Thông tin phần cứng hệ thống máy tính.	45
7.1	Bảng phân chia các bệnh nhân thành các tập dữ liệu.	48
7.2	Thông số các thí nghiệm so sánh tính hiệu quả trước và sau điều chỉnh số tầng ở mô hình U-Net sử dụng convolution 3D.	50
7.3	Kết quả so sánh tính hiệu quả trước và sau điều chỉnh số tầng ở mô hình U-Net sử dụng convolution 3D.	50
7.4	Thông số các thí nghiệm tham khảo.	51
7.5	Kết quả các thí nghiệm tham khảo.	51
7.6	Kết quả thí nghiệm 2 của từng bệnh nhân trong tập kiểm tra.	51
7.7	Thông số các thí nghiệm đề xuất.	52
7.8	Kết quả các thí nghiệm đề xuất.	52
7.9	Kết quả thí nghiệm 8 của từng bệnh nhân trong tập kiểm tra.	52
7.10	Thông số các thí nghiệm có sự kết hợp của DenseNet và ResNet.	53
7.11	Kết quả các thí nghiệm DenseUNet và ResUNet.	53
A.1	Thông tin về tập dữ liệu <i>3D-IRCADb-01</i>	71

LIST OF ABBREVIATIONS

The following list shows the abbreviations that will be used in the text of this thesis.

BG	Background
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
FG	Foreground
GVLab	Graphics and Computer Vision Laboratory
HPCC	High Performance Computing Center
IoU	Intersection over Union
RAM	Random Access Memory
SGD	Stochastic Gradient Descent
VTK	The Visualization Toolkit
YAML	YAML Ain't Markup Language

TABLE OF CONTENTS

Preface	vii
List of Figures	ix
List of Tables	xi
List of Abbreviations	xiii
Chapter 1 INTRODUCTION	1
1.1 Motivation and problem statement	2
1.2 Objectives	3
1.3 Challenges	3
1.4 Tentative structure of the thesis	3
Chapter 2 BACKGROUNDS	5
2.1 Web Application Firewall	6
2.2 Các toán tử trong xử lý hình ảnh	12
2.3 Thư viện và công cụ	15
Chapter 3 LITERATURE REVIEW	17
Chapter 4 DATASET	19
4.1 Giới thiệu chung	20
4.2 Các vấn đề cần xử lý	21
Chapter 5 PROPOSED APPROACH	27
5.1 API security problems	28
5.2 Designs	34
Chapter 6 IMPLEMENTATION	39
6.1 Tiền xử lý dữ liệu	40
6.2 Xây dựng mã nguồn hệ thống	44
6.3 Đặc tả phần cứng	45
6.4 Hậu xử lý kết quả phân đoạn	45
6.5 Trực quan hóa kết quả thí nghiệm	46
Chapter 7 EXPERIMENTS	47
7.1 Chuẩn bị dữ liệu	48
7.2 Phương pháp đánh giá	48
7.3 Kết quả thí nghiệm	50

Chapter 8 CONCLUSION	67
8.1 Kết quả đạt được	68
8.2 Hạn chế và hướng phát triển	69
A THÔNG SỐ TẬP DỮ LIỆU 3D-IRCADB-01	71
B KẾ HOẠCH THỰC HIỆN LUẬN VĂN	73
References	75
List of Keywords	75

1

INTRODUCTION

In this chapter, we introduce issues in the field of cyber security, clarifies the necessity of building a firewall system. Next, we present an overview of the goals, challenges and structure of the thesis.

Table of Contents

1.1	Motivation and problem statement	2
1.2	Objectives	3
1.3	Challenges	3
1.4	Tentative structure of the thesis	3

1.1 Motivation and problem statement

Websites have become necessary for every business, brand, institution, organization, and individual. Due to the amount of information it collects, web services are becoming targets of cybercriminals. In the modern environment, data retention via web apps is the most efficient method. A corporation may be scammed using fraudulent data, which could have disastrous consequences for its finances and credibility. Therefore, it is necessary to draw attention to newly created web applications and developers.

Web-based applications offer the general public and enterprises fast and simple services, they may be used for social media, email, banking, online shopping, education, or entertainment. There are many widely used apps, and each one of them has the potential to be the next big exploit for threat actors. Web applications can be attacked for a variety of reasons, including system flaws caused by incorrect coding, misconfigured web servers, application design flaws, or failure to validate forms. These flaws and vulnerabilities enable attackers to gain access to databases containing sensitive information. Web applications are an easy target for attackers because they must be available to customers at all times.

Threat actors detect easy targets in the enormous section of insecure online applications. Cyber attacks are malicious attempts to access a person's or an organization's computer systems, networks, or data without authorization. Private data can be stolen, held for ransom, or destroyed. Recently, cyber attacks have improved in sophistication and tenacity, making it simpler for attackers to compromise a weak system and do serious harm.

Cyber attacks devastate businesses of all sizes and in any sector. Not only do they put your data at risk, but they can also lead to financial losses, reputational damage, and disruption of operations. The most common cyber attacks are phishing, malware, distributed denial of services, and ransomware attacks¹.

Let's take a look at the statistics about cybersecurity by industry. Healthcare, throughout the past 12 years, this sector has experienced the most costly data breaches, the costs have even increased by 41.6% from 2020 until 2022. At least 849 healthcare cybersecurity incidents and 571 data breaches were reported in 2022. The average financial loss due to data breaches in healthcare has skyrocketed from around USD 9 million to USD 10.10 million (2022). In the Finance industry, phishing attacks against banks and other financial institutions held the largest share, accounting for 23.2% of all cyberattacks targeting the financial sector. In the first quarter of 2022, ransomware assaults increased by 35% in the financial sectors. On average, financial organizations bore the second-highest data breach costs, at USD 5.97 million, just behind healthcare institutions (2022)².

From the above situation, we can see that tools like Firewalls play a crucial role in organizations' security systems. They are a powerful tool for preventing malicious traffic from entering or leaving an organization's systems. Our group wanted to make some efforts to strengthen firewalls' cyber security, so we decided to choose this topic: Building a Security module for Web firewalls.

1 Parachute. *Cyber Attack Statistics to Know in 2023.* <https://parachute.cloud/cyber-attack-statistics-data-and-trends/>

2 Parachute. *Cyber Attack Statistics to Know in 2023.* <https://parachute.cloud/cyber-attack-statistics-data-and-trends/>

1.2 Objectives

Given the necessity of cybersecurity in this era, we chose to construct a firewall add-on. This thesis will concentrate on identifying malicious requests that pass through the firewall. We want to apply machine learning to assist the WAF process a tremendous amount of requests per second swiftly and efficiently. The developed module should be able to validate the request and identify whether or not it is an attack.

To eliminate the high FP, a inherent weakness of rule-based WAF and assure the speed of the WAF, which must react nearly instantly to deliver an consistancy customer experience, the module will use two fast and simple machine learning models. Each model will run independently to generate an output, then combined the result to achieve the accuracy of at least 95% as well as low latency of 5 miliseconds¹ for the WAF.

1.3 Challenges

In this topic, we have encountered some problems such as:

1.4 Tentative structure of the thesis

The content of the thesis proposal is demonstrated by these 8 following parts:

Chapter 1 introduces issues in the field of cyber security, clarifies the necessity of building a firewall system. Next, we present an overview of the goals, challenges and structure of the thesis.

Chapter 2 introduces the background knowledge of this thesis, including the information about API, Web Application Firewall, Machine Learning Model and Deep Neural Network.

Chapter 3 mentions some related studies on the use of machine techniques to detect malicious requests, their strengths and weaknesses and evaluate an appropriate approach for this thesis.

Chapter 4 displays the problems as well as the malicious request validator's input and output. Then we offer the design and architecture of the problem's solutions.

Chapter 5 provides the dataset that would be used to train and evaluate the malicious request validator, its problems and pre-processing details

Chapter 6 shows the implementation and deployment processes, including system frameworks and system specifications.

¹ Abdalslam. Web Application Firewalls (WAF) Statistics, Trends And Facts 2023.
https://abdalslam.com/web-application-firewalls-waf-statistics?utm_source=rss&utm_medium=rss&utm_campaign=web-application-firewalls-waf-statistics

Chapter 7 This chapter discusses the division of the data set prior to training, assessment techniques, and experimental results. Compare the outcomes of the experiments we suggest to the reference experiments from relevant works.

Chapter 8 summarizes the results for the thesis until now. Finally, we want to present working plan to improve the thesis

2

BACKGROUNDS

introduces the background knowledge of this thesis, including the information about API, Web Application Firewall, Machine Learning Model and Deep Neural Network

Table of Contents

2.1	Web Application Firewall	6
2.2	Các toán tử trong xử lý hình ảnh	12
2.3	Thư viện và công cụ	15

2.1 Web Application Firewall

2.1.1 What is WAF ?

WAF stands for **Web Application Firewall**. This firewall solution commonly monitors data packets and filters them for the presence of malware or viruses. It performs the data monitoring/filtering for to and from data packets.

The WAF tool can be distributed using network-based, cloud-based, or host-based architectures. It needs a reverse proxy to make sure that one or more web apps are in front of it while facing forward.

It can be utilized either alone or in conjunction with other applications. WAF may function at a lower level or a higher level depending on the requirement ¹.

2.1.2 How does WAF work?

As mentioned before, WAF is placed at the application layer and functions as a two-way protection there. At work, WAF keeps an eye on HTTP or HTTPS traffic entering or leaving a certain web app. When a malicious object is seen in the traffic, WAF activates and destroys it .

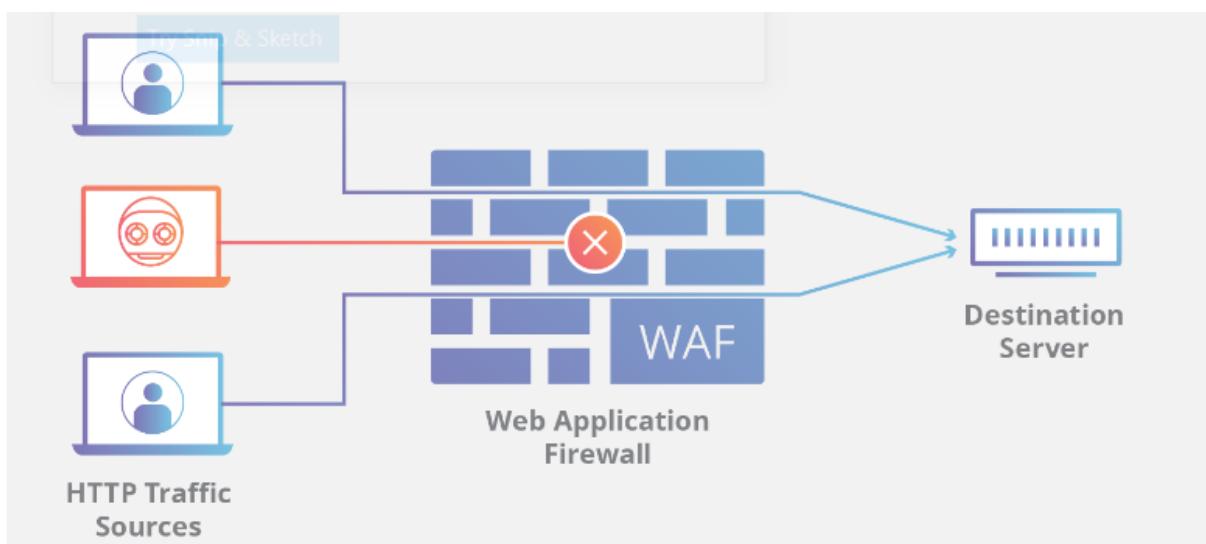


Figure 2.1: Normal users (top-left and bottom-left) are permitted access to the server with a WAF enabled, but attackers (middle-left) are prevented from doing so.

To make the process a bit more simplified, WAF predefined what is vindictive and what's not. WAF follows these rules all through the process. Mainly, WAF analyses the GET and POST part of the HTTP traffic. GET retrieves data from the server while POST is used to guide the data to the server to alter its original condition ².

1 Wallarm. *WAF Meaning*. <https://www.wallarm.com/what/waf-meaning>

2 Wallarm. *WAF Meaning*. <https://www.wallarm.com/what/waf-meaning>

2.1.3 Lớp Pooling

Thông thường, lớp pooling được chèn định kỳ giữa các lớp convolution liên tiếp trong kiến trúc mạng. Chức năng của nó là giảm dần kích thước không gian của dữ liệu để giảm số lượng tham số và lượng tính toán trong mạng, và do đó cũng có thể kiểm soát việc overfitting³. Tuy nhiên, việc sử dụng pooling cũng dẫn đến việc mất mát thông tin, điều này khiến cho mô hình trở nên nhạy cảm với các phép biến đổi trên dữ liệu đầu vào như xoay (rotate) hoặc lật hình (flip).

Lớp pooling hoạt động độc lập từng vùng trên dữ liệu đầu vào và thay đổi kích thước không gian của dữ liệu. Phép max-pooling là phép toán thường được sử dụng nhất, đặc biệt là max-pooling với filter có kích thước 2×2 và $stride = 2$. Khi áp dụng phép max-pooling này, filter sẽ trượt trên dữ liệu đầu vào và lấy giá trị lớn nhất trong mỗi vùng làm giá trị đầu ra, kích thước đầu ra sẽ giảm hai lần cho cả chiều rộng và chiều cao. Figure 2.2 mô phỏng cho phép toán này. Ngoài ra cũng có thể sử dụng các phép pooling khác như lấy giá trị trung bình hoặc chuẩn hóa $L2$.

³ Overfitting là hiện tượng mạng ghi nhớ thông tin của dữ liệu thay vì học các đặc trưng. Điều này không tốt cho quá trình sử dụng mạng sau này.

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Figure 2.2: Phép toán Max-pooling với kích thước filter 2×2 và stride 2 trên ảnh đầu vào có kích thước 4×4 cho kết quả đầu ra có kích thước 2×2 (Source: [1]).

2.1.4 Receptive field

Receptive field trong lĩnh vực thị giác máy tính được định nghĩa là vùng trên không gian đầu vào tương ứng cho một đặc trưng được trích xuất ở kết quả đầu ra. Receptive field của một đặc trưng được mô tả bằng điểm chính giữa và kích thước của nó. Trong một receptive field, điểm ảnh càng gần tâm càng đóng góp nhiều vào đặc trưng đầu ra. Điều đó có nghĩa rằng, một đặc trưng không chỉ nhìn vào một vùng riêng biệt trong ảnh đầu vào mà còn tập trung nhiều vào vùng giữa của receptive field đó.

Figure 2.3 minh họa receptive field của toán tử convolution có kích thước filter 3×3 , stride là 2 và padding “same”. Áp dụng toán tử này trên ảnh đầu vào có kích thước 5×5 cho kết quả đầu ra có kích thước 3×3 (màu xanh). Tiếp tục áp dụng toán tử này trên đầu ra thu được cho kết quả cuối cùng có kích thước 2×2 (màu cam). Vùng màu xám trong hình chính là receptive field của một đặc trưng trên kết quả cuối cùng này.

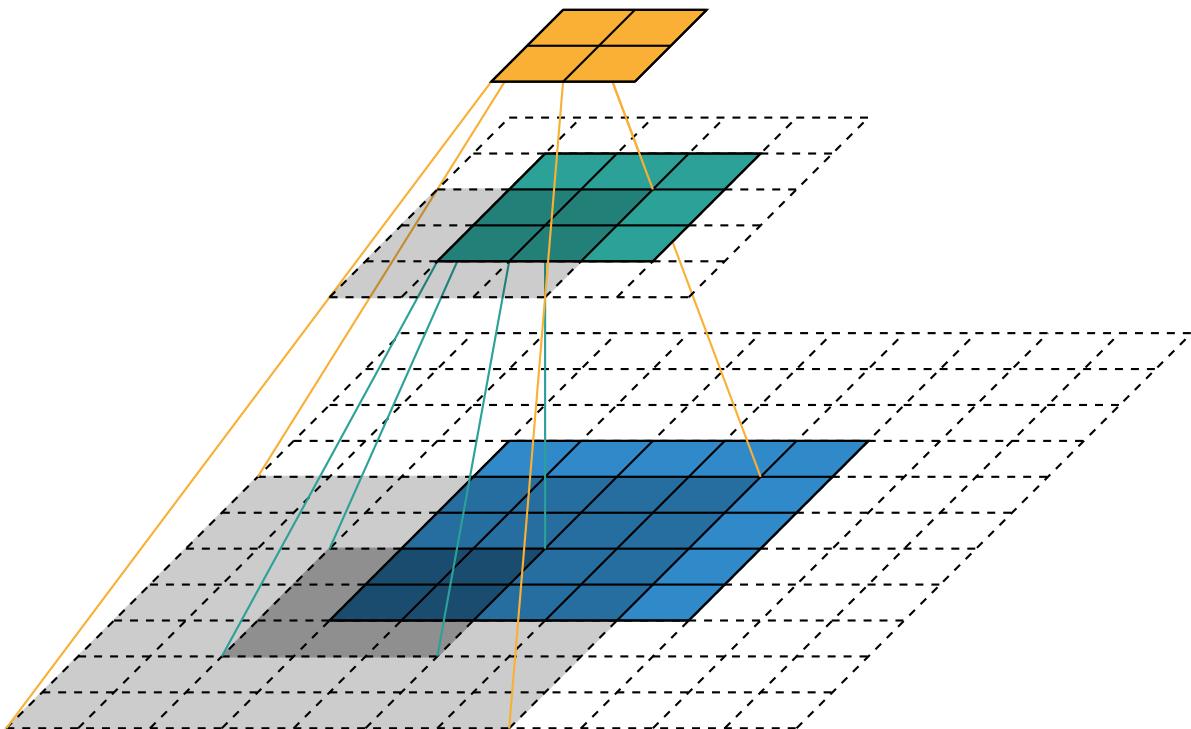


Figure 2.3: Receptive field của phép toán convolution có kích thước filter 3×3 , stride 2 và padding “same”(Source: [2]).

2.1.5 Lớp Batchnorm

Thông thường chúng ta chuẩn hoá dữ liệu đầu vào bằng cách điều chỉnh và co giãn miền giá trị của chúng để việc học không bị thiên vị về một thuộc tính bất kỳ nào của dữ liệu. Ví dụ, nếu dữ liệu của chúng ta có một thuộc tính với miền giá trị trong khoảng từ 0 đến 1 và một thuộc tính khác có miền giá trị trong khoảng từ 0 đến 1000 thì chúng ta nên thực hiện bước chuẩn hoá dữ liệu để việc huấn luyện đạt kết quả tốt hơn. Tuy nhiên việc chuẩn hoá này chỉ mới được áp dụng trên dữ liệu đầu vào. Ioffe và Szegedy [3] đã đề xuất việc chuẩn hoá dữ liệu nên được thực hiện trên cả những lớp ẩn của kiến trúc mạng.

Dữ liệu sẽ được chuẩn hoá nhờ lớp batchnorm trước khi đi qua một lớp mạng, giúp việc học trên từng lớp mạng có tính độc lập cao hơn, ít bị phụ thuộc bởi giá trị đầu ra của lớp mạng phía trước như trước đây. Đồng thời, việc sử dụng batchnorm cho phép chúng ta sử dụng giá trị learning rate (tốc độ học) lớn hơn vì lớp batchnorm đảm bảo dữ liệu đầu vào của một lớp mạng không quá cao hoặc quá thấp.

Lớp batchnorm thực hiện chuẩn hoá dữ liệu đầu ra của một lớp mạng bằng cách lấy giá trị đầu ra trừ đi giá trị trung bình (mean) rồi chia cho độ lệch chuẩn (standard deviation) của chính nó. Chi tiết về giải thuật batchnorm được mô tả trong Figure 2.4.

Input:	Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;
	Parameters to be learned: γ, β
Output:	$\{y_i = BN_{\gamma, \beta}(x_i)\}$
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$	//mini-batch mean
$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$	//mini-batch variance
$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$	// normalize
$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma, \beta}(x_i)$	// scale and shift

Figure 2.4: Giải thuật Batchnorm áp dụng lên dữ liệu đầu vào của một mini-batch trong quá trình huấn luyện (Source: [3]).

2.1.6 Lớp Sigmoid

Trong toán học, hàm sigmoid với phương trình toán học như sau

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.1)$$

là một hàm số biến một giá trị thực bất kỳ thành một giá trị thực trong phạm vi $(0, 1)$. Một số dương càng lớn sẽ cho ra giá trị gần với 1 và một số âm càng nhỏ sẽ cho ra giá trị gần với 0. Hàm sigmoid có thể được sử dụng sau một lớp convolution hoặc là lớp cuối cùng của kiến trúc mạng nhằm mục đích phân loại. Đồ thị biểu diễn hàm sigmoid được mô tả như Figure 2.5.

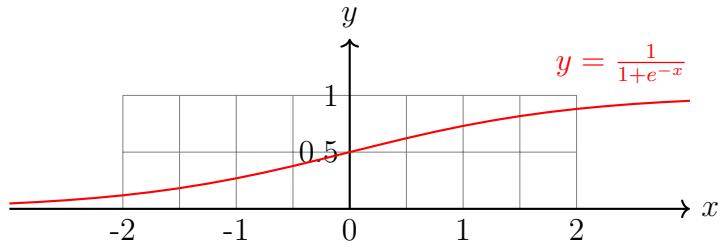


Figure 2.5: Đồ thị biểu diễn hàm sigmoid.

2.1.7 Lớp ReLU

ReLU (Rectified Linear Unit) do Mair và Hinton [4] đề xuất với công thức toán học như sau

$$ReLU(x) = \max(0, x), \quad (2.2)$$

nghĩa là hàm ReLU đặt một ngưỡng tại 0 chỉ cho những giá trị dương đi qua. So với hàm Sigmoid, hàm ReLU giúp tăng tốc độ hội tụ khi huấn luyện bằng SGD (Stochastic Gradient Descent). Một ưu điểm nữa của hàm ReLU là trong quá trình lập trình, hàm ReLU có thể hiện thực dễ dàng bằng cách áp một mặt nạ với ngưỡng bằng 0 cho cả quá trình lan truyền xuôi và lan truyền ngược. Tuy nhiên, chính việc áp mặt nạ làm cho gradient bằng 0 tại một số điểm trong mạng dẫn đến gradient từ điểm đó về trước trong quá trình lan truyền ngược bằng 0 và mạng không học được. Đồ thị biểu diễn hàm ReLU như Figure 2.6.

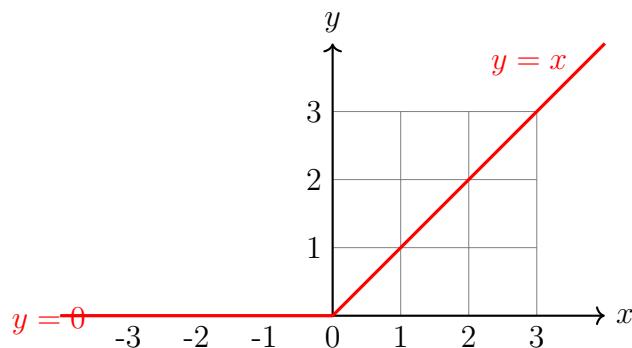


Figure 2.6: Đồ thị biểu diễn hàm ReLU.

2.1.8 Lớp Softmax

Trong toán học, hàm softmax (hay hàm trung bình mũ) là một hàm số biến không gian K chiều với giá trị thực bất kỳ đến không gian K chiều mang giá trị trong phạm vi $(0, 1)$. Phương trình toán học của hàm softmax được biểu diễn như sau

$$a_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}, \quad (2.3)$$

trong đó, z_i là giá trị đầu vào, K là số khả năng khác nhau có thể xảy ra và a_i là giá trị đầu ra của hàm softmax. Hàm softmax là một hàm đồng biến đảm bảo nếu giá trị z_i càng lớn thì xác suất rơi vào khả năng thứ i trong K khả năng càng cao. Đồng thời, hàm này đảm bảo các giá trị đầu ra a_i dương và tổng của chúng bằng 1. Figure 2.7 mô tả một số giá trị đầu vào và đầu ra tương ứng của hàm softmax.

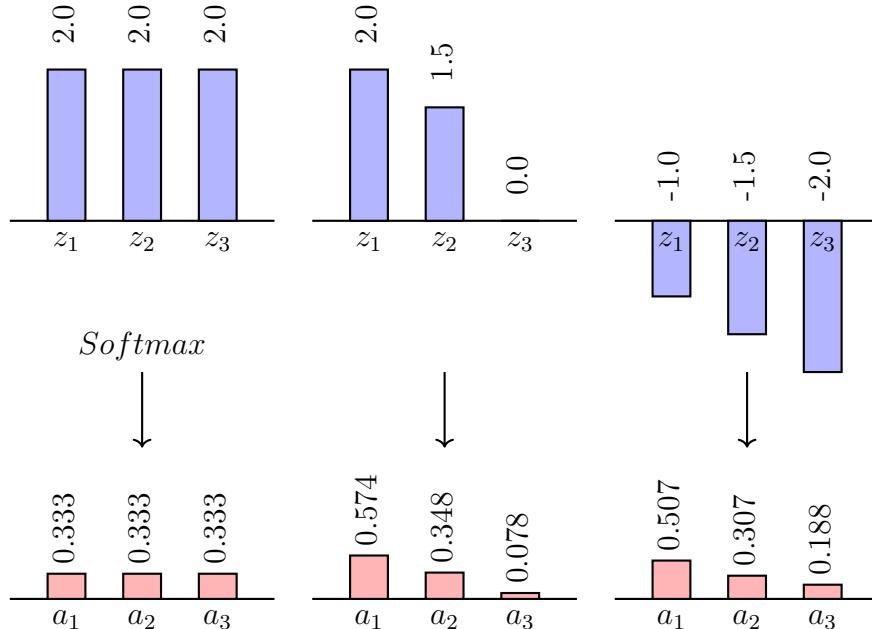


Figure 2.7: Một số ví dụ về đầu vào và đầu ra của hàm Softmax (Source: [5]).

Trong lý thuyết xác suất, giá trị xuất ra của hàm softmax có thể được sử dụng để đại diện cho một loại phân phối – đó là phân phối xác suất trên K khả năng khác nhau có thể xảy ra.

Hàm softmax được sử dụng trong nhiều phương pháp phân loại đa lớp như hồi quy logistic đa biến, biệt thức tuyến tính phân tích nhiều lớp và mạng nơ-ron. Đặc biệt, trong mạng nơ-ron, lớp softmax được sử dụng làm lớp cuối cùng của kiến trúc mạng để xác định độ chắc chắn trong kết quả dự đoán.

2.2 Các toán tử trong xử lý hình ảnh

Trong mục này, chúng tôi trình bày ba phép toán trong xử lý hình ảnh bao gồm phép toán giãn nở đối tượng dilation, phép toán làm co đối tượng erosion và phép toán trích xuất khung xương đối tượng skeleton. Trong đó, phép toán dilation và phép toán skeleton được chúng tôi sử dụng trong luận văn này.

2.2.1 Phép toán Dilatation

Phép toán giãn nở đối tượng dilation (thường được biểu diễn bởi ký hiệu \oplus) là một trong các toán tử cơ bản thuộc miền toán học hình thái (mathematical morphology) được Weeks [6] trình bày trong cuốn *Fundamentals of electronic image processing*. Toán tử dilation có tác dụng làm tăng kích thước đối tượng, nối đứt đoạn và lấp lỗ trống. Ban đầu toán tử này được phát triển để sử dụng trên ảnh nhị phân, về sau được mở rộng và áp dụng được cho cả ảnh xám.

Trong các phép toán biến đổi hình thái, phép toán dilation thuộc loại toán tử shift-invariant¹. Nó thường sử dụng một thành phần cấu trúc (structuring element) để thăm dò và mở rộng đối tượng chứa trong ảnh đầu vào. Gọi A là tập các điểm thuộc các đối tượng trong ảnh nhị phân, B là một thành phần cấu trúc, khi đó phép toán dilation trên A áp dụng B được định nghĩa là

$$A \oplus B = \bigcup_{x \in I^2} \{x | \hat{B}_x \cap A \neq \emptyset\}, \quad (2.4)$$

trong đó, I^2 là miền không gian hai chiều của ảnh nhị phân, \hat{B}_x là một tịnh tiến của thành phần cấu trúc đối xứng qua gốc toạ độ của B theo vec-tơ x thuộc miền I^2 . Formula 2.4 có thể được hiểu như sau. Đầu tiên đối xứng thành phần cấu trúc B qua gốc toạ độ ta được thành phần cấu trúc mới \hat{B} , sau đó tịnh tiến \hat{B} theo một vec-tơ x bất kỳ trên ảnh đầu vào sao cho giao của nó và A khác rỗng. Lúc đó, tập tất cả các điểm x chính là kết quả của phép toán dilation.

Nếu B có tâm đặt tại gốc toạ độ, lúc đó phép toán dilation trên A áp dụng B được hiểu là quỹ tích các điểm bao phủ bởi B với tâm của B dịch chuyển bên trong A như mô phỏng ở Figure 2.8.

2.2.2 Phép toán Erosion

Phép toán làm co đối tượng erosion (thường được biểu diễn bởi ký hiệu \ominus) là toán tử cơ bản thứ hai thuộc miền toán học hình thái được giới thiệu bởi Weeks [6]. Toán tử erosion có tác dụng làm mảnh đối tượng, xóa bỏ nhiễu và chi tiết thừa. Ban đầu toán tử này được phát triển để sử dụng trên ảnh nhị phân, về sau được mở rộng và áp dụng được cho cả ảnh xám.

¹ Shift-invariant là khái niệm chỉ phép trượt một đối tượng trong không gian của nó và giữ nguyên cấu trúc của đối tượng.

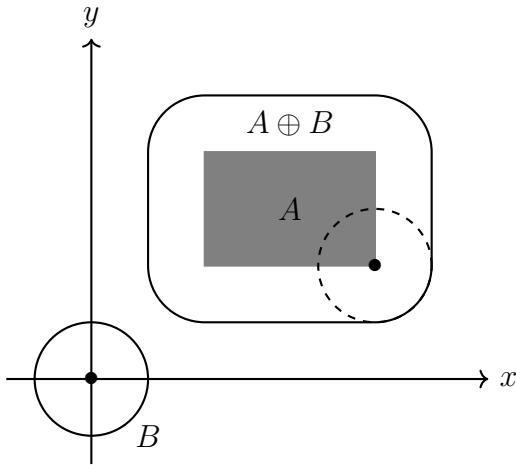


Figure 2.8: Phép toán Dilation trên A áp dụng thành phần cấu trúc B (Source: [6]).

Tương tự phép toán dilation, phép toán erosion thuộc loại toán tử shift-invariant. Nó cũng sử dụng một thành phần cấu trúc để thăm dò và thu giảm đối tượng chứa trong ảnh đầu vào. Gọi A là tập các điểm thuộc các đối tượng trong ảnh nhị phân, B là một thành phần cấu trúc, khi đó phép toán erosion trên A áp dụng B được định nghĩa là

$$A \ominus B = \bigcup_{x \in I^2} \{x | B_x \subseteq A\}, \quad (2.5)$$

trong đó, I^2 là miền không gian hai chiều của ảnh nhị phân, B_x là một tịnh tiến của B theo vec-tơ x thuộc miền I^2 . Formula 2.4 có thể được hiểu như sau. Tịnh tiến B theo một vec-tơ x bất kỳ trên ảnh đầu vào sao cho nó là một tập con hoặc bằng A . Lúc đó, tập tất cả các điểm x chính là kết quả của phép toán erosion.

Nếu B có tâm đặt tại gốc toạ độ, lúc đó phép erosion trên A áp dụng B được hiểu là quỹ tích các điểm bao phủ bởi tâm B với B dịch chuyển bên trong A như mô phỏng ở Figure 2.9.

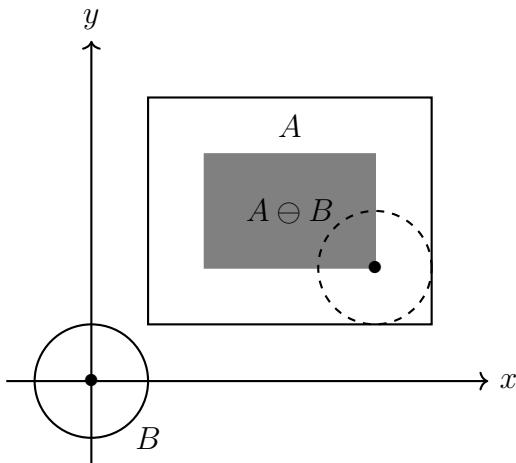


Figure 2.9: Phép toán Erosion trên A áp dụng thành phần cấu trúc B (Source: [6]).

2.2.3 Phép toán Skeleton

Phép toán trích xuất khung xương đối tượng skeleton là một trong các phép toán biến đổi hình thái được giới thiệu bởi Fisher và cộng sự [7]. Phép toán này được sử dụng để rút trích thành phần chính đại diện cho hình dạng của đối tượng trong ảnh nhị phân. Được ứng dụng trong nhận dạng mẫu (nhận dạng kí tự), nén ảnh, phát hiện lỗi trên sản phẩm công nghiệp (đứt đoạn).

Gọi $\{nB\}, n = 0, 1, \dots$ là một họ các hình khối với B là một thành phần cấu trúc, ta có

$$nB = \underbrace{B \oplus \cdots \oplus B}_{n \text{ lần}}. \quad (2.6)$$

Khi $n = 0$, $nB = \{o\}$ với o biểu diễn cho thành phần cấu trúc ban đầu. Phép toán skeleton lên ảnh nhị phân A được định nghĩa như sau

$$S(A) = \bigcup_{n=0}^N (A \ominus nB) - ((A \ominus nB) \ominus B) \oplus B, \quad (2.7)$$

trong đó, N là số lần lặp lớn nhất trước khi A trở thành tập rỗng. N được xác định theo công thức

$$N = \max\{k | (A \ominus kB) \neq \emptyset\}. \quad (2.8)$$

Figure 2.10 là một ví dụ áp dụng phép toán skeleton để rút trích khung xương đối tượng trong ảnh nhị phân.

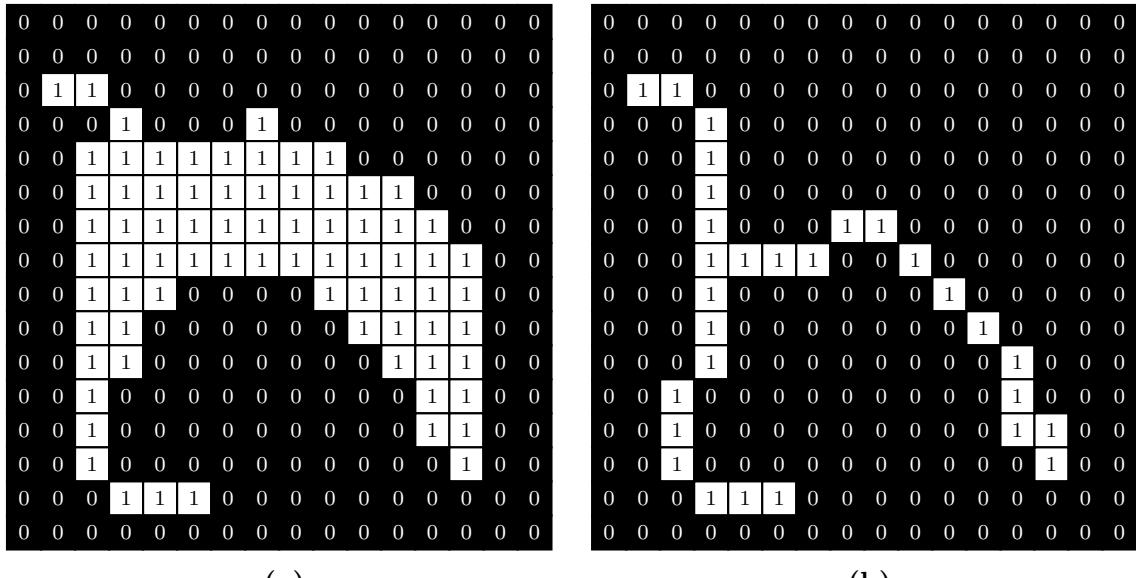


Figure 2.10: Áp dụng phép toán Skeleton để rút trích khung xương đối tượng trong không gian 2D. (a) ảnh nhị phân ban đầu. (b) ảnh nhị phân sau khi trích xuất khung xương đối tượng. (Source: [7]).

Trong luận văn này chúng tôi sử dụng kỹ thuật rút trích khung xương đối tượng trong không gian 3D được đề xuất bởi Lee và cộng sự [8] để tìm đường chính giữa của mạch máu trong ???. Figure 2.11 minh họa việc áp dụng phép toán skeleton làm mảnh đối tượng trong không gian 3D.

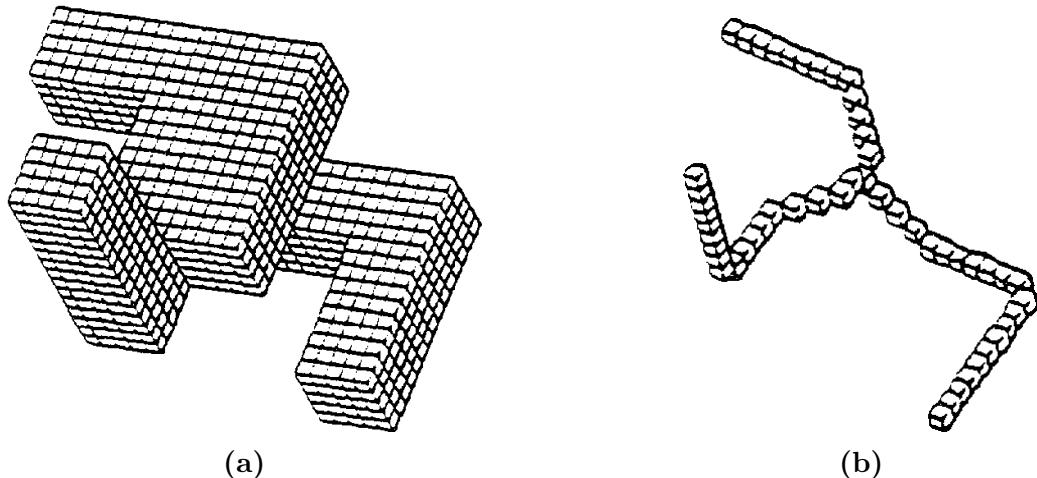


Figure 2.11: Áp dụng phép toán Skeleton để rút trích khung xương đối tượng trong không gian 3D. (a) đối tượng ban đầu. (b) khung xương được tạo ra từ phép toán skeleton. (Source: [8]).

2.3 Thư viện và công cụ

Trong mục này, chúng tôi giới thiệu các thư viện và công cụ mà chúng tôi đã sử dụng trong quá trình thực hiện Luận văn này, bao gồm nền tảng PyTorch, thư viện VTK và ứng dụng Slicer.

2.3.1 Nền tảng PyTorch

PyTorch¹ là một nền tảng mã nguồn mở được phát triển và duy trì bởi nhóm nghiên cứu về trí thông minh nhân tạo của Facebook. PyTorch có khả năng tự động ghi lại các toán tử đã xử lý vào một đồ thị tính toán trong bước forward (lên truyền xuôi) và sau đó tái thực thi các toán tử này để tính toán gradient một cách nhanh chóng ở bước backward (lên truyền ngược). Kỹ thuật này giúp PyTorch trở nên hiệu quả trong việc xây dựng một mạng nơ-ron vì nó giúp tiết kiệm thời gian nhờ vào việc tính toán đạo hàm các tham số trước khi thực hiện bước forward. Với phiên bản mới nhất của PyTorch (phiên bản 1.0.1, xem [9]), các nhà phát triển có thể chuyển từ giai đoạn nghiên cứu sang giai đoạn triển khai sản phẩm một cách dễ dàng, biến PyTorch trở thành một nền tảng học sâu vô cùng mạch mẽ. PyTorch được viết bằng ngôn ngữ C++, CUDA và Python và được tối ưu hóa hoàn toàn cho các tác vụ học sâu trên cả CPU và GPU như xử lý hình ảnh, huấn luyện các mạng học sâu với khả năng huấn luyện song song trên nhiều GPU, v.v.

Với các tính năng phong phú và sự năng động của cộng đồng hỗ trợ và phát triển, chúng tôi chọn PyTorch làm nền tảng xây dựng và phát triển hệ thống trong luận văn này.

¹ <https://pytorch.org>

2.3.2 Thư viện VTK

VTK (The Visualization Toolkit)¹ là một thư viện mã nguồn mở cung cấp cho các nhà phát triển một bộ công cụ đa dạng phục vụ cho đồ họa máy tính trong không gian ba chiều, xử lý hình ảnh và trực quan hóa. Thư viện này được hiện thực từ nhiều ngôn ngữ khác nhau như C++, Java và Python. Nó hỗ trợ nhiều giải thuật trực quan hóa bao gồm phương pháp vô hướng, phương pháp vec-tơ, phương pháp kết cấu và thể tích. Đặc biệt, thư viện này còn hỗ trợ các kỹ thuật mô hình hóa nâng cao như mô hình ẩn, thu giảm đa giác, làm mịn bề mặt, tạo đường viền và tam giác phân Delaunay.

Trong luận văn này, chúng tôi sử dụng thư viện VTK trên ngôn ngữ Python nhằm trích xuất và làm mịn lưới bề mặt hệ thống mạch máu cũng như đường chính giữa mạch máu và các điểm phân nhánh từ kết quả dự đoán, phục vụ quá trình trực quan hóa sẽ được đề cập tới trong Sector 6.5.

2.3.3 Phần mềm Slicer

Slicer² là ứng dụng mã nguồn mở phục vụ cho tin học về hình ảnh y tế, xử lý hình ảnh y khoa và trực quan hóa trong không gian ba chiều. Được xây dựng trong hơn hai thập kỷ qua với sự hỗ trợ của các viện y tế quốc gia (National Institutes of Health) và cộng đồng các nhà phát triển trên toàn thế giới, trong đó phải kể đến sự đóng góp của cộng đồng quốc tế các nhà khoa học, bao gồm cả kỹ thuật và y sinh. Slicer mang đến các công cụ xử lý đa nền tảng miễn phí, mạnh mẽ cho các bác sĩ, các nhà nghiên cứu và cộng đồng.

Trong luận văn này, chúng tôi sử dụng ứng dụng Slicer để trực quan hóa kết quả thu được từ hệ thống, nhằm đưa ra những nhận xét mang tính định tính chất lượng, giúp kết quả đánh giá được toàn diện hơn. Chúng tôi sẽ đề cập rõ hơn trong Sector 6.5.

¹ <https://pypi.org/project/vtk/>

² <https://www.slicer.org/>

3

LITERATURE REVIEW

mentions some related studies on the use of machine techniques to detect malicious requests, their strengths and weaknesses and evaluate an appropriate approach for this thesis.

Table of Contents

Malicious request detection has been a focus of recent study, with several malicious request detection systems developed. They propose solutions for a particular or different types of harmful requests. For example, reference [10] show us their solution to SQLI attack that is one of the most common malicious requests. Joshi and Geetha [10] detect SQLI attacks using the Naive Bayes machine learning algorithm. Or [11] automatically generates malicious requests containing sql injection to exploiting flaws and taking over of database servers. Reference [12] train a Support Vector Machine(SVM) model to detect SQLI attacks.

Fawaz Mereani and Jacob Howe et al[13] has demonstrated that SVM, k-NN, and Random Forest can be used to build classifiers for XSS coded in JavaScript giving high accuracy (up to 99.75%) and precision (up to 99.88%) when applied to a large real world dataset. One especially intriguing component of this study is that, unlike other studies, a binary measure was employed for all features. This has resulted in greater accuracy and precision than previous tests employing weighted measurements.

Laughter et al [14] integrated the http request features in the process of visiting the website into the detection feature set. By extracting the content of each field in the request header and request body, classification methods such as decision tree and SVM were used to complete the research.

Alshammari, Amirah and Aldribi, Abdulaziz et al [15] present a reliable model running in Real-time to detect malicious data flow traffic on cloud depending on the ML supervised techniques based on the ISOT-CID dataset that contains network traffic data features. When tested using cross-validation and split-validation, DTREE and Random Forest both produced the best accuracy results. Their two models did not fail in any of the classification processes used to assess different portions or folds of the dataset.

This group of researcher [16] proposed an approach to detect malicious URL using ensemble learning. They use TF-IDF to pre-processed input URLs, then applied Rain Forest Ensemble-Based for prediction and an artificial neural network (ANN) classifier was constructed for decision making. Results show that this approach significantly improved the detection performance, achieving 96.80% compared with the best 90.4% achieved by the URL-based features. The false-positive rate was significantly decreased to 3.1% compared with 12% performed by the URL-based model.

Khoi. Le et al [17] suggested an approach of extracting WAF rules and trained a machine learning with the decision model totally independent from the rules itself. This makes the model more self-reliant and the overall result more neutral. Our module is inspired by this approach.

4

DATASET

Trong chương này, chúng tôi giới thiệu tập dữ liệu 3D-IRCADb-01 mà chúng tôi đã sử dụng để đánh giá hệ thống. Sau đó, chúng tôi trình bày các vấn đề cần xử lý đối với tập dữ liệu này trước khi có thể sử dụng.

Table of Contents

4.1	Giới thiệu chung	20
4.2	Các vấn đề cần xử lý	21

4.1 Giới thiệu chung



Figure 4.1: Logo Viện nghiên cứu chống ung thư đường tiêu hoá ở Pháp.
 (Source: <https://www.irccad.fr/>.)

3D-IRCADb (3D Image Reconstruction for Comparison of Algorithm Database) là tập dữ liệu do Viện nghiên cứu chống ung thư đường tiêu hoá *IRCAD* cung cấp và là tập dữ liệu đáp ứng tốt nhất cho bài toán trong đề tài của chúng tôi. Tập dữ liệu này cung cấp một số bộ ảnh y khoa từ các bệnh nhân được ẩn danh. Với mỗi bộ ảnh, các cơ quan, cấu trúc khác nhau được phân đoạn thủ công bởi các chuyên gia lâm sàng.

3D-IRCADb có hai gói dữ liệu là *3D-IRCADb-01* và *3D-IRCADb-02*. Chúng tôi chọn sử dụng gói *3D-IRCADb-01* [18] để đánh giá hệ thống trong luận văn này. *3D-IRCADb-01* bao gồm ảnh CT của 20 bệnh nhân (10 nam và 10 nữ) với sự xuất hiện của khối u gan trong 75% các trường hợp. Hệ thống mạch máu trong các bộ ảnh CT được chia thành ba phần là tĩnh mạch chủ, tĩnh mạch cửa và động mạch và được phân đoạn riêng cho từng loại mạch máu. Figure 4.2 là ảnh trực quan dữ liệu cho từng bệnh nhân trong gói *3D-IRCADb-01*. Các thông số chi tiết về gói dữ liệu này được chúng tôi trình bày trong Appendix A.

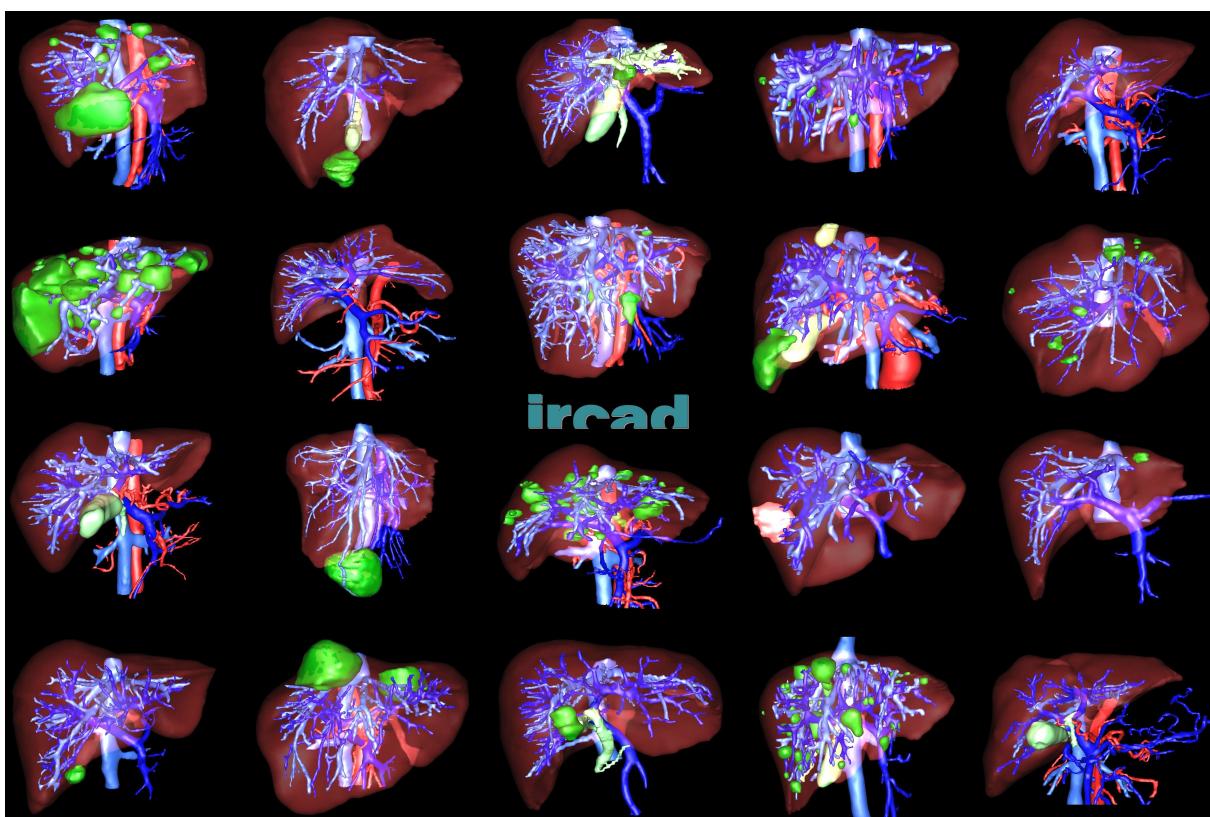


Figure 4.2: Ảnh trực quan gói dữ liệu *3D-IRCADb-01* (Source: [18]).

4.2 Các vấn đề cần xử lý

Trong mục này, chúng tôi trình bày các vấn đề gặp phải đối với tập dữ liệu cần được chú ý và (hoặc) xử lý trước khi có thể sử dụng bao gồm đơn vị hounsfield, nhãn phân đoạn không đầy đủ và giá trị nhãn không nhất quán.

4.2.1 Đơn vị Hounsfield

Nếu trong khoa học máy tính giá trị của một điểm ảnh được đo bằng cường độ sáng (intensity), thì trong y khoa giá trị của một điểm ảnh trên ảnh CT được đo bằng đơn vị hounsfield (xem [19]). Cách tính giá trị hounsfield như sau

$$HU = m * P + b, \quad (4.1)$$

trong đó, HU là giá trị hounsfield, P là giá trị số của điểm ảnh, m là giá trị tại trường (0028,1053) “Rescale slope” và b là giá trị tại trường (0028,1052) “Rescale intercept” được lưu trong file DICOM¹. Đối với bộ dữ liệu *3D-IRCADb-01*, m và b đều có giá trị là 1 nên ta có thể sử dụng trực tiếp không phải thực hiện bước tính giá trị hounsfield.

4.2.2 Nhãn phân đoạn không đầy đủ

Trong đề tài này, chúng tôi quan tâm các thành phần trong tập dữ liệu liên quan đến cơ quan gan và hệ thống mạch máu. Tuy nhiên, sau khi khảo sát tình trạng của tập dữ liệu, chúng tôi phát hiện tập dữ liệu có hai vấn đề về tính đầy đủ của nhãn phân đoạn.

Thứ nhất, nhãn phân đoạn không đầy đủ cho các loại mạch máu. Khi tìm kiếm các tệp tin liên quan đến hệ thống mạch máu trong tập dữ liệu, chúng tôi phát hiện có rất nhiều bệnh nhân không có nhãn phân đoạn cho động mạch.

Thứ hai, nhãn phân đoạn không đầy đủ trên tất cả các lớp ảnh. Figure 4.3 là ví dụ về việc gán nhãn không đầy đủ trên các lớp ảnh cho tĩnh mạch ở bệnh nhân số 2. Figure 4.3a và Figure 4.3b lần lượt là lớp ảnh CT thứ 74 chứa tĩnh mạch và nhãn phân đoạn tương ứng bị gán thiếu. Trong khi Figure 4.3c là lớp ảnh CT liền kề với nhãn phân đoạn tĩnh mạch đầy đủ như Figure 4.3d.

Tables 4.1 tổng hợp tình trạng nhãn phân đoạn của cơ quan gan và hệ thống mạch máu bao gồm tĩnh mạch chủ, tĩnh mạch cửa và động mạch. Ở mỗi đối tượng, bảng cho biết nhãn phân đoạn tương ứng có được cung cấp hay không và những lớp ảnh nào trong tổng số các lớp ảnh được gán nhãn.

1 DICOM là định dạng chuyên dụng cho ảnh CT.

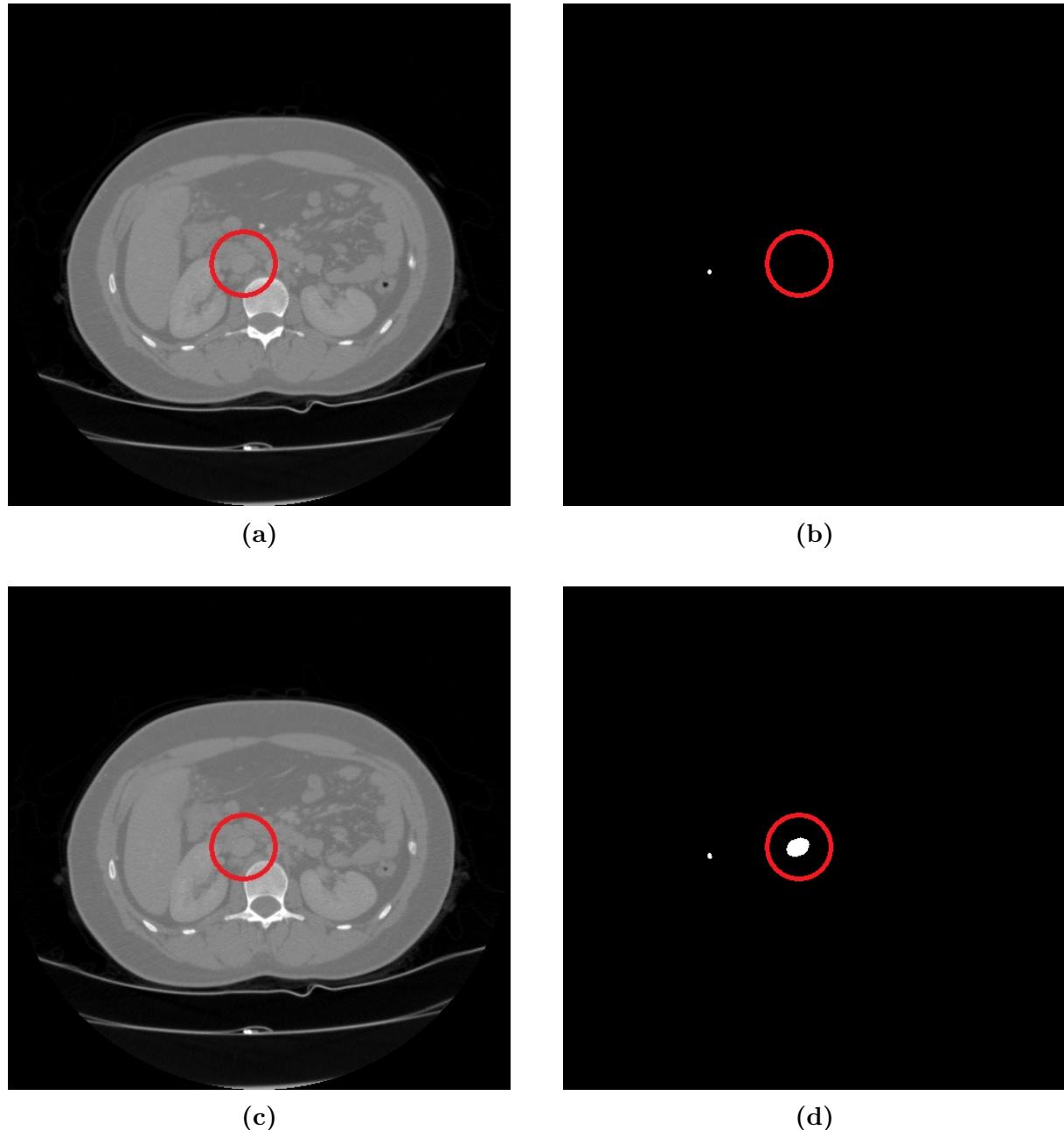


Figure 4.3: Nhãn phân đoạn bị thiếu trên một số lớp ảnh ở bệnh nhân số 2. (a) ảnh CT lớp thứ 74 có tĩnh mạch chủ. (b) nhãn phân đoạn cho ảnh CT lớp thứ 74 không được đánh nhãn tĩnh mạch chủ. (c) ảnh CT lớp thứ 75 có tĩnh mạch chủ. (d) nhãn phân đoạn cho ảnh CT lớp thứ 75 được đánh nhãn tĩnh mạch chủ.

Tables 4.1: Tình trạng nhãn phân đoạn của cơ quan gan và hệ thống mạch máu bao gồm tĩnh mạch chủ, tĩnh mạch cửa và động mạch. Kết quả khảo sát ở mỗi đối tượng cho biết nhãn phân đoạn của đối tượng đó có được cung cấp hay không và danh sách các lớp ảnh được gán nhãn.

STT	Số lớp ảnh	Gan		Tĩnh mạch chủ		Tĩnh mạch cửa		Động mạch	
		Tồn tại	Được gán nhãn	Tồn tại	Được gán nhãn	Tồn tại	Được gán nhãn	Tồn tại	Được gán nhãn
1	129	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
2	172	Có	Đủ	Có	75-152	Có	75-152	Không	—
3	200	Có	Đủ	Có	104-182	Có	104-182	Không	—
4	91	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
5	139	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
6	135	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
7	151	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
8	124	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
9	111	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
10	122	Có	Đủ	Có	49-113	Có	49-113	Không	—
11	132	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
12	260	Có	Đủ	Có	65-260	Có	65-260	Có	66-236
13	122	Có	Đủ	Có	1-117	Có	1-117	Có	Đủ
14	113	Có	Đủ	Có	51-113	Có	51-113	Không	—
15	125	Có	Đủ	Có	62-123	Có	62-123	Không	—
16	155	Có	Đủ	Có	32-143	Có	32-143	Không	—
17	119	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ
18	74	Có	Đủ	Có	25-69	Có	25-69	Không	—
19	124	Có	Đủ	Có	Đủ	Có	Đủ	Không	—
20	225	Có	Đủ	Có	Đủ	Có	Đủ	Có	Đủ

4.2.3 Giá trị nhãn không nhất quán

Vấn đề tiếp theo của tập dữ liệu là giá trị nhãn phân đoạn không nhất quán. Đây là vấn đề nghiêm trọng mà nếu không xử lý trước khi sử dụng thì việc huấn luyện sẽ không thể thành công.

Trong hầu hết các nhãn phân đoạn có trong bộ dữ liệu, các giá trị background và foreground được gán số lần lượt là 0 và 255. Tuy nhiên, rất nhiều nhãn phân đoạn ở các cơ quan ở các bệnh nhân khác nhau được đánh giá trị khác nhau. Figure 4.4 là ví dụ việc đọc lên khối nhãn phân đoạn tĩnh mạch chủ của bệnh nhân số 1 và in ra các giá trị riêng biệt, kết quả là tập giá trị chứa 0 và 255. Thực hiện thao tác tương tự trên bệnh nhân số 3, ta có kết quả là tập giá trị chứa 0 và 1 (Figure 4.5).

```

1 path = "3Dircadb1/3Dircadb1.1/MASKS_DICOM/venoussystem/"
2
3 # Load all dicom files to numpy array.
4 volume = []
5 for i in range(len(os.listdir(path))):
6     dicom = pydicom.dcmread(path + "image_" + str(i))
7     volume.append(dicom.pixel_array)
8 volume = numpy.asarray(volume)
9
10 # Print unique value in volume.
11 numpy.unique(volume)

array([ 0, 255], dtype=uint8)

```

Figure 4.4: Giá trị nhãn phân đoạn tĩnh mạch chủ trên bệnh nhân số 1

```

1 path = "3Dircadb1/3Dircadb1.3/MASKS_DICOM/venoussystem/"
2
3 # Load all dicom files to numpy array.
4 volume = []
5 for i in range(len(os.listdir(path))):
6     dicom = pydicom.dcmread(path + "image_" + str(i))
7     volume.append(dicom.pixel_array)
8 volume = numpy.asarray(volume)
9
10 # Print unique value in volume.
11 numpy.unique(volume)

array([ 0, 1], dtype=uint8)

```

Figure 4.5: Giá trị nhãn phân đoạn tĩnh mạch chủ trên bệnh nhân số 3

Tables 4.2 tổng hợp các giá trị background và foreground cho từng nhãn phân đoạn ở các cơ quan gan, tĩnh mạch chủ, tĩnh mạch cửa và động mạch. Từ đây, chúng tôi tiến hành chuẩn hoá giá trị nhãn phân đoạn về chung một bộ giá trị với background và foreground lần lượt là 0 và 1.

Tables 4.2: Giá trị nhãn phân đoạn của cơ quan gan và hệ thống mạch máu.

STT	Gan		Tĩnh mạch chủ		Tĩnh mạch cửa		Động mạch	
	BG	FG	BG	FG	BG	FG	BG	FG
1	0	255	0	255	0	255	0	255
2	0	1	0	255	0	255	—	—
3	0	255	0	1	0	1	—	—
4	0	255	0	255	0	255	0	255
5	0	255	0	255	0	255	0	255
6	0	255	0	255	0	255	0	255
7	0	255	0	255	0	255	0	255
8	0	255	0	255	0	255	0	255
9	0	255	0	255	0	255	0	255
10	0	255	0	255	0	255	—	—
11	0	255	0	255	0	255	0	1, 255
12	0	1, 255	0	1	0	1	0	1
13	0	255	0	255	0	255	0	255
14	0	255	0	255	0	1, 255	—	—
15	0	255	0	255	0	255	—	—
16	0	255	0	255	0	255	—	—
17	0	255	0	255	0	255	0	255
18	0	1	0	255	0	255	—	—
19	0	255	0	255	0	255	—	—
20	0	1	0	1, 255	0	255	0	1, 255

5

PROPOSED APPROACH

Trong chương này, chúng tôi mô tả chi tiết bài toán trước khi bắt tay vào hiện thực. Sau đó, chúng tôi đề xuất kiến trúc mạng và các hàm lõi sẽ sử dụng. Cuối cùng, chúng tôi đề xuất phương pháp tìm đường chính giữa và điểm phân nhánh của mạch máu.

Table of Contents

5.1 API security problems	28
5.2 Designs	34

WAFs, as previously indicated, are frequently used, however they suffer from high FP. Our aim is to enhance the accuracy of WAFs, using the help of machine learning rather than the rule-based approaches.

Because WAFs only cover the Application Layer, the network requests are the model's input (mostly HTTP requests). Our methodology produces the same result as WAFs: whether the request is malicious or not.

With its nature, WAF is obligated to have fast processing speed in deciding whether an incoming request is reliable or not. For user experiences, we can't examine the request's veracity for minutes before granting or denying access. Our system's time constraint must be in milliseconds.

5.1 API security problems

Here are the most common vulnerabilities¹ :

- **Lack of rate limiting, DoS and brute force attacks on APIs**

- *Principle and functioning of DoS attacks*

An assault known as **a denial of service (DoS)** aims to render services unavailable. In fact, a DoS attack functions by depleting a resource that an API requires to respond to valid requests. By overloading an API with erroneous requests, its resources are only able to reply to the ones that were submitted.

The objective of DoS attacks is not to alter, delete or steal data. The aim is simply to damage the operation of a web service or the reputation of a company offering such services.

It is obvious that slowing down or even blocking their services for a few minutes could lead to significant financial losses and alter users' trust. It is therefore necessary to find solutions to protect against this, including request verification, traffic monitoring, the implementation of rules and rate limiting, etc. Similarly, during an API or web application pentest, DoS tests should be included to assess the resistance of the services to this type of attack.

- *Brute force attacks on APIs*

In a brute force attack, an attacker use tools to send a steady stream of requests to an application or API in an effort to try every combination of a parameter in the hopes of making the right guesses. The goals can vary: brute-forcing an authentication form to steal an account, brute-forcing a login to retrieve private information, etc.

This is a “trivial” attack method, easy to perform, but still very effective and widely used by attackers.

- *Implement rate limiting mechanisms to counter DoS and brute force attacks*

Securing APIs against DoS or brute force attacks requires the implementation

¹ Vaadata. *How to strengthen the security of your APIs to counter the most common attacks?*. April 2022. <https://www.vaadata.com/blog/how-to-strengthen-the-security-of-your-apis-to-counter-the-most-common-attacks/>

of rate limiting mechanisms. These mechanisms protect APIs and other services from excessive and abusive use, in order to ensure their availability.

Rate restriction works on a pretty straightforward principle. It entails making requests in advance of when one or more clients—systems—might utilize more than their "fair" share of a resource. Additionally, one can lessen the danger of DoS or brute force attacks by restricting the number of requests that a specific user is permitted to send in a given window of time.

For example, after a user has been authenticated, your API or application may apply quotas that restrict what they are allowed to do, including the limit of requests they can send. For example, you can limit each user to a certain number of API requests per hour, to prevent them from flooding the system with too many requests.

Similar to this, you can set limitations before authenticating a user to lower the overall number of requests, or just those coming from a specific IP address or time period. Therefore, if rate limitation is enabled, your API will monitor the volume of requests and reject those that are greater than the permitted threshold. Additionally, rules can be applied to totally shut off connections when the limit is reached or to sluggish down request processing. This action is referred to as "throttling."

In short, Rate Limiting prevents resource depletion by managing rules and quotas. There are different techniques for applying rate limiting, each with its own specificities: Token bucket, Leaky bucket, Fixed window and Sliding window.

- **Lack of user input validation and injection attacks**

- *Code injections*

Code injection is one of the most common types of injection attacks. If attackers know the programming language used by an application or API, they can inject code through text input fields to force the web server to execute the desired instructions.

- *SQL injections (SQLi)*

Injections represent a significant part of the vulnerabilities encountered in applications and APIs. The best known and most dangerous is SQL injection.

In an SQL injection attack, an attacker injects data to manipulate SQL commands, thereby interacting with the database through unintended queries. These flaws can lead to theft, deletion or manipulation of stored data. Worse still: if the rights are too permissive, this can even lead to a compromise of the server.

Let's look at this in more detail with a concrete example:

One can imagine an API endpoint that returns the information of a country based on its "CountryCode".

```
URL: http://localhost:8042/?CC=FR
```

```
[  
 {  
   "name": "France",  
   "code": "FR"  
 }  
]
```

Figure 5.1: Returned information

To perform the desired action, the query interacts with a database. Below is the SQL query that the server must run on the database.

```
SELECT name_fr as name , alpha2 as code FROM Country WHERE alpha2= "FR"
```

Figure 5.2: Compulsory query

Example of a vulnerable PHP code:

```
<?php

try{
    $db = new PDO('sqlite:database/base.sqlite3');
    $db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); // 
ERRMODE_WARNING | ERRMODE_EXCEPTION | ERRMODE_SILENT
} catch(Exception $e) {
    echo "Unable to access the SQLite database:". $e->getMessage();
    die();
}

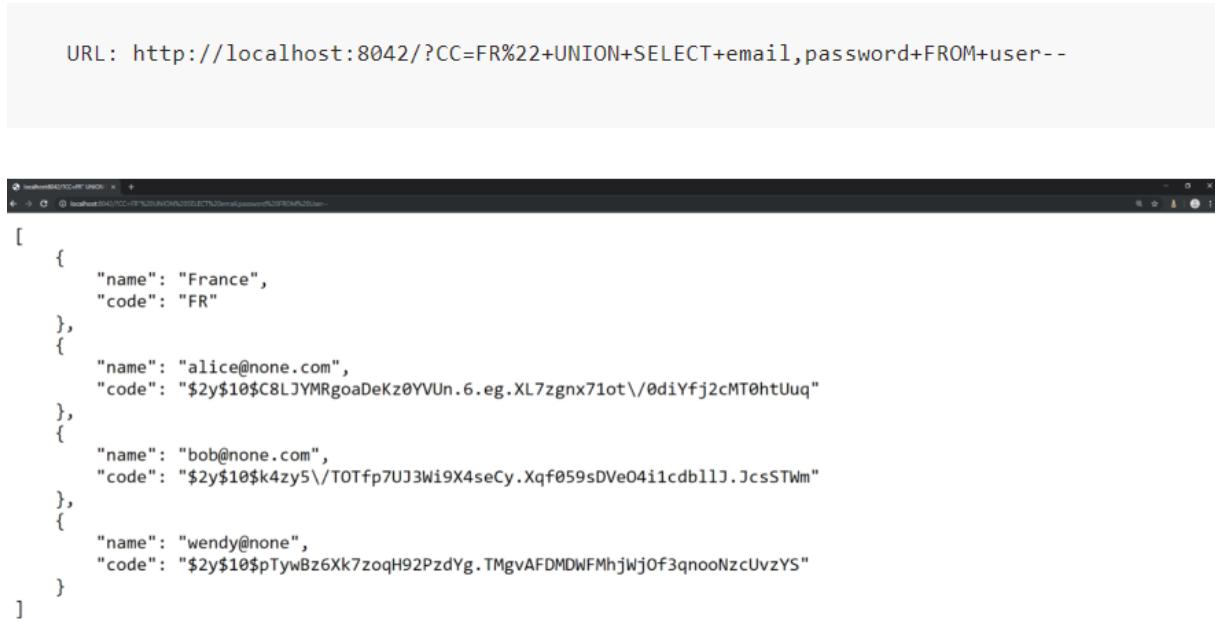
$recipesStatement = $db->prepare('SELECT name_fr as name ,alpha2 as code FROM
Country where alpha2 = "'.$_GET['CC'].'"');
$recipesStatement->execute();
$recipes = $recipesStatement->fetchAll();

header('Content-Type: application/json; charset=utf-8');
echo json_encode($recipes, JSON_PRETTY_PRINT);
```

Figure 5.3: Vulnerable PHP Code

As we can see, the CC parameter that is controlled by the user is directly concatenated to the query.

Now, let's assume that on this same database, there is a 'users' table that contains the email addresses and passwords of the users registered on the application. Let's look at what happens if an attacker makes the following query:



The screenshot shows a browser window with the URL `http://localhost:8042/?CC=FR%22+UNION+SELECT+email,password+FROM+user--`. The page displays a JSON array containing four objects. Each object has two properties: 'name' and 'code'. The first object represents France ('name': 'France', 'code': 'FR'). The subsequent three objects represent users ('name': 'alice@none.com', 'code': '...'), ('name': 'bob@none.com', 'code': '...'), and ('name': 'wendy@none', 'code': '...'). The 'code' values for the users are extremely long and complex, likely generated by a hash function.

```
[
  {
    "name": "France",
    "code": "FR"
  },
  {
    "name": "alice@none.com",
    "code": "$2y$10$c8LjYMRgoaDeKz0YVUn.6,eg.XL7zgnx71ot/0diYfj2cMT0htUuq"
  },
  {
    "name": "bob@none.com",
    "code": "$2y$10$kt4zy5/0Tfp7UJ3Wi9X4seCy.Xqf059sDVe04i1cdb11J.JcsSTWm"
  },
  {
    "name": "wendy@none",
    "code": "$2y$10$pTywBz6Xk7zoqH92PzdYg.TMgvAFDMDFMhjWj0f3qnooNzcUvzYS"
  }
]
```

Figure 5.4: The hacker's query and result

In addition to the country name, this query retrieves all users and their password hashes.

This SQL injection flaw was discovered. A flaw that enables an attacker to "pervert" the SQL query the program generates. The attacker may be able to view or even change the database's data with this behavior. Evidently, if an API or online application penetration test were conducted, this significant vulnerability would be discovered and disclosed.

- *Validate user input to prevent injection attacks*

The strongest defense against SQL and code injections is validating user input. In theory, it should be recognized that data received by an application or API cannot be deemed "always" safe. In order to prevent such vulnerabilities, it is necessary to put in place methods to verify that user input meets the anticipated parameters.

The most effective method of protecting against SQL injections is the use of prepared statements, which separate the SQL commands from the data sent by a user.

The fix is to use prepared queries. On the PHP documentation we can see the following information:

- * The query can be run multiple times with the same or different parameters after only one analysis (or preparation). The database will parse, construct, and optimize its plan to execute the query once it is prepared. If you have to repeat the same query multiple times with various parameters, this procedure can be very time-consuming for sophisticated queries, which will slow down your apps. You can avoid repeating the cycle of analysis, compilation, and optimization by using prepared statements. In short, prepared queries execute more quickly and with fewer resources.
- * You don't need to surround query parameters in quotes; the driver takes care of it. If your application only employs prepared statements, you can

be certain that SQL injection is not a possibility (however, if you build other parts of the statement based on user input, you are still taking a risk).

Thus, the following code is correctly protected against SQL injections:

```
<?php

try{
    $db = new PDO('sqlite:database/base.sqlite3');
    $db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); // 
    ERRMODE_WARNING | ERRMODE_EXCEPTION | ERRMODE_SILENT
} catch(Exception $e) {
    echo "Unable to access the SQLite database:". $e->getMessage();
    die();
}

$recipesStatement = $db->prepare('SELECT name_fr as name ,alpha2 as code FROM
Country where alpha2 = :CC');
$recipesStatement->execute([
    'CC' => $_GET['CC']
]);

$recipesStatement->execute();
$recipes = $recipesStatement->fetchAll();
header('Content-Type: application/json; charset=utf-8');
echo json_encode($recipes, JSON_PRETTY_PRINT);
```

Figure 5.5: The code is protected against SQL injections

As we can see, the difference with the vulnerable code is that the parameters coming from a user are no longer concatenated with the query, but directly provided at query execution. This also shows that a prepared statement can still be vulnerable to SQL injection if the data is concatenated. Because in the vulnerable example it was already a prepared statement that was used.

- **Lack of data encryption and Man In The Middle attacks**

- *Man In The Middle attacks*

An attack known as a "Man in the Middle" occurs when a hostile person intrudes on a communication or data transfer taking place between a client and a server, a server and a server, or a client and a client. Its goals might vary, from merely intercepting sensitive data—such as passwords, financial information, personal information, and sensitive documents—to manipulating communication in order to, for instance, implant malware.

This type of attack is possible if and only if the communications are not encrypted. It is therefore quite easy to protect against it.

- *Encrypting data with TLS to counter Man In The Middle attacks*

One of the most fundamental components of guaranteeing the security of an API or service is encryption. Indeed, the encryption protocol TLS (the successor to SSL) ensures safe communications over a computer network. Connections

between a client and a server that are TLS-secured include one or more of the following characteristics:

- * The connection is private (i.e. secure) because the data transmitted is encrypted.
- * The encryption keys are uniquely generated for each connection and are based on a shared secret negotiated at the beginning of the session.
- * The connection guarantees integrity because each transmitted message includes a signature verification of the integrity of the message, thus avoiding any undetected loss or alteration of the data during transmission.

The use of this encryption protocol therefore reduces or even eliminates the risks of Man In The Middle attacks. Furthermore, to reinforce security, we recommend implementing the HSTS (http Strict Transport Security) header on your servers in order to force a browser to use HTTPS secure connections. Without this setting, you run the risk of users accessing your domain without the HTTPS protocol, which can lead to a breach in communications.

5.2 Designs

To summary, numerous strategies for identifying malicious URLs have been proposed. Most of these solutions utilize supervised-based machine learning techniques for classification. This technique can detect irregularities between requests, but it cannot extract these abnormalities into human-readable form in order to reconstruct the WAF. Relying solely on machine learning even though brought higher accuracy like in [16], WAFs need to be time-efficient. We can not compromise accuracy for speed, hence these methods don't work for WAFs.

We come to the approach is to categorize the incoming request and analyze its structure, with moderate reliability, then combining the result of these two process to achieve the high precision yet does not expending an excessive amount of time

5.2.1 Architecture

5.2.1.1 Data exploration and Sanitization

Our goal is to classify request supplied as inputs in order to determine whether they are harmful or inoffensive. The sample of request data consists of different categories including:

- *Plain text*: request that contains data but does not trigger any machine execution, typically in the form of HTML, JSON, XML, and CSV, etc.
- *Client-side script*: request in form of programming languages or scripts that can be executed on client's machine

- *Server-side script*: request in the form of programming languages (mostly back-end programming languages like PHP, JAVA, PYTHON, and so on) that can change the behavior of the application or web
- *Shell script*: request in the form of shell scripts that can run jobs on the server or change the server's or operating system's behavior
- *SQL script*: request that in form of SQL, can be used to query data from the database.

The textual data acquired in the previous phase is going to be cleaned and standardized. To minimize feature complexity and improve classification performance, the request was preprocessed by eliminating symbols and punctuation. The text data collected will be transformed to lower case and then normalized. The goals of the normalization procedure were dual. First, the text must be converted from unstructured data to a structured word vector. Second, by deleting unneeded words and decreasing the number of words by roots them to their originals, we may reduce the scarcity of feature vectors. However, malicious attacker usually use methods like slight modification to make an imposter URL or request looks legitimate, we skip severals step in conventional Natural Language Processing:

- Removing stop word
- Stemming (the process of reducing infected words to their stem)
- Lemmatization (returning to the base form of the words...)

5.2.1.2 Model Selection and applying Natural language processing

Because four of the five categories are structured languages, we must first create a special tokenizer to map the request into a defined set of keywords (client-side script, server- side script, and SQL script). To convert the words (the tokens) to their numerical equivalents, a corpus containing a list of unique tokens based on their frequency of occurrence in each class was created. To put it in more formal mathematical terms, the TF-IDF score for the word t in the document d from the document set D is calculated as follows:

$$tf\ idf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (5.1)$$

Where:

$$tf(t, d) = 1 + \log(freq(t, d)) \quad (5.2)$$

$$idf(t, D) = \log\left(\frac{N}{count(d \in D : t \in d)}\right) \quad (5.3)$$

The statistical-based text representation, TF-IDF, was then computed using the following equation:

$$tf_idf = tf \cdot \log\left(\frac{N}{df}\right) \quad (5.4)$$

where tf is the term frequency of the word in a specific instance, df is the document frequency for the word, N is the number of samples in the dataset. The term frequency tf is the number of times a word has occurred in the sample while the inverse document frequency idf refers to the inverse number of documents where the word has occurred. The higher the tf_idf of a word in a document, the more relevant the document. The output of this phase was three numerical vectors for each sample.

5.2.1.3 Distinguishing request

We present a hypothesis: all normal server requests fall into the same category. A static web request, for example, may only contain plain text, whereas API server requests are mostly in JSON format, incoming database queries are SQL, and online compilers use programming language-format requests. A malicious request must be classified differently than normal requests, such as a script request to a static web server or API server, which can be classified as code injection ¹ or command injection ². SQL requests to an API server can be classified as SQLi, and script requests to a database can be classified as command injection or stored XSS attacks. We can determine whether a request is 'normal' to a server by comparing the types of incoming sketchy requests to the average categories of normal requests. If the similarity is low, we can consider that the origin of the incoming request is unusual. For example, if a typical request to a secured server is in JSON format (plain text), but a suspicious request is in JavaScript (client-side script), we can conclude that the incoming request is malicious. If a suspicious request is in XML format (plain text), we can safely assume that the user made a mistake and the alert was false.

From the hypothesis, We can create a CNN model to detect the type of request. Then we compare the incoming requests with the "normal" corresponding category and decide whether the requests is malicious or not using Logistic regression.

Logistic Regression is used when the dependent variable(target) is categorical.

For example, in our case:

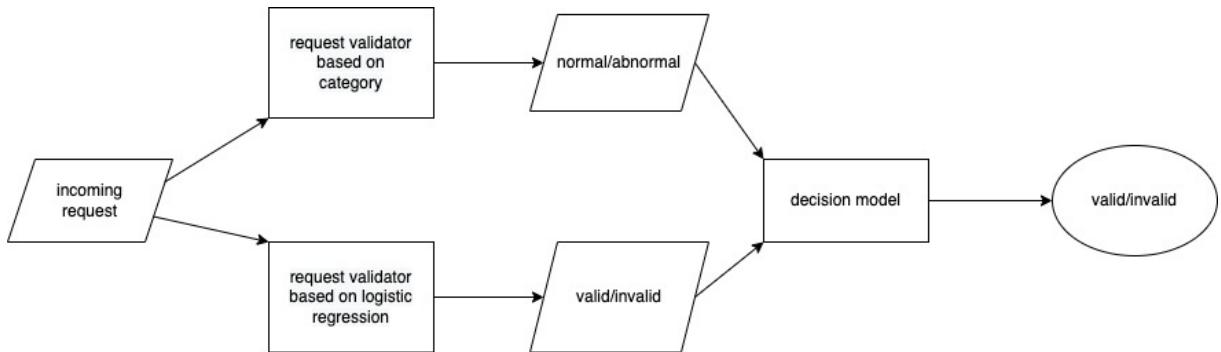
- To predict whether a request is malicious or not.

Our architecture is described in the below figure (Figure 13). Suggest a reasonable decision model for the combined result: When two prediction is the same, the result is straight forward. When the logistic regression model decided that the request is malicious, but the CNN model predicted the request is nornal, the CNN is favored. Otherwise, the Regression model classified the request as valid but the CNN predicted as abnormal, we'll favor the Regression. The desicion model can be expressed in a decision table:

The CNN validator will run for a set period of time (usually one or two weeks) to collect the familiar category of incoming requests and assign a threshold (which can be the mean or maximum (if we are optimistic) distance between each request vector in the observing stage and the sum vector). The same will also happened with the Regression model. After that training phase, the module will run on 'active phase', parallel with the WAF.

1 Code injection is the exploitation of a computer bug that is caused by processing invalid data. The Injection is used by an attacker to introduce code into a vulnerable computer program and change the course of execution.

2 Command injection is an attack in which the goal is the execution of arbitrary commands on the host operating system via a vulnerable application.

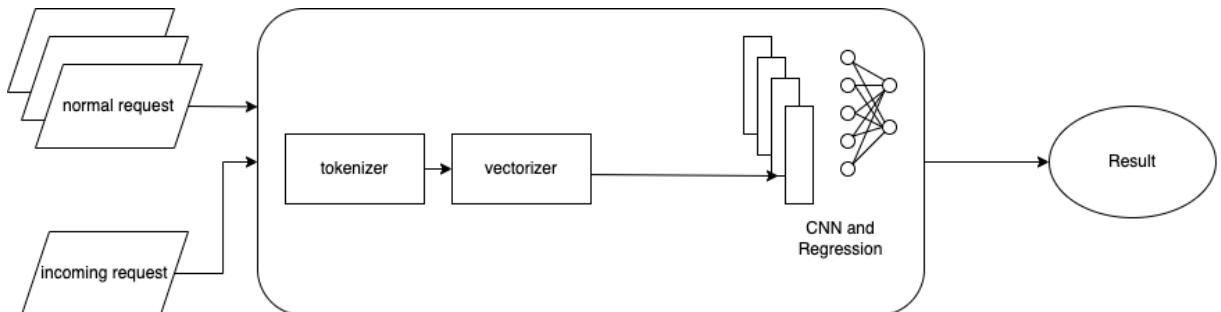
**Figure 5.6:** Malicious request validator architecture

	Regression	CNN	Result
Valid	Normal	Valid	
Valid	Abnormal	Valid	
Invalid	Normal	Valid	
Invalid	Abnormal	Invalid	

Tables 5.1: Table 4.1

- To predict whether a request is malicious or not.

Our architecture is described in Figure 14 below, suggest a reasonable decision model for the combination of CNN and the Regression model

**Figure 5.7:** Decision model for the combination of CNN and the Regression model

The request is routed through the module, which predicts the category. The category of suspicious request is then compared with the common category. Then the Regression model will determined whether the request is good or bad, combining with the normal or abnormal status to decide the result.

6

IMPLEMENTATION

Trong chương này, chúng tôi trình bày các bước trong quá trình hiện thực hệ thống bao gồm tiền xử lý dữ liệu, xây dựng mã nguồn hệ thống, đặc tả phần cứng, hậu xử lý kết quả phân đoạn và trực quan hóa kết quả thí nghiệm.

Table of Contents

6.1	Tiền xử lý dữ liệu	40
6.2	Xây dựng mã nguồn hệ thống	44
6.3	Đặc tả phần cứng	45
6.4	Hậu xử lý kết quả phân đoạn	45
6.5	Trực quan hóa kết quả thí nghiệm	46

6.1 Tiề̂n xử lý dữ liệu

Trong mục này, chúng tôi trình bày các công việc chúng tôi sẽ thực hiện trên tập dữ liệu trước lúc sử dụng cho quá trình huấn luyện bao gồm nội suy dữ liệu, trích xuất thành phần gan và biến đổi cường độ sáng điểm ảnh.

6.1.1 Nội suy dữ liệu

Khi xem xét thông số kích thước điểm ảnh trong khối ảnh CT (cột thứ ba Tables A.1), chúng tôi nhận thấy kích thước điểm ảnh giữa các bộ ảnh CT là khác nhau, đặc biệt khoảng cách giữa hai lớp ảnh CT ở một số bộ ảnh rất lớn. Điều này không tốt cho việc học của lớp convolution bởi tính chất khai thác thông tin về cấu trúc không gian của nó. Chúng tôi thực hiện nội suy dữ liệu như mô phỏng ở Figure 6.1. Ở đây chúng tôi sử dụng nội suy bậc ba với hàm `zoom` trong gói `ndimage` của thư viện `scipy` trong Python với kích thước đầu ra của mỗi điểm ảnh là $1\text{mm}(W) \times 1\text{mm}(H) \times 1\text{mm}(D)$.

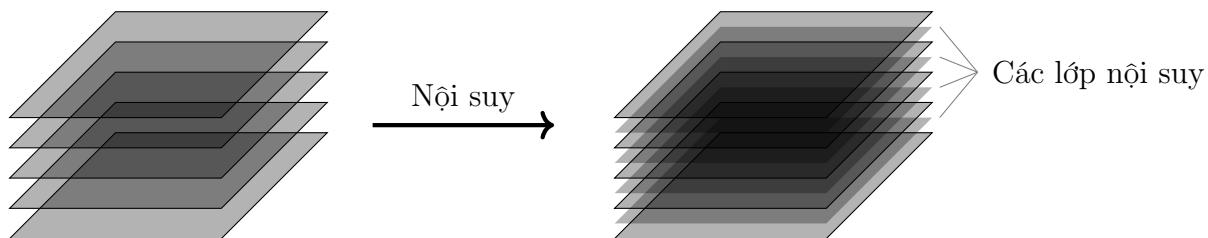


Figure 6.1: Nội suy được sử dụng để lấp các lớp bị thiếu trong khối ảnh CT (Nguồn [20]).

6.1.2 Trích xuất thành phần gan

Từ kết quả khảo sát tình trạng bộ dữ liệu (Tables 4.1), chúng tôi nhận thấy việc sử dụng khối ảnh CT gốc cho việc huấn luyện là bất khả thi vì nhãn phân đoạn cho động mạch ở một số bệnh nhân không được cung cấp. Tuy nhiên, phần động mạch trong cơ quan gan là không đáng kể và hầu hết động mạch đều nằm bên ngoài gan, chúng tôi đề xuất thực hiện trích xuất thành phần gan và chỉ sử dụng thành phần này để huấn luyện hệ thống. Figure 6.2 mô tả việc trích xuất thành phần gan trong khối ảnh CT nhờ sử dụng nhãn phân đoạn gan được cung cấp.

Từ đây trở đi, hệ thống của chúng tôi sẽ đặt trên một giả thiết rằng nhãn phân đoạn cơ quan gan đã biết trước và hệ thống chỉ phân đoạn hệ thống mạch máu bên trong cơ quan gan.

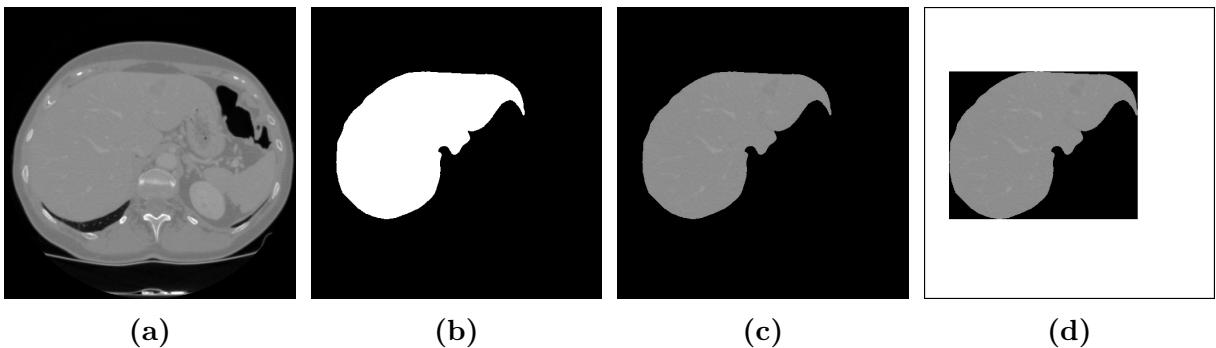


Figure 6.2: Mô phỏng quá trình trích xuất thành phần gan trong khối ảnh CT. (a) ảnh CT gốc. (b) nhãn phân đoạn gan được sử dụng để trích xuất thành phần gan trong ảnh CT. (c) phần ảnh CT chứa gan sau trích xuất. (d) thu giảm kích thước dữ liệu bằng cách giữ lại khối dữ liệu nhỏ nhất chứa gan.

6.1.3 Biến đổi cường độ sáng

Chúng tôi thực hiện biến đổi cường độ sáng điểm ảnh với mong muốn làm rõ hình ảnh mạch máu trên dữ liệu đầu vào, giúp quá trình học dễ dàng hơn. Chúng tôi tiến hành phân tích phân phối mức sáng của các điểm ảnh trong cơ quan gan. Figure 6.3 là biểu đồ phân phối mức sáng nền gan và mạch máu gan của hai đại diện là bệnh nhân số 5 và bệnh nhân số 8. Từ biểu đồ chúng ta thấy phân phối mức sáng trên các bệnh nhân khác nhau có sự chênh lệch. Tuy nhiên, phân phối này có dạng phân phối chuẩn. Chúng tôi đề xuất thực hiện chuẩn hoá để đưa dữ liệu về cùng một phân phối. Chúng tôi thực hiện tính các giá trị ngưỡng bao gồm

$$\text{Left limit} = \mu - \alpha\sigma \quad (6.1)$$

và

$$\text{Right limit} = \mu + \beta\sigma. \quad (6.2)$$

Trong đó, *Left limit* là ngưỡng dưới, *Right limit* là ngưỡng trên, μ là giá trị trung bình và σ là độ lệch chuẩn của các điểm ảnh trong khối cơ quan gan. Hai giá trị α và β chúng tôi đề xuất lần lượt là 3 và 3.5. Giá trị β cao hơn α do độ sáng của các điểm ảnh thuộc mạch máu hầu hết cao hơn độ sáng nền gan, mục đích giữ lại các điểm ảnh thuộc mạch máu có độ sáng cao. Chúng tôi áp dụng các giá trị ngưỡng trên dữ liệu đầu vào. Gọi I^3 là không gian khối dữ liệu, công thức áp dụng ngưỡng vào khối dữ liệu như sau

$$P_{1x} = \begin{cases} \text{Left limit}, & \text{nếu } P_{0x} < \text{Left limit} \\ \text{Right limit}, & \text{nếu } P_{0x} > \text{Right limit} \\ P_{0x}, & \text{còn lại,} \end{cases} \quad (6.3)$$

trong đó, P_{0x} là giá trị điểm ảnh đầu vào và P_{1x} là giá trị điểm ảnh đầu ra tại toạ độ x với $x \in I^3$. Sau đó, chúng tôi chuẩn hoá miền giá trị dữ liệu về khoảng 0 đến 1 theo công thức sau

$$P_{2x} = \frac{P_{1x} - \text{Left limit}}{\text{Right limit} - \text{Left limit}}, \quad (6.4)$$

trong đó, P_{1x} là giá trị đầu ra trong Formula 6.3 và P_{2x} là giá trị sau chuẩn hoá.

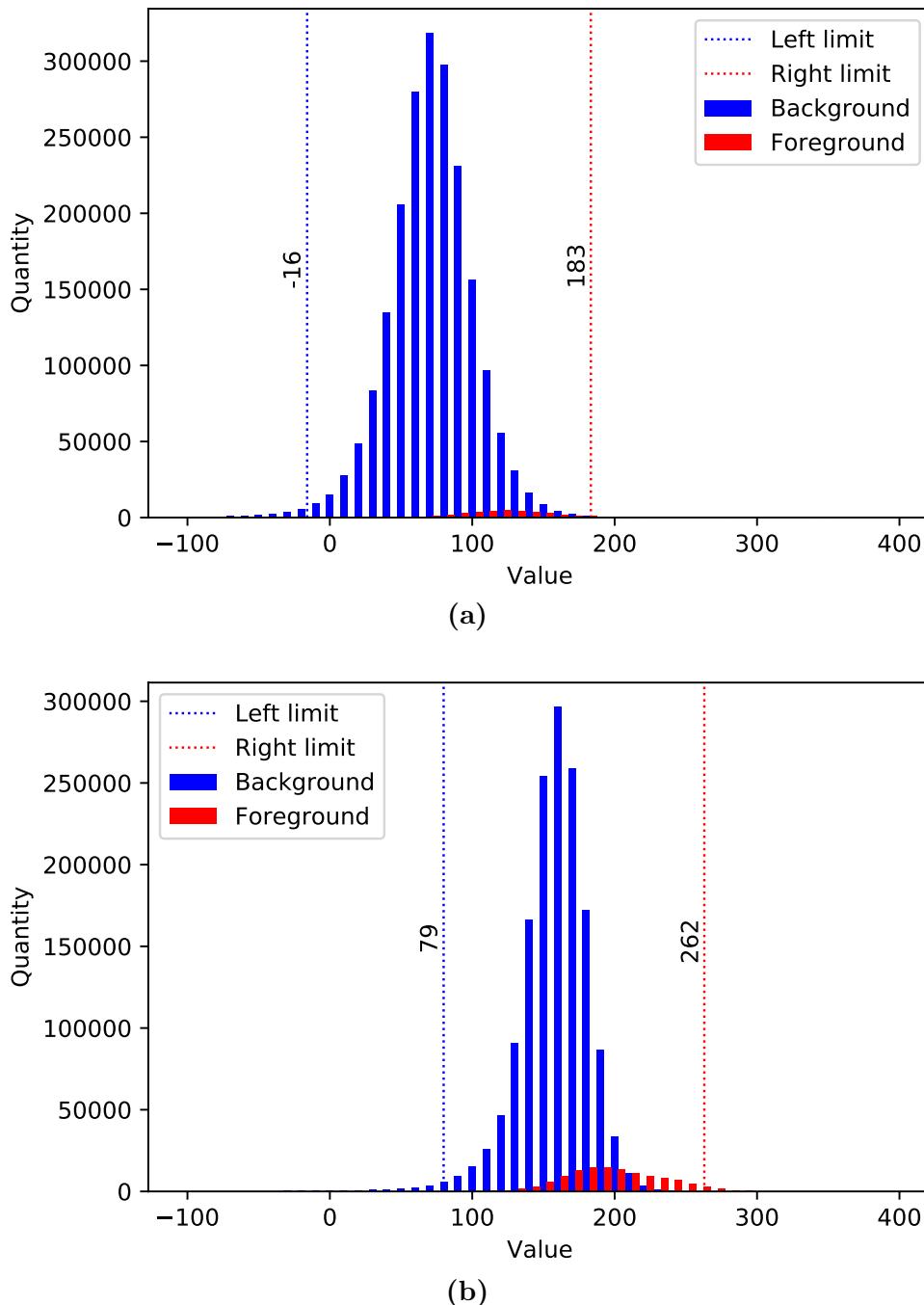


Figure 6.3: Phân phối mức sáng của các điểm ảnh thuộc nền cơ quan gan (Background) và mạch máu gan (Foreground) cùng ngưỡng giới hạn trái (Left limit) và phải (Right limit). **(a)** bệnh nhân số 5. **(b)** bệnh nhân số 8.

Để hình ảnh mạch máu trở nên rõ hơn, chúng tôi tiến hành biến đổi giá trị mức sáng của các điểm ảnh theo công thức sau

$$P_{3x} = P_{2x}^2. \quad (6.5)$$

Figure 6.4 mô tả kết quả chuẩn hóa dữ liệu sau mỗi giai đoạn. So sánh kết quả trước và sau chuẩn hóa, chúng ta có thể thấy hình ảnh mạch máu đã trở nên rõ ràng hơn rất nhiều. Đây là cơ sở quan trọng giúp công tác huấn luyện đạt hiệu quả tốt hơn.

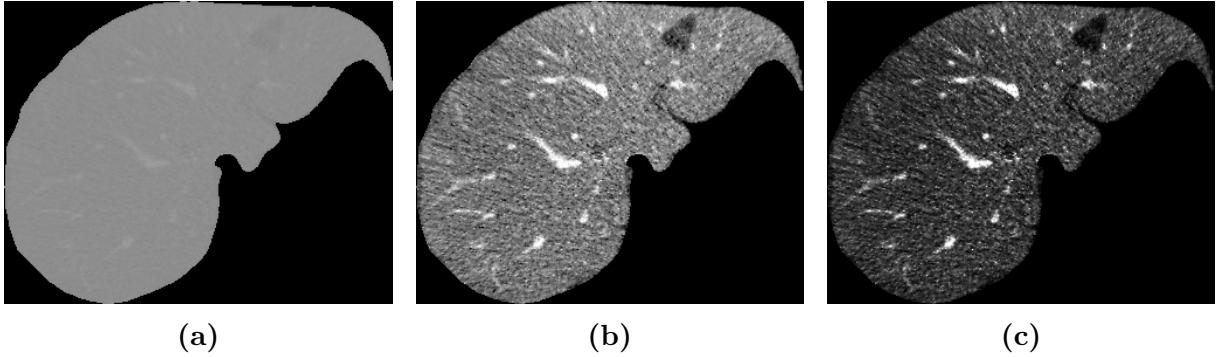


Figure 6.4: Biến đổi cường độ sáng điểm ảnh để làm rõ mạch máu. (a) ảnh trước khi biến đổi. (b) ảnh sau khi thực hiện đặt ngưỡng trên và dưới. (c) biến đổi cường độ sáng bằng hàm bình phương giá trị điểm ảnh.

6.1.4 Làm giàu dữ liệu

Khi huấn luyện một mạng học sâu, thực chất chúng ta đang học một hàm biến đổi có thể ánh xạ từ dữ liệu đầu vào tới nhãn tương ứng. Mô hình càng lớn, số lượng tham số cần học càng nhiều và quá trình huấn luyện đòi hỏi cần nhiều dữ liệu để đạt hiệu quả cao. Do đó, đối với những tập dữ liệu nhỏ, bước làm giàu dữ liệu có vai trò quan trọng trong việc cải thiện hiệu năng mô hình.

Có nhiều cách làm giàu dữ liệu, ví dụ như phép lật hoặc xoay hình. Tuy nhiên, đối với ảnh y khoa như ảnh CT, những phép biến đổi này không có nhiều ý nghĩa vì khi chụp ảnh CT, bệnh nhân được yêu cầu nằm ở một vị trí cố định. Việc xoay, lật ảnh CT sẽ tạo ra các ảnh chụp với các cơ quan bị đảo ngược. Điều này không xảy ra trong thực tế.

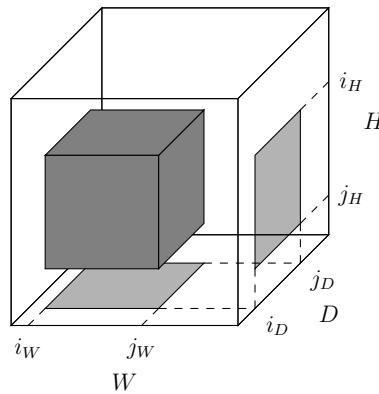


Figure 6.5: Làm giàu dữ liệu bằng phép trích xuất khối ngẫu nhiên.

Trong luận văn này, chúng tôi làm giàu dữ liệu bằng cách thực hiện cắt ngẫu nhiên một khối dữ liệu từ khối dữ liệu ban đầu như Figure 6.5. Kích thước của khối dữ liệu đầu ra là ngẫu nhiên. Tuy nhiên, với mong muốn giữ lại được nhiều thông tin khi đưa vào huấn luyện, chúng tôi đặt ràng buộc kích thước cắt cho khối dữ liệu. Gọi S_H, S_W, S_D lần lượt là kích thước chiều cao, chiều rộng và chiều sâu của khối dữ liệu ban đầu; i_k, j_k lần lượt là chỉ số bắt đầu và kết thúc của khối dữ liệu đầu ra trên khối dữ liệu đầu vào theo chiều k , với $k \in \{H, W, D\}$. Chúng tôi giới hạn i_k trong khoảng $[0, \lfloor 0.1S_k \rfloor]$ và j_k trong khoảng $[\lfloor 0.9S_k \rfloor, S_k]$. Sau đó, chúng tôi thực hiện chia không chồng lấp khối dữ liệu có được thành các khối có kích thước $112 \times 112 \times 112$ và lần lượt đưa vào huấn luyện.

6.2 Xây dựng mã nguồn hệ thống

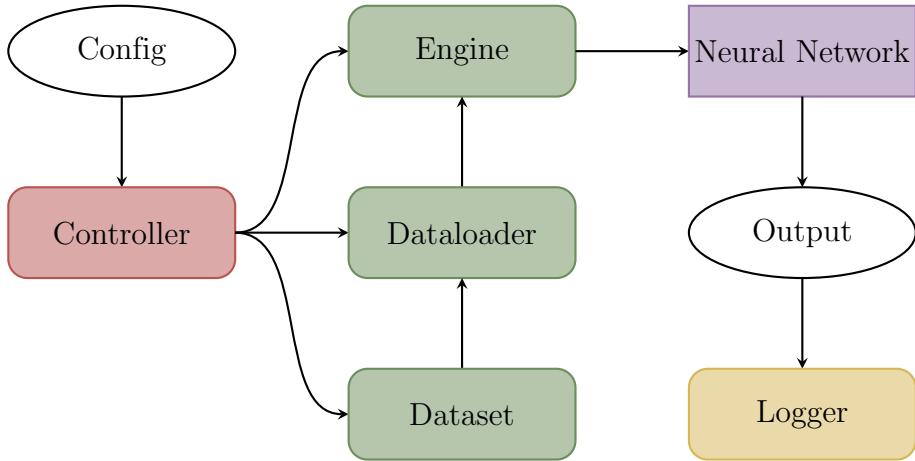


Figure 6.6: Kiến trúc của bộ mã nguồn *Insight Deep Learning* (Source: [21]).

Để thuận tiện hơn trong quá trình thí nghiệm cũng như giúp hệ thống có thể dễ dàng phát triển và bảo trì trong tương lai, chúng tôi sử dụng bộ mã nguồn *Insight Deep Learning* của GVLab được hiện thực bằng PyTorch với những ưu điểm sau đây:

- Bộ mã được tổ chức thành các khái niệm rõ ràng, hợp lý giúp dễ dàng trong quá trình sử dụng, mở rộng và bảo trì.
- Hỗ trợ huấn luyện song song trên nhiều GPU.
- Hỗ trợ giám sát thời gian thực trong quá trình huấn luyện với nhiều dạng khác nhau như biểu đồ, chữ,...
- Trong trường hợp quá trình huấn luyện bị dừng đột ngột vì lý do không mong muốn, hệ thống cho phép phục hồi quá trình huấn luyện nhờ khả năng sao lưu và khôi phục trạng thái trong quá trình huấn luyện.
- Các siêu tham số và các cài đặt cho quá trình huấn luyện được quản lý trong một tệp YAML¹. Nhờ vậy, khi cần thay đổi thông số cài đặt, lập trình viên chỉ cần chỉnh sửa thông tin trong tệp này.
- Sử dụng chung một bộ mã nguồn với cùng một ngôn ngữ, cấu trúc cho phép các thành viên của GVLab hợp tác với nhau thuận lợi hơn cũng như quá trình kế thừa và phát triển mã nguồn dễ dàng hơn.

Kiến trúc của bộ mã được mô tả trong Figure 6.6. Controller là khái niệm được thực thi đầu tiên với các thông số lấy từ khái niệm Config. Controller có nhiệm vụ điều khiển các thành phần Dataset, DataLoader, Engine làm việc với nhau. Dataset xử lý tất cả các tác vụ liên quan đến dữ liệu như đọc tệp, làm giàu dữ liệu. DataLoader quản lý số lượng mẫu và cách mà mỗi mẫu được đưa vào huấn luyện. Engine sẽ lấy dữ liệu từ DataLoader để đưa vào Neural Network. Phụ thuộc vào loại Engine, đầu ra của Neural Network sẽ được dùng để tính giá trị lỗi, đưa ra kết quả dự đoán hoặc để tính toán các độ đo ghi vào Logger.

¹ YAML là một chuẩn tuân tự hóa dữ liệu cho nhiều ngôn ngữ lập trình dưới dạng chữ, giúp con người dễ dàng đọc, hiểu.

Trong luận văn này, chúng tôi kế thừa và phát triển bộ mã để phục vụ cho quá trình xây dựng hệ thống. Những bổ sung chính mà chúng tôi thực hiện trên bộ mã liên quan đến các mô hình trong các công trình liên quan chúng tôi tham khảo và khả năng làm việc với dữ liệu 3D.

6.3 Đặc tả phần cứng

Huấn luyện mạng học sâu với một lượng lớn dữ liệu đòi hỏi sức mạnh tính toán cao để công tác huấn luyện diễn ra nhanh chóng và hiệu quả. Để đáp ứng yêu cầu đó, trong luận văn này, chúng tôi sử dụng hệ thống máy tính hiệu năng cao (HPCC) được hỗ trợ bởi GVLab nhằm mục đích huấn luyện hệ thống. Tables 6.1 mô tả chi tiết thông số cấu hình hệ thống mà chúng tôi đã sử dụng.

Tables 6.1: Thông tin phần cứng hệ thống máy tính.

STT	Phần cứng	Cấu hình
1	CPU	Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz 16 cores
2	RAM	128 GB
3	GPU	NVIDIA Tesla P100 16GB

6.4 Hậu xử lý kết quả phân đoạn

Hệ thống mạch máu là một cấu trúc khép kín. Nếu coi hệ thống này là một đối tượng trong không gian thì nó sẽ gói gọn trong một thành phần liên thông duy nhất. Đối với hệ thống mạch máu trong khối ảnh CT của một phần cơ thể, cụ thể với giả thiết được nêu ra trong Sector 6.1.2, hệ thống này sẽ bao gồm một số lượng hữu hạn ¹ các nhánh mạch máu mà mỗi nhánh này là một thành phần liên thông.

Hệ thống mạch máu có được từ kết quả phân đoạn không thể tránh khỏi có những điểm ảnh bị phân đoạn sai. Đặc biệt, những điểm ảnh rời rạc không thuộc mạch máu bị phân đoạn sai sẽ làm xấu đi kết quả thí nghiệm. Nhằm cải thiện kết quả phân đoạn, chúng tôi đề xuất thực hiện xác định các thành phần liên thông trong cây mạch máu được sinh ra. Sau đó, lần lượt loại bỏ những thành phần liên thông có tổng số lượng điểm ảnh là nhỏ nhất sao cho tổng số lượng điểm ảnh bị loại bỏ không vượt quá 10% tổng số điểm ảnh được phân đoạn là mạch máu.

Việc xác định các thành phần liên thông và loại bỏ các thành phần liên thông nhỏ được chúng tôi thực hiện thông qua các hàm đã được hiện thực sẵn trong ngôn ngữ Python lần lượt là hàm `label` trong gói `ndimage.measurements` của thư viện `scipy` và hàm `histogram` và `argsort` của thư viện `numpy`.

¹ Có khoảng 6 nhánh mạch máu trong cơ quan gan với 3 nhánh xuất phát từ tĩnh mạch chủ và 3 nhánh còn lại xuất bạn từ tĩnh mạch cửa.

6.5 Trực quan hoá kết quả thí nghiệm

Sau khi có kết quả phân đoạn mạch máu cũng như đường chính giữa và điểm phân nhánh mạch máu. Chúng tôi thực hiện trực quan hoá kết quả thông qua hai bước.

Bước thứ nhất, chúng tôi tiến hành trích xuất lưới bề mặt đối tượng cần trực quan bằng thư viện VTK. Bước này nhằm giảm thiểu khối lượng tính toán trên GPU lúc hiển thị, bởi nếu sử dụng ngay kết quả đầu ra của hệ thống là mảng ba chiều thì khối lượng tính toán cần thực hiện trên GPU là rất lớn. Từ đó, cải thiện chất lượng hiển thị nhờ tăng số lượng khung hình trên giây và việc trực quan sẽ mượt mà hơn.

Bước thứ hai, hiển thị lưới bề mặt của đối tượng trên ứng dụng Slicer. Với ứng dụng Slicer, chúng tôi có thể hiển thị chồng kết quả phân đoạn và nhãn phân đoạn tương ứng lên nhau, từ đó có thể dễ dàng so sánh sự sai khác của kết quả phân đoạn. Figure 6.7 là ví dụ về việc sử dụng ứng dụng Slicer để trực quan hệ thống tĩnh mạch của một bệnh nhân.

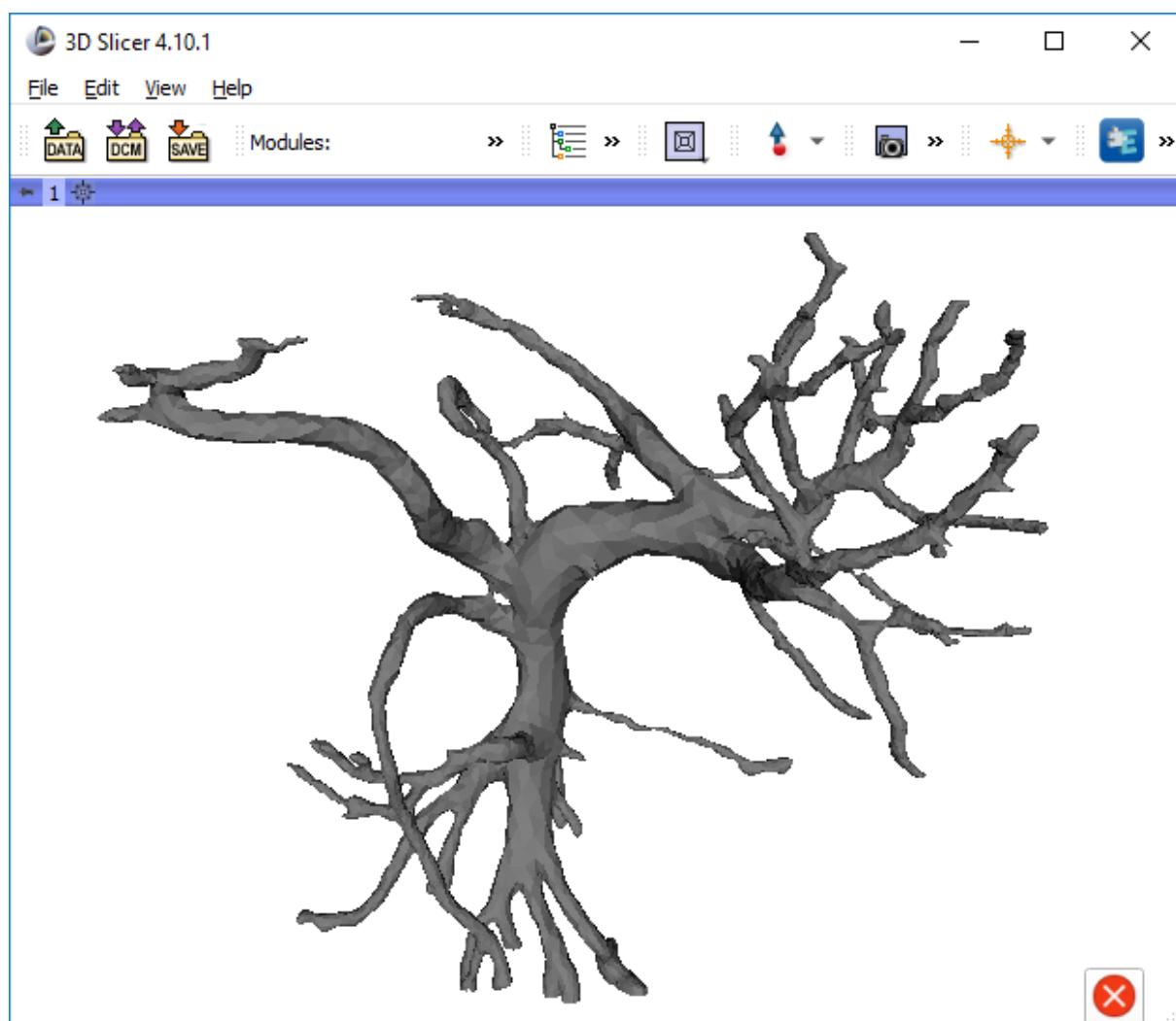


Figure 6.7: Trực quan hoá kết quả trên ứng dụng Slicer.

7

EXPERIMENTS

Trong chương này, chúng tôi trình bày việc phân chia tập dữ liệu trước khi huấn luyện, đưa ra các phương pháp đánh giá và kết quả thí nghiệm. So sánh kết quả giữa các thí nghiệm tham khảo từ các công trình liên quan và các thí nghiệm do chúng tôi đề xuất.

Table of Contents

7.1 Chuẩn bị dữ liệu	48
7.2 Phương pháp đánh giá	48
7.3 Kết quả thí nghiệm	50

7.1 Chuẩn bị dữ liệu

Đối với các giải thuật học máy nói chung cũng như mạng học sâu nói riêng, việc chuẩn bị dữ liệu là vô cùng quan trọng. Nếu sử dụng mô hình đã huấn luyện để dự đoán trên tập dữ liệu không cùng phân bố với tập dữ liệu được sử dụng để huấn luyện mô hình trước đó thì kết quả dự đoán không thể chính xác. Do đó, phân chia dữ liệu sao cho phân bố trên các tập đồng đều với nhau là một trong những yếu tố quan trọng quyết định tới mức hiệu quả của mạng học sâu.

Sau khi khảo sát các ảnh chụp CT của 20 bệnh nhân, chúng tôi nhận thấy sự xuất hiện các khối u có ảnh hưởng đến phổi mức sáng của các điểm ảnh trong cơ quan gan. Do đó, chúng tôi chia các bệnh nhân ra 03 nhóm: bệnh nhân không có khối u trong gan, bệnh nhân có một khối u trong gan và bệnh nhân có nhiều khối u trong gan. Chúng tôi chia tập dữ liệu thành ba tập dữ liệu bao gồm tập huấn luyện, tập kiểm thử và tập kiểm tra sao cho mỗi tập đều có bệnh nhân không có u, bệnh nhân có một khối u và bệnh nhân có nhiều khối u. Chi tiết các tập dữ liệu được chúng tôi trình bày trong Tables 7.1.

Tables 7.1: Bảng phân chia các bệnh nhân thành các tập dữ liệu.

STT	Tập	Số lượng khối u gan			Tổng
		0 khối u	1 khối u	Nhiều khối u	
1	Huấn luyện	5, 7, 11	2, 3, 9, 12	1, 4, 8, 10, 15, 17, 19	14
2	Kiểm thử	14	16	6	3
3	Kiểm tra	20	18	13	3

7.2 Phương pháp đánh giá

Quá trình đánh giá một mô hình phân đoạn hình ảnh y khoa liên quan đến việc đánh giá độ chính xác của kết quả dự đoán. Hai hoạt động cơ bản cần được tiến hành để đánh giá một cách toàn diện một mô hình là đánh giá định tính và đánh giá định lượng. Đánh giá định lượng cho biết một cách tổng quát độ tốt của mô hình, trong khi đó đánh giá định tính cho biết mức độ ổn định của mô hình trong quá trình làm việc (trong trường hợp xấu nhất, trung bình và tốt nhất). Trong phần này, chúng tôi trình bày 4 độ đo có liên quan bao gồm Precision, Recall, IoU và Dice. Trong đó, IoU và Dice được chúng tôi sử dụng để đánh giá kết quả dự đoán của các mô hình.

Ta quy ước,

- TP (true positive): số lượng điểm ảnh foreground được dự đoán đúng,
- TN (true negative): số lượng điểm ảnh background được dự đoán đúng,
- FP (false positive): số lượng điểm ảnh background bị dự đoán sai,
- FN (false negative): số lượng điểm ảnh foreground bị dự đoán sai.

Precision trả lời cho câu hỏi “Số dự đoán thực sự chính xác chiếm bao nhiêu phần trong số các dự đoán foreground?”. Công thức Precision được định nghĩa trong Formula 7.1. Precision càng cao, kết quả dự đoán càng tốt.

$$Precision = \frac{TP}{TP + FP} \quad (7.1)$$

Recall trả lời cho câu hỏi “Số dự đoán thực sự chính xác chiếm bao nhiêu phần trong số các mẫu foreground”. Công thức Recall được định nghĩa trong Formula 7.2. Recall càng cao, kết quả dự đoán càng tốt.

$$Recall = \frac{TP}{TP + FN} \quad (7.2)$$

Dice là độ đo cân bằng giữa Precision và Recall. Trong trường hợp mô hình dự đoán chính xác một lượng nhỏ mẫu thuộc foreground, giá trị Precision sẽ cao, tuy nhiên số lượng false negative sẽ cao theo, nghĩa là giá trị Recall thấp. Trong trường hợp mô hình dự đoán một lượng lớn mẫu thuộc foreground, giá trị Recall sẽ cao, tuy nhiên số lượng false positive sẽ cao theo, tức là giá trị Precision thấp. Dice được tính toán bằng cách sử dụng đồng thời Precision và Recall. Khi giá trị Precision và giá trị Recall cùng cao, giá trị Dice sẽ cao, ngược lại, khi một trong hai giá trị Precision và Recall thấp, giá trị Dice sẽ thấp. Do đó, Dice phù hợp cho các bài toán phân loại có sự mất cân bằng về nhãn. Công thức Dice được định nghĩa trong Formula 7.2.

$$Dice = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (7.3)$$

IoU cho biết mức độ tương tự giữa hai mẫu X và Y . IoU được định nghĩa trong Formula 7.4.

$$IoU = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|} \quad (7.4)$$

7.3 Kết quả thí nghiệm

Trước khi tiến hành các thí nghiệm để đánh giá hệ thống, chúng tôi thực hiện kiểm chứng tính hiệu quả của đề xuất cắt giảm độ sâu trong mô hình U-Net, để từ đó chọn ra mô hình tốt hơn phục vụ cho các thí nghiệm về sau. Tables 7.2 đặc tả thông số chi tiết của các thí nghiệm dùng để so sánh tính hiệu quả trước và sau điều chỉnh số tầng ở mô hình U-Net sử dụng convolution 3D.

Tables 7.2: Thông số các thí nghiệm so sánh tính hiệu quả trước và sau điều chỉnh số tầng ở mô hình U-Net sử dụng convolution 3D.

STT	Kiến trúc	Số tầng	BatchNorm	Hàm lỗi	Learning rate	Momentum
1	U-Net	5	Không	CrossEntropy	0.0001	0.90
2	U-Net*	3	Không	CrossEntropy	0.0001	0.90

Tables 7.3 là kết quả thu được khi so sánh hai mô hình. Từ kết quả này chúng ta thấy rằng, chất lượng phân đoạn hệ thống mạch máu trên tập kiểm tra trước và sau hậu xử lý (cột **Tập kiểm tra** và **Tập kiểm tra***) ở mô hình U-Net* không có nhiều sự khác biệt so với mô hình U-Net. Chứng tỏ, việc cắt giảm hai tầng 4 và 5 trong mô hình U-Net là hợp lý vì chúng không có nhiều đóng góp trong quá trình học. Hơn nữa, việc cắt giảm số tầng giúp cho mô hình trở nên nhẹ hơn, thời gian inference¹ trung bình cho một tập ảnh CT ở mô hình U-Net* là 4.46 giây, nhanh hơn gần 1 giây so với mô hình U-Net là 5.40 giây. Chênh lệch này tuy nhỏ nhưng sẽ có ý nghĩa rất lớn khi áp dụng mô hình vào thực tiễn với khối lượng dữ liệu khổng lồ. Thời gian dành cho công tác chẩn đoán càng được rút ngắn bao nhiêu, cơ hội chữa trị thành công cho bệnh nhân càng lớn bấy nhiêu. Vì vậy, chúng tôi lựa chọn sử dụng mô hình U-Net* trong các thí nghiệm về sau để đánh giá hệ thống.

Tables 7.3: Kết quả so sánh tính hiệu quả trước và sau điều chỉnh số tầng ở mô hình U-Net sử dụng convolution 3D.

STT	Kiến trúc	Thời gian inference (s)	Tập kiểm tra		Tập kiểm tra*	
			IoU	Dice	IoU	Dice
1	U-Net	5.40	0.365	0.533	0.379	0.548
2	U-Net*	4.46	0.380	0.550	0.375	0.545

¹ Inference trong lĩnh vực học sâu là thuật ngữ dùng để chỉ giai đoạn mô hình sau huấn luyện được sử dụng để dự đoán các mẫu dữ liệu thử nghiệm. Không giống quá trình huấn luyện, inference không thực hiện lan truyền ngược để tính toán lỗi và cập nhật trọng số.

Dầu tiên, chúng tôi tiến hành 2 thí nghiệm tham khảo từ các công trình có liên quan. Tables 7.4 mô tả chi tiết các thí nghiệm và các siêu tham số được sử dụng.

Tables 7.4: Thông số các thí nghiệm tham khảo.

STT	Thí nghiệm	Kiến trúc	BatchNorm	Hàm lõi	Learning rate	Momentum
1	Thí nghiệm 1	DeepVesselNet	Không	CrossEntropy	0.0001	0.90
2	Thí nghiệm 2	U-Net*	Không	CrossEntropy	0.0001	0.90

Tables 7.5 là kết quả huấn luyện của các thí nghiệm được kiểm tra trên các tập dữ liệu. Từ kết quả thu được, chúng ta thấy rằng mô hình U-Net hoạt động hiệu quả hơn mô hình DeepVesselNet với kết quả tốt nhất trên hai giá trị đo IoU và Dice lần lượt là 0.380 và 0.550 trên tập kiểm tra. Tuy nhiên, các giá trị này trở nên xấu đi sau bước hậu xử lý. Điều này có thể được giải thích rằng, mô hình phân đoạn cho kết quả hệ thống mạch máu rời rạc, những điểm dự đoán đúng mạch máu đã bị loại bỏ vì thể tích thành phần liên thông của nó quá nhỏ.

Tables 7.5: Kết quả các thí nghiệm tham khảo.

STT	Thí nghiệm	Tập huấn luyện		Tập kiểm thử		Tập kiểm tra		Tập kiểm tra*	
		IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice
1	Thí nghiệm 1	0.364	0.530	0.459	0.627	0.365	0.533	0.362	0.531
2	Thí nghiệm 2	0.392	0.557	0.518	0.681	0.380	0.550	0.375	0.545

Chi tiết kết quả phân đoạn cho từng bệnh nhân trong tập kiểm tra được thể hiện trong Tables 7.6. Trong đó, giá trị IoU và Dice trong trường hợp tốt nhất và xấu nhất lần lượt là 0.404, 0.575 và 0.344, 0.512. Figure 7.1, Figure 7.2 và Figure 7.3 là hình ảnh trực quan kết quả thí nghiệm 2 cho trường hợp tốt nhất. Figure 7.4, Figure 7.5 và Figure 7.6 là hình ảnh trực quan kết quả thí nghiệm 2 cho trường hợp xấu nhất.

Tables 7.6: Kết quả thí nghiệm 2 của từng bệnh nhân trong tập kiểm tra.

STT	Bệnh nhân	IoU	Dice
1	Bệnh nhân 13	0.404	0.575
2	Bệnh nhân 18	0.344	0.512
3	Bệnh nhân 20	0.378	0.549

Tiếp theo, chúng tôi tiến hành 6 thí nghiệm do chúng tôi đề xuất. Tables 7.7 mô tả chi tiết các thí nghiệm và các siêu tham số được sử dụng.

Tables 7.7: Thông số các thí nghiệm đề xuất.

STT	Thí nghiệm	Kiến trúc	BatchNorm	Hàm lỗi	Learning rate	Momentum
1	Thí nghiệm 3	DeepVesselNet	Có	CrossEntropy	0.001	0.90
2	Thí nghiệm 4	DeepVesselNet	Không	Dice	0.0001	0.90
3	Thí nghiệm 5	DeepVesselNet	Có	Dice	0.001	0.90
4	Thí nghiệm 6	U-Net*	Có	CrossEntropy	0.001	0.90
5	Thí nghiệm 7	U-Net*	Không	Dice	0.0001	0.90
6	Thí nghiệm 8	U-Net*	Có	Dice	0.001	0.90

Tables 7.8 là kết quả huấn luyện cho 6 thí nghiệm này. Với mỗi thí nghiệm, chúng tôi đánh giá trên tất cả các tập dữ liệu cũng như kết quả sau bước hậu xử lý. Từ số liệu thu được, chúng ta thấy được rằng, các thí nghiệm có sử dụng lớp batchnorm và hàm lỗi dice (thí nghiệm 5 và thí nghiệm 8) cho kết quả tốt hơn hẳn với kết quả tốt nhất trên các độ đo IoU, Dice lần lượt là 0.384, 0.552 trên thí nghiệm sử dụng mô hình U-Net. Đồng thời, ở các thí nghiệm này, việc thực hiện hậu xử lý cho thấy sự hiệu quả, các giá trị IoU, Dice được cải thiện với giá trị tương ứng là 0.400, 0.569. So với kết quả trong các thí nghiệm tham khảo từ các công trình liên quan, kết quả thí nghiệm do chúng tôi đề xuất cải thiện kết quả của hệ thống 4%.

Tables 7.8: Kết quả các thí nghiệm đề xuất.

STT	Thí nghiệm	Tập huấn luyện		Tập kiểm thử		Tập kiểm tra		Tập kiểm tra*	
		IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice
1	Thí nghiệm 3	0.347	0.508	0.457	0.624	0.256	0.402	0.261	0.409
2	Thí nghiệm 4	0.417	0.584	0.555	0.713	0.351	0.517	0.371	0.538
3	Thí nghiệm 5	0.411	0.578	0.531	0.692	0.373	0.541	0.381	0.550
4	Thí nghiệm 6	0.328	0.483	0.402	0.565	0.377	0.545	0.371	0.537
5	Thí nghiệm 7	0.406	0.572	0.558	0.715	0.352	0.519	0.360	0.527
6	Thí nghiệm 8	0.439	0.606	0.539	0.699	0.384	0.552	0.400	0.569

Chi tiết kết quả phân đoạn cho từng bệnh nhân trong tập kiểm tra được thể hiện trong Tables 7.9. Trong đó, giá trị IoU và Dice trong trường hợp tốt nhất và xấu nhất lần lượt là 0.474, 0.643 và 0.342, 0.510. Figure 7.7, Figure 7.8 và Figure 7.9 là hình ảnh trực quan kết quả thí nghiệm 8 cho trường hợp tốt nhất. Figure 7.10, Figure 7.11 và Figure 7.12 là hình ảnh trực quan kết quả thí nghiệm 8 cho trường hợp xấu nhất.

Tables 7.9: Kết quả thí nghiệm 8 của từng bệnh nhân trong tập kiểm tra.

STT	Bệnh nhân	IoU	Dice
1	Bệnh nhân 13	0.474	0.643
2	Bệnh nhân 18	0.342	0.510
3	Bệnh nhân 20	0.384	0.555

Với mong muốn tiếp tục cải thiện kết quả thí nghiệm và nhờ hiểu được những ưu điểm của mạng DenseNet, ResNet; chúng tôi đề xuất sử dụng ý tưởng của hai mạng này cho mô hình U-Net. Tables 7.10 mô tả chi tiết kiến trúc và hàm lỗi được sử dụng cũng như các siêu tham số cho các thí nghiệm.

Tables 7.10: Thông số các thí nghiệm có sự kết hợp của DenseNet và ResNet.

STT	Thí nghiệm	Kiến trúc	BatchNorm	Hàm lỗi	Learning rate	Momentum
1	Thí nghiệm 9	Dense-U-Net	Có	Dice	0.001	0.90
2	Thí nghiệm 10	Res-U-Net	Có	Dice	0.001	0.90

Tables 7.11 là kết quả huấn luyện hai mô hình Dense-U-Net và Res-U-Net. Từ kết quả thí nghiệm, chúng ta thấy rằng, mô hình kết hợp ý tưởng DenseNet cho kết quả tốt hơn mô hình kết hợp ý tưởng ResNet. Tuy nhiên, kết quả trong các thí nghiệm này không cải thiện hơn so với các thí nghiệm trước.

Tables 7.11: Kết quả các thí nghiệm DenseUNet và ResUNet.

STT	Thí nghiệm	Tập huấn luyện		Tập kiểm thử		Tập kiểm tra		Tập kiểm tra*	
		IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice
1	Thí nghiệm 9	0.417	0.584	0.529	0.691	0.356	0.521	0.382	0.549
2	Thí nghiệm 10	0.422	0.590	0.536	0.697	0.355	0.520	0.377	0.544

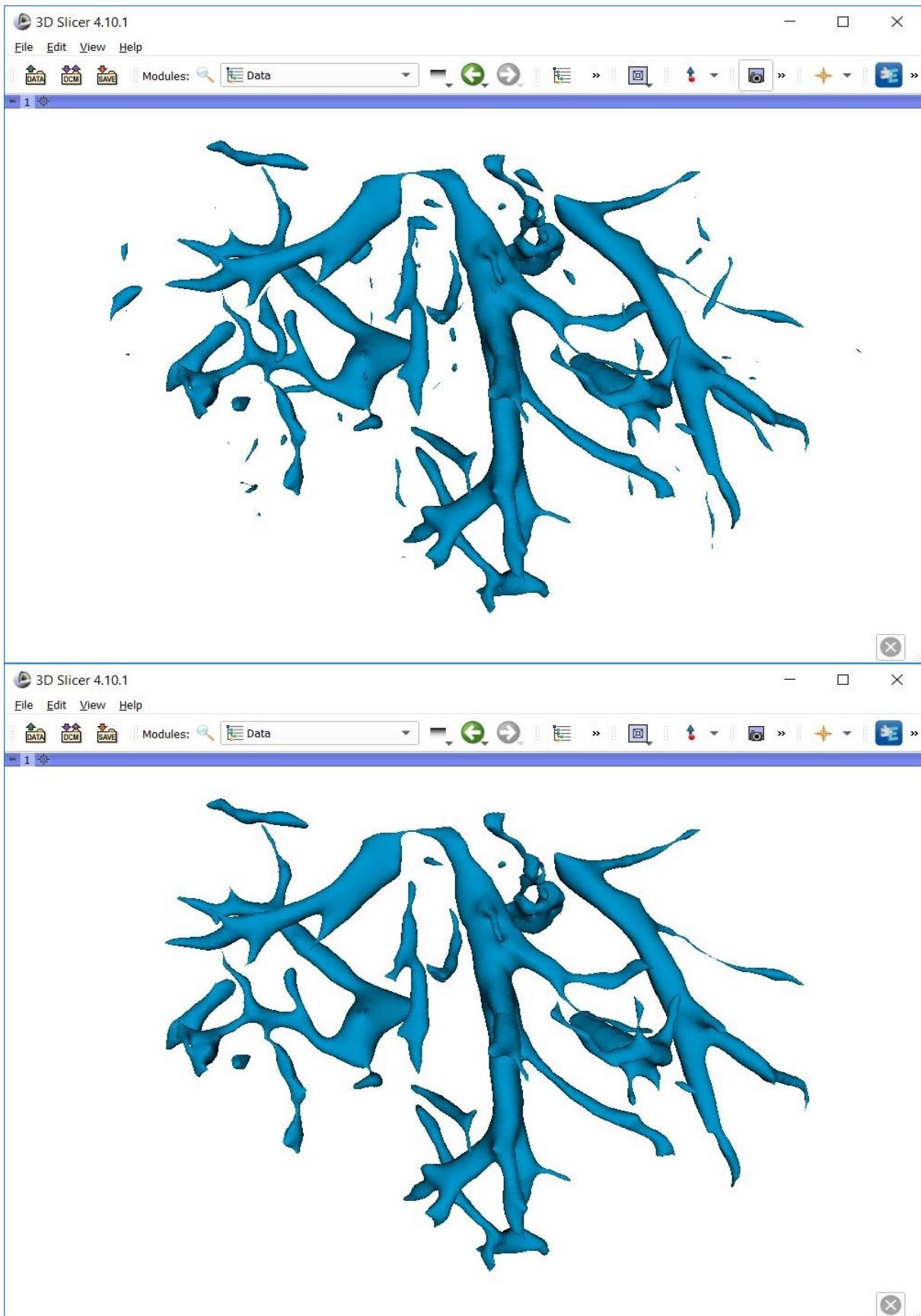


Figure 7.1: Kết quả hệ thống mạch máu của trường hợp tốt nhất trong thí nghiệm 2 trước và sau hậu xử lý. Các thành phần nhỏ và rời rạc đã được loại bỏ.

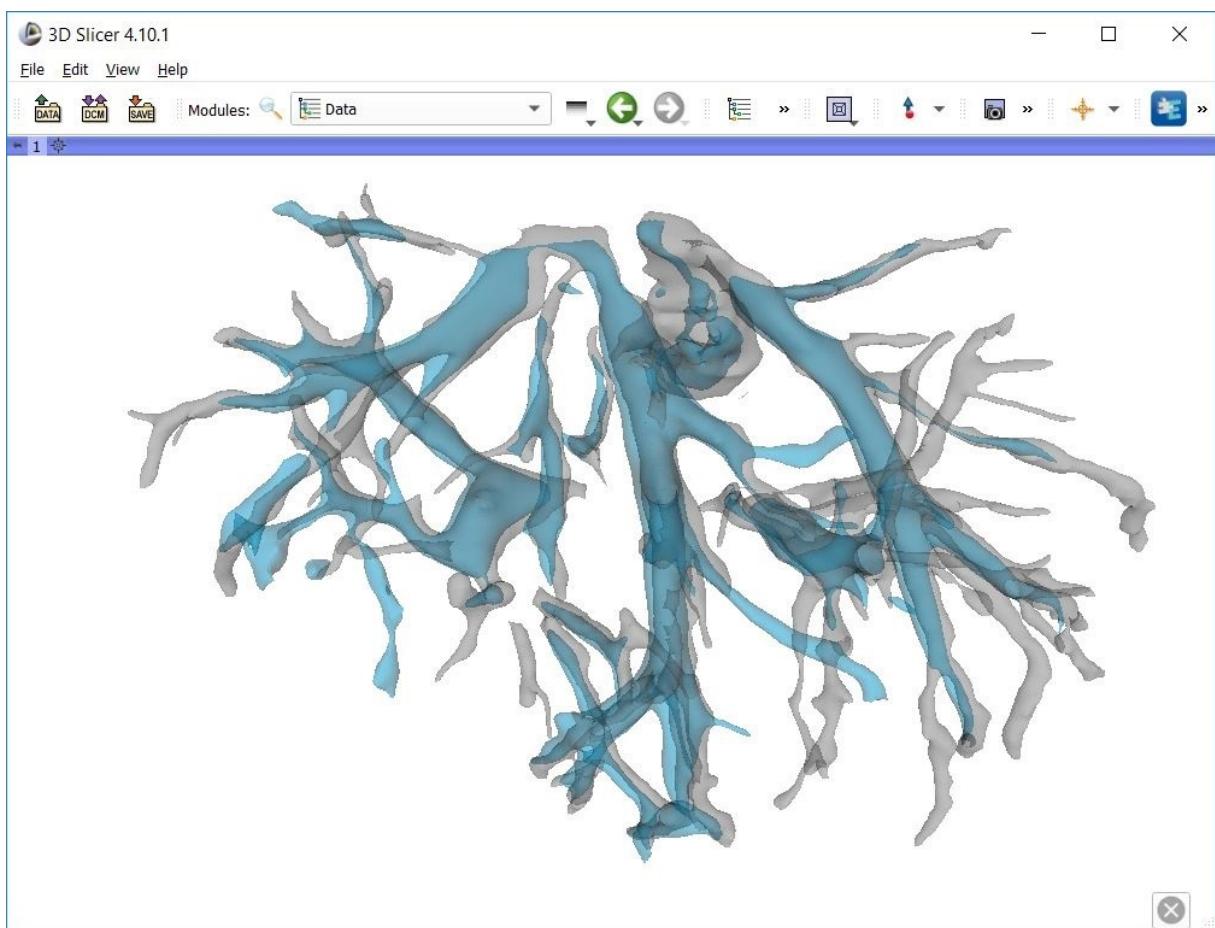


Figure 7.2: Kết quả hệ thống mạch máu (màu xanh) của trường hợp tốt nhất trong thí nghiệm 2 khi so sánh với nhãn phân đoạn (màu xám). Hệ thống đã phân đoạn được các nhánh chính của mạch máu, tuy nhiên, còn rất nhiều nhánh mạch máu nhỏ hệ thống chưa phân đoạn được.

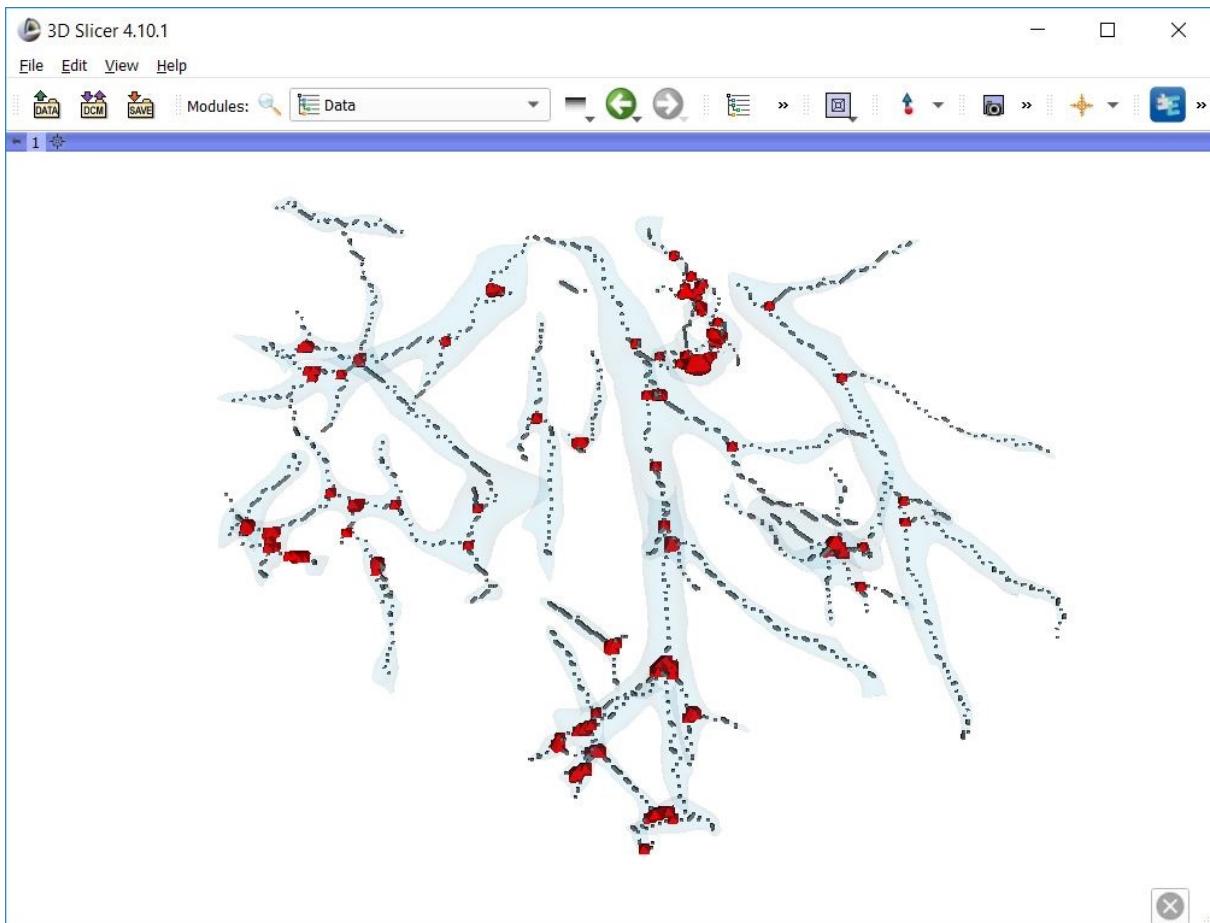


Figure 7.3: Kết quả tìm đường chính giữa (đường màu xám) và điểm phân nhánh (màu đỏ) của trường hợp tốt nhất trong thí nghiệm 2. Hệ thống đã xác định được đường chính giữa và các điểm phân nhánh của mạch máu.

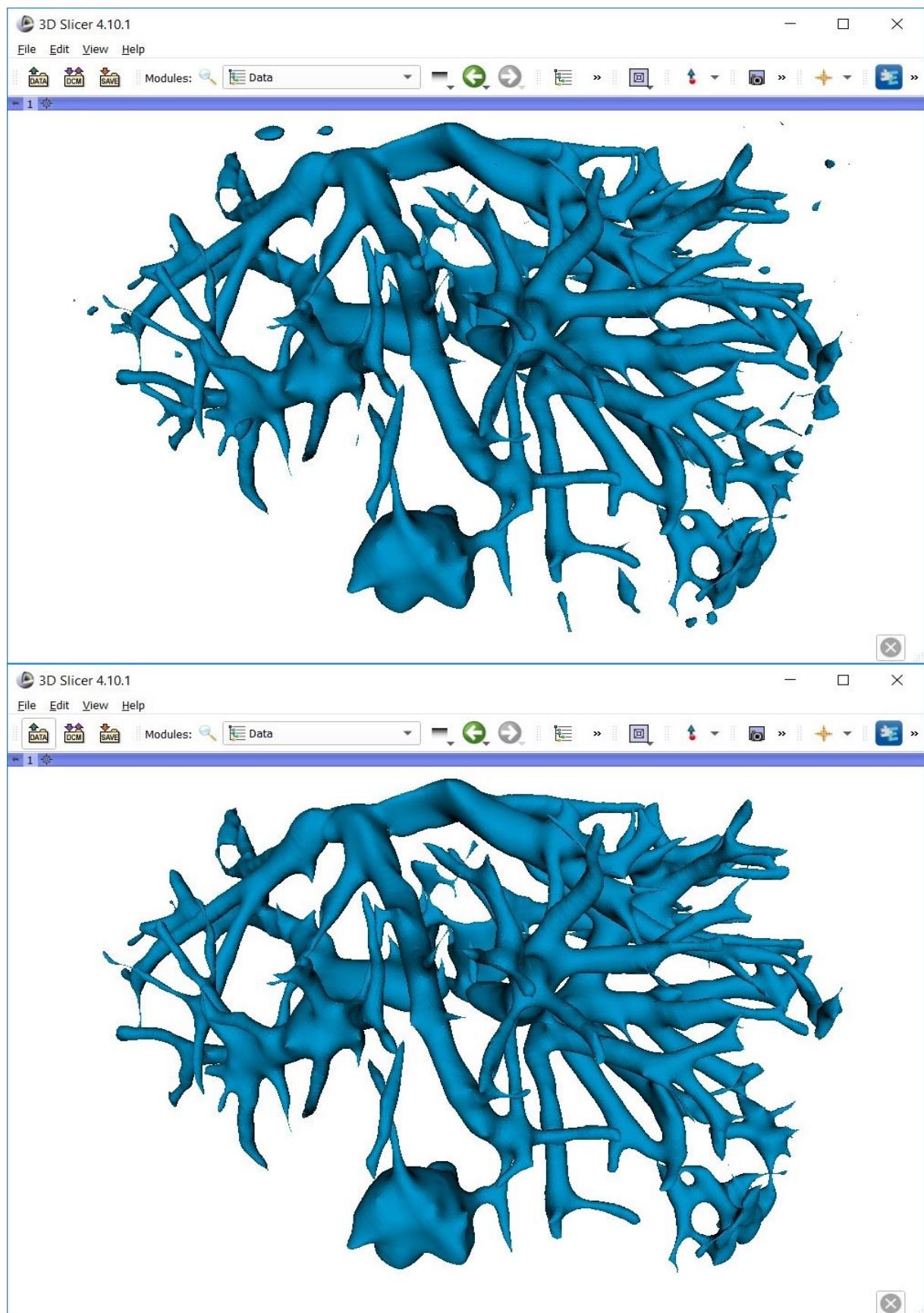


Figure 7.4: Kết quả hệ thống mạch máu của trường hợp xấu nhất trong thí nghiệm 2 trước và sau hậu xử lý. Hệ thống phân đoạn mạch máu quá lớn, bước hậu xử lý không có nhiều hiệu quả.

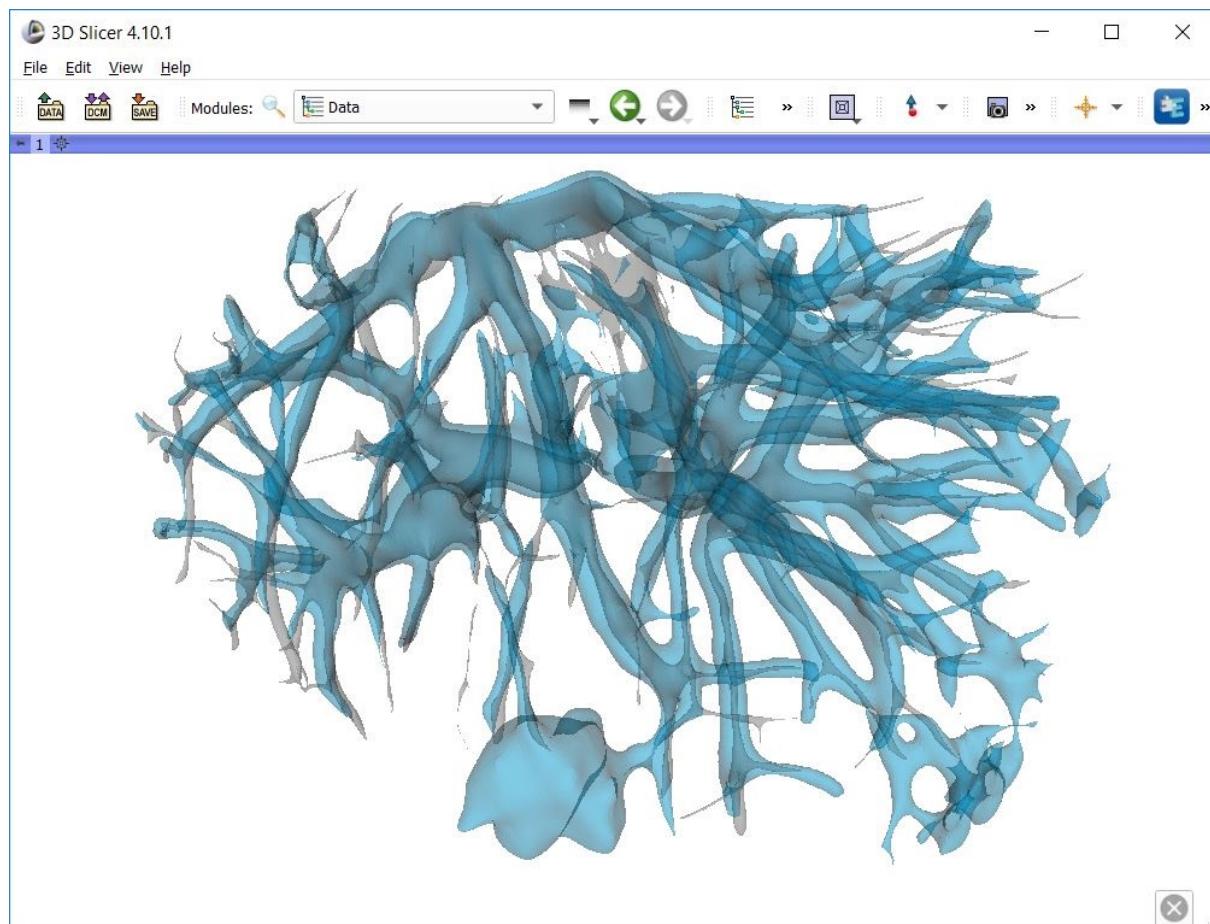


Figure 7.5: Kết quả hệ thống mạch máu (màu xanh) của trường hợp xấu nhất trong thí nghiệm 2 khi so sánh với nhän phân đoạn (màu xám). Các mạch máu được phân đoạn lớn hơn rất nhiều so với thực tế. Ngoài ra, xuất hiện một khối phân đoạn sai lớn có thể là khối u.

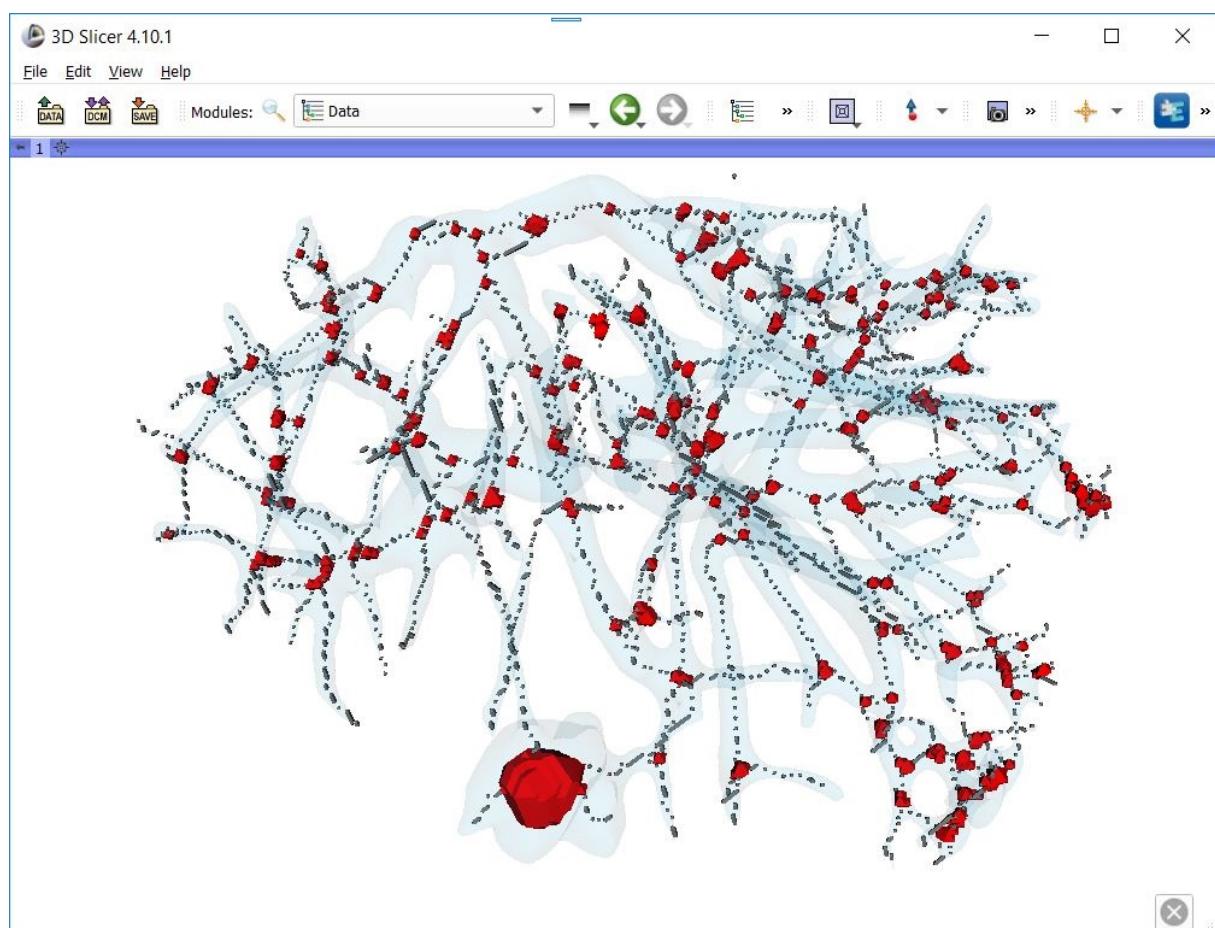


Figure 7.6: Kết quả tìm đường chính giữa (đường màu xám) và điểm phân nhánh (màu đỏ) của trường hợp xấu nhất trong thí nghiệm 2. Hệ thống đã xác định được đường chính giữa và các điểm phân nhánh của mạch máu. Tuy nhiên, xuất hiện rất nhiều điểm phân nhánh sai.

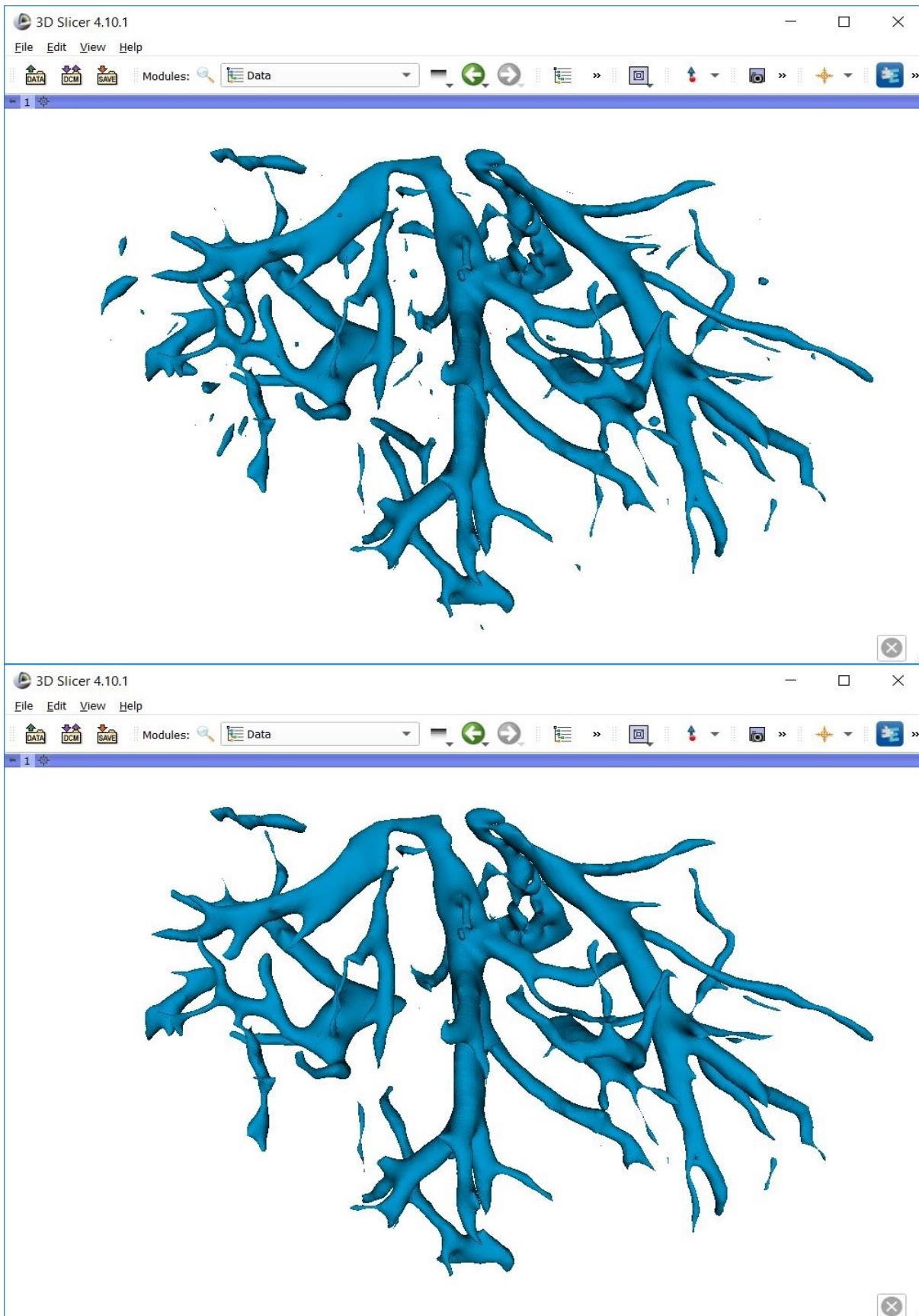


Figure 7.7: Kết quả hệ thống mạch máu của trường hợp tốt nhất trong thí nghiệm 8 trước và sau hậu xử lý. Các thành phần nhỏ và rời rạc hầu như đã được loại bỏ hoàn toàn.

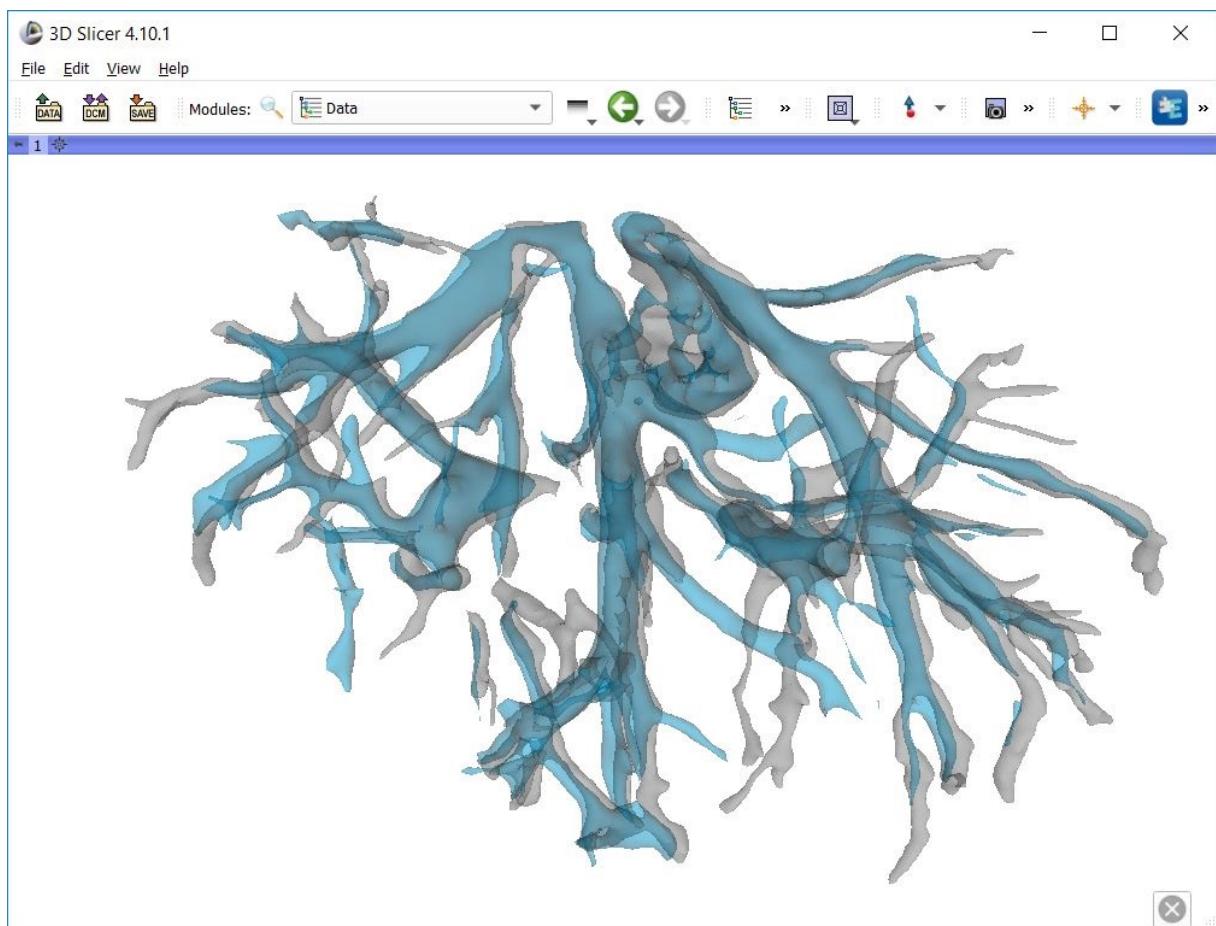


Figure 7.8: Kết quả hệ thống mạch máu (màu xanh) của trường hợp tốt nhất trong thí nghiệm 8 khi so sánh với nhãn phân đoạn (màu xám). Kết quả phân đoạn khá sát với thực tế, chỉ một vài nhánh nhỏ chưa phân đoạn được.

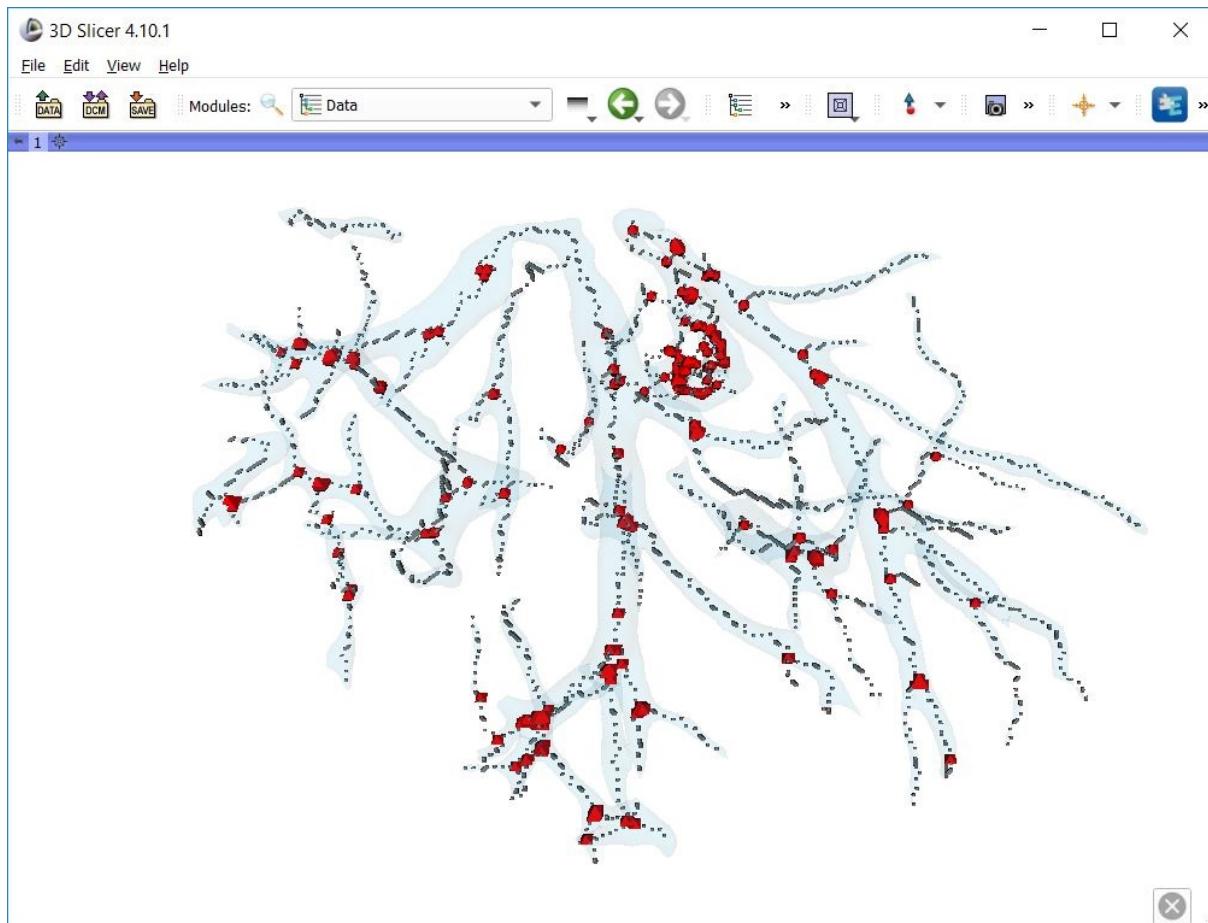


Figure 7.9: Kết quả tìm đường chính giữa (đường màu xám) và điểm phân nhánh (màu đỏ) của trường hợp tốt nhất trong thí nghiệm 8. Hệ thống đã xác định được đường chính giữa và các điểm phân nhánh của mạch máu.

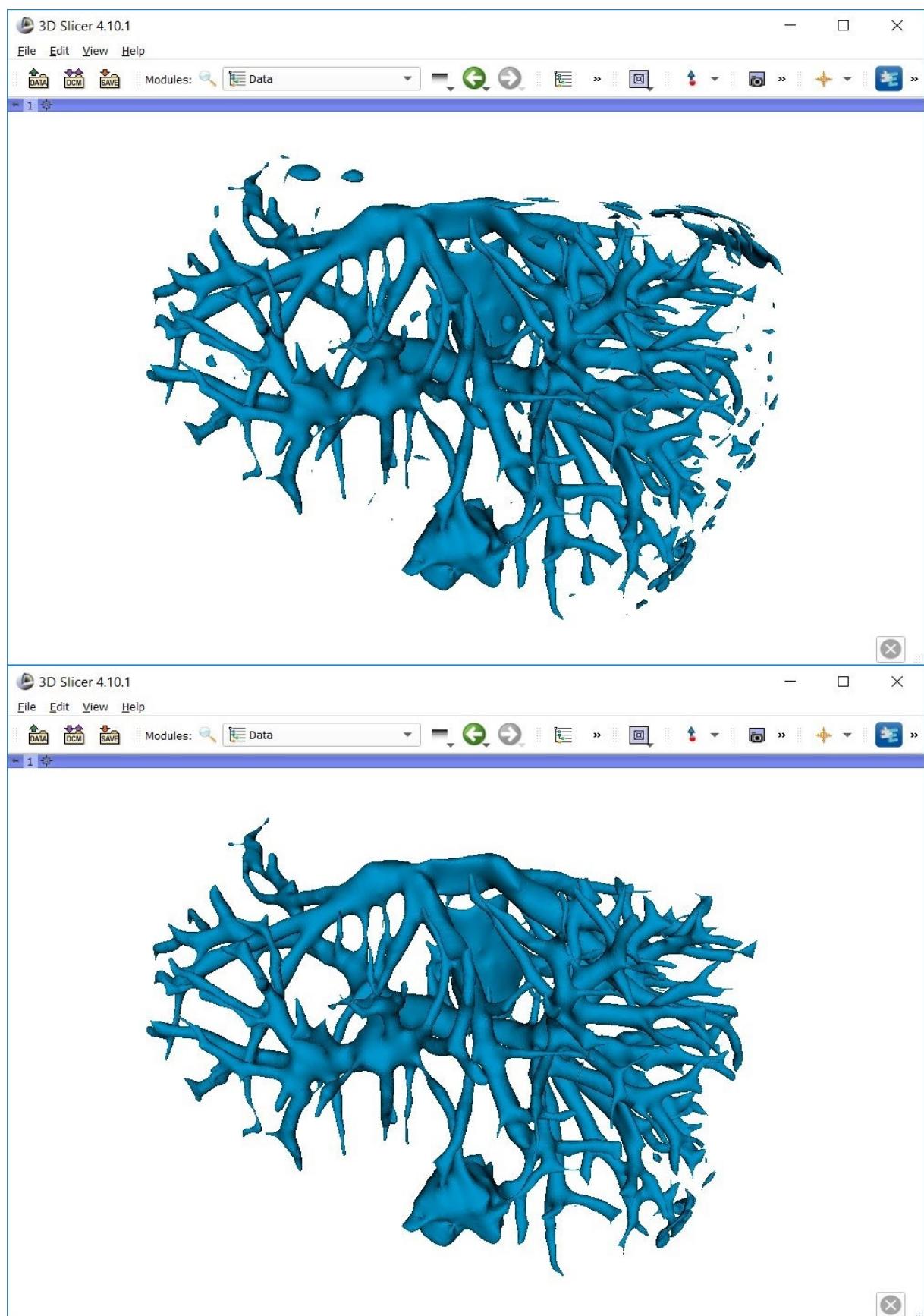


Figure 7.10: Kết quả hệ thống mạch máu của trường hợp xấu nhất trong thí nghiệm 8 trước và sau hậu xử lý. Bước hậu xử lý đã loại bỏ đi rất nhiều vị trí dự đoán sai.

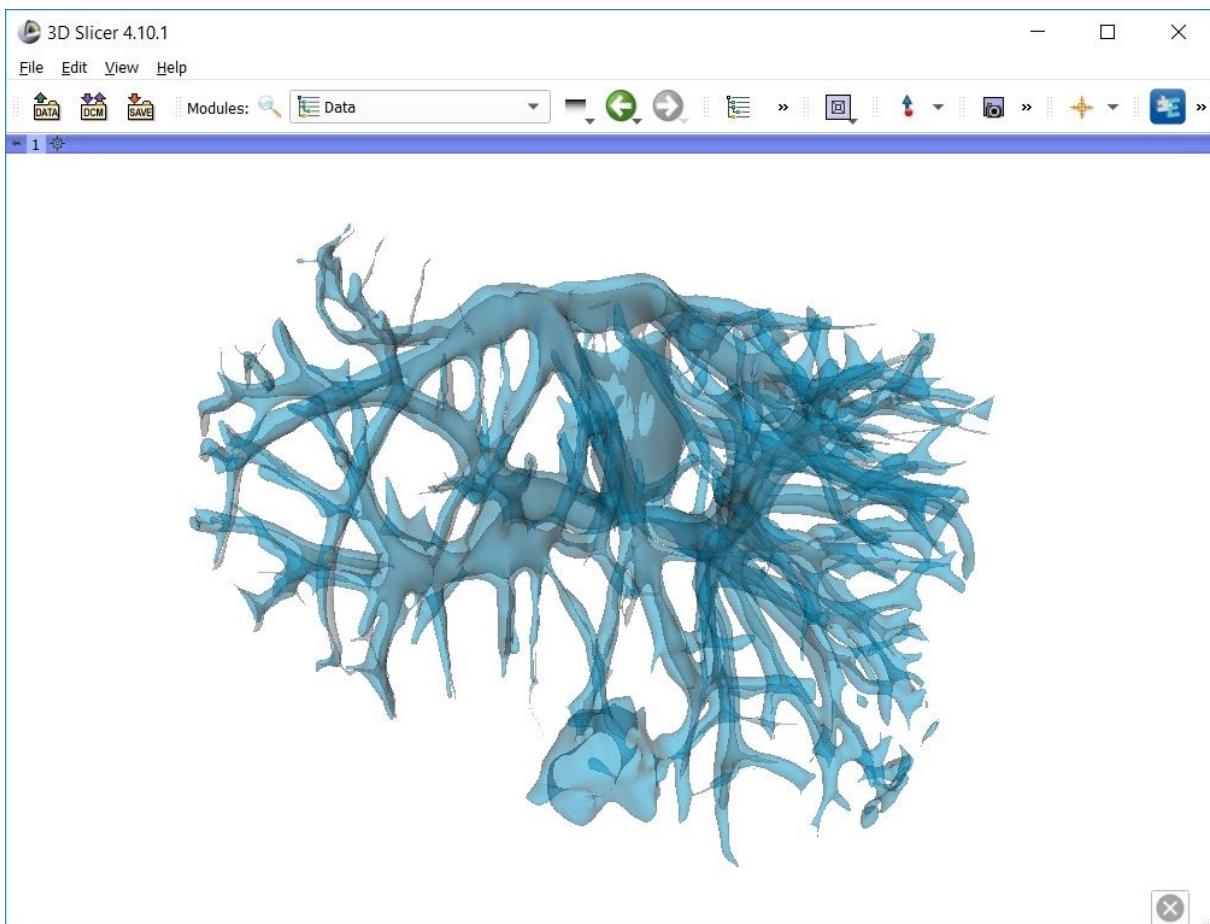


Figure 7.11: Kết quả hệ thống mạch máu (màu xanh) của trường hợp xấu nhất trong thí nghiệm 8 khi so sánh với nhãn phân đoạn (màu xám). Các mạch máu được phân đoạn lớn hơn rất nhiều so với thực tế. Ngoài ra, xuất hiện một khối phân đoạn sai lớn có thể là khối u.

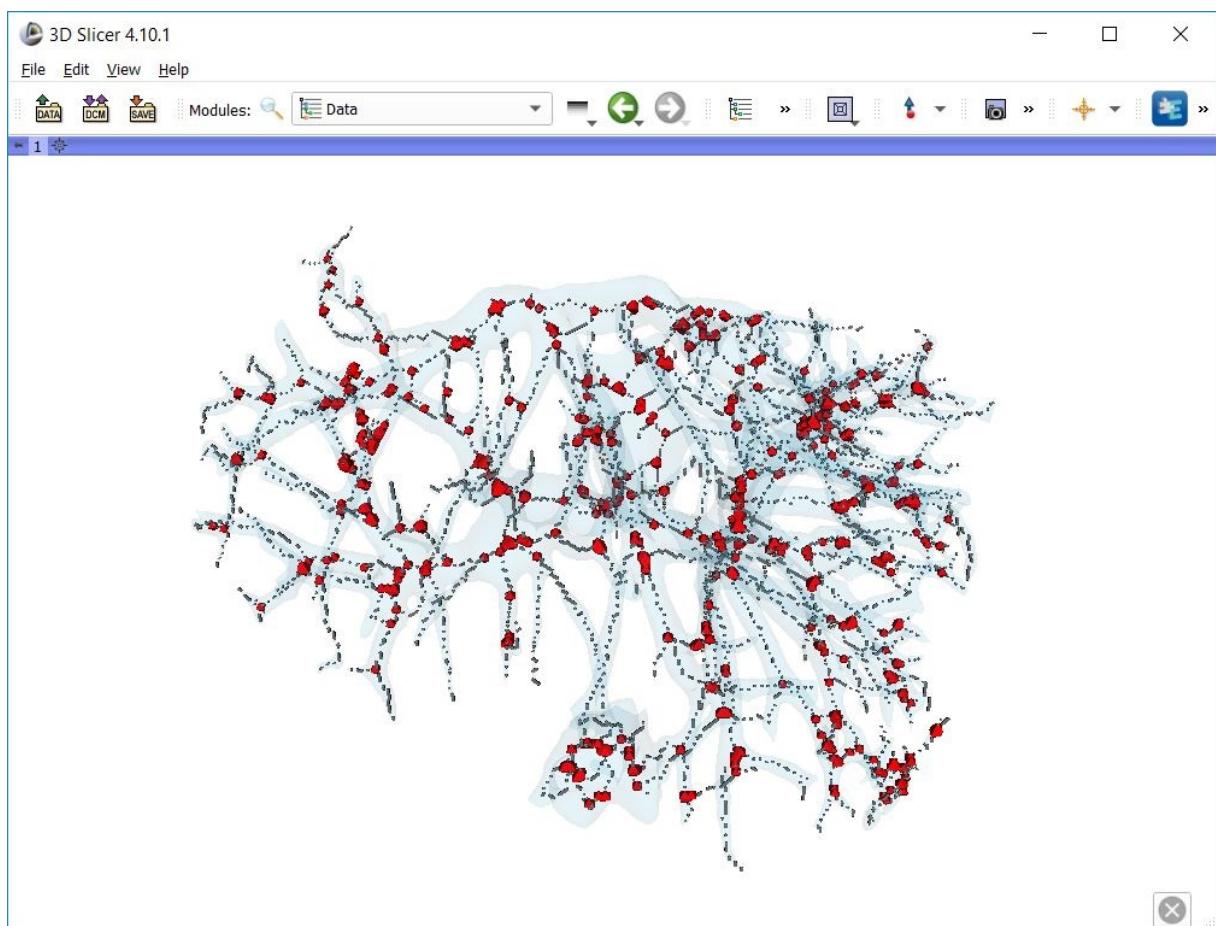


Figure 7.12: Kết quả tìm đường chính giữa (đường màu xám) và điểm phân nhánh (màu đỏ) của trường hợp xấu nhất trong thí nghiệm 8. Hệ thống đã xác định được đường chính giữa và các điểm phân nhánh của mạch máu. Tuy nhiên, xuất hiện rất nhiều đường chính giữa và điểm phân nhánh sai.

8

CONCLUSION

Trong chương này, chúng tôi tổng kết lại các kết quả đã đạt được, đồng thời nêu ra các mặt hạn chế trong quá trình nghiên cứu và xây dựng hệ thống. Cuối cùng, chúng tôi trình bày hướng mở rộng, phát triển của hệ thống trong tương lai.

Table of Contents

8.1	Kết quả đạt được	68
8.2	Hạn chế và hướng phát triển	69

8.1 Kết quả đạt được

Kết thúc giai đoạn Luận văn tốt nghiệp, chúng tôi đã được trang bị thêm các kiến thức trong lĩnh vực Xử lý ảnh nói chung, cũng như các phương pháp tiếp cận trong lĩnh vực Phân đoạn hình ảnh y khoa nói riêng. Chúng tôi đã hiện thực được một hệ thống xây dựng hệ thống mạch máu của cơ quan gan từ ảnh chụp CT.

Hệ thống mà chúng tôi xây dựng đã đáp ứng được đầy đủ các yêu cầu trong bài toán đặt ra, bao gồm:

- Phân đoạn được hệ thống mạch máu của cơ quan gan từ ảnh chụp CT sử dụng mạng học sâu. Kết quả phân đoạn trên tập kiểm tra có độ chính xác theo độ đo Dice là 0.569 và kết quả trực quan rất khả quan.
- Xác định được đường chính giữa của mạch máu sau kết quả phân đoạn thông qua đề xuất thực hiện trích xuất khung xương đối tượng bằng phép toán skeleton trong không gian 3D.
- Xác định được điểm phân nhánh của mạch máu thông qua đề xuất xem xét số lượng điểm ảnh liền kề cũng thuộc đường chính giữa mạch máu của từng điểm ảnh trong đường chính giữa mạch máu.

Bên cạnh các kết quả nêu trên, chúng tôi còn có hai đóng góp quan trọng liên quan đến tiền xử lý và hậu xử lý dữ liệu. Những đề xuất này đã giúp kết quả phân đoạn của hệ thống cải thiện đáng kể.

Về tiền xử lý dữ liệu, chúng tôi đề xuất sử dụng phương pháp nội suy để lấp các lớp CT bị thiếu trong khối ảnh CT. Bên cạnh đó, chúng tôi đề xuất thực hiện biến đổi độ sáng điểm ảnh thông qua phương pháp đặt ngưỡng giới hạn và hàm bình phương giá trị điểm ảnh giúp hình ảnh mạch máu hiển thị rõ hơn. Từ đó, kết quả học trứ nên tốt hơn.

Về hậu xử lý dữ liệu, chúng tôi đã cải thiện được kết quả phân đoạn thông qua đề xuất xác định các thành phần liên thông và loại bỏ các thành phần liên thông nhỏ không thuộc mạch máu trong kết quả phân đoạn.

Với những kết quả có được của luận văn này, chúng tôi hy vọng có thể đóng góp một phần vào các công trình nghiên cứu trong lĩnh vực Phân đoạn hình ảnh y khoa về sau. Sớm đưa được các công trình nghiên cứu vào ứng dụng thực tiễn giúp cải thiện chất lượng trong công tác y tế cũng như cuộc sống xã hội.

8.2 Hạn chế và hướng phát triển

Hạn chế. Bên cạnh các kết quả đạt được, chúng tôi cũng gặp các khó khăn trong quá trình hiện thực hệ thống. Nếu các vấn đề này sớm được khắc phục rất có thể kết quả hệ thống sẽ tốt hơn nữa.

- Tập dữ liệu quá nhỏ dẫn tới quá trình huấn luyện không thực sự tốt. Trong nhiều trường hợp hệ thống nhanh chóng rơi vào overfitting trong khi chưa học được toàn diện các đặc trưng của hệ thống mạch máu trong ảnh CT.
- Hệ thống vẫn phải đặt trên giả thiết nhãn phân đoạn cơ quan gan đã được cung cấp do nhãn phân đoạn động mạch ở bên ngoài cơ quan gan không được cung cấp ở một số bệnh nhân trong bộ dữ liệu.
- Hệ thống chưa thực sự dễ sử dụng. Các công đoạn của hệ thống vẫn phải thực hiện một cách riêng lẻ thông qua sự điều khiển, chưa xây dựng được một ứng dụng end-to-end¹.

Hướng phát triển. Không thể nào phủ nhận sự cần thiết của một hệ thống chẩn đoán thông minh trong lĩnh vực y tế, đặc biệt là một hệ thống cho phép cải thiện khả năng nhìn và phán đoán về hình ảnh y khoa. Chính vì vậy, việc mở rộng và phát triển hệ thống mà chúng tôi đã xây dựng mở ra nhiều hướng mới trong tương lai. Do giới hạn về thời gian thực hiện luận văn nên nhiều ý tưởng chúng tôi vẫn chưa thực hiện được.

Thứ nhất, chúng tôi muốn phát triển hệ thống để có thể phân đoạn hệ thống mạch máu trên toàn ảnh CT. Xây dựng hệ thống mạch máu một cách toàn diện không chỉ giúp nhìn thấy hệ thống mạch máu bên trong cơ quan mà còn thấy được sự tương quan của cơ quan gan và các cơ quan khác trong cơ thể.

Thứ hai, chúng tôi muốn hệ thống không chỉ phân đoạn hệ thống mạch máu và còn phân đoạn được các cơ quan khác nhau cũng như các tổn thương bất thường trong cơ thể. Xác định chính xác vị trí tổn thương giúp quá trình chữa trị có thể tiến hành thuận lợi hơn.

Thứ ba, chúng tôi muốn khai thác các thông tin đặc thù trong lĩnh vực y khoa. Tìm hiểu mối liên quan giữa kết quả chẩn đoán và quá trình điều trị. Từ đó, giúp phát triển một hệ thống có khả năng đưa ra phác đồ điều trị từ kết quả dự đoán.

Cuối cùng, chúng tôi muốn xây dựng giao diện người dùng cho hệ thống để đưa hệ thống đến với người dùng. Chúng tôi mong muốn quá trình chẩn đoán thông qua hình ảnh y khoa được thực hiện một cách nhanh chóng và chính xác thay vì phương pháp thủ công trước đây.

¹ End-to-end là khái niệm dùng để chỉ một hệ thống có khả năng tiếp nhận dữ liệu thô ở đầu vào và cho ra kết quả cuối cùng ở đầu ra, mọi công đoạn trung gian được thực hiện một cách tự động.

A

THÔNG SỐ TẬP DỮ LIỆU 3D-IRCADB-01

Tables A.1 trình bày thông số chi tiết về tập dữ liệu 3D-IRCADb-01. Bảng thông số này được lấy từ trang chủ của Viện nghiên cứu chống ung thư đường tiêu hoá IRCAD.

Tables A.1: Thông tin về tập dữ liệu 3D-IRCADb-01 (Nguồn: [18]).

STT	Giới tính	Kích thước voxel (mm)	Kích thước ảnh (pixel)	Kích thước gan (cm)	Bệnh lý gan	Trở ngại phân đoạn
1	Nữ	0,57	512	18,3	7 khối u (III, IV/V, VII, VIII)	Dạ dày
		0,57	512	15,1		Tụy
		1,6	129	14,1		Tá tràng
2	Nữ	0,78	512	20,1	1 khối u (V)	Tụy
		0,78	512	16,9		
		1,6	172	15,7		Tá tràng
3	Nam	0,62	512	16,7	1 khối u (IV)	Kim loại
		0,62	512	14,9		
		1,25	200	15,2		
4	Nam	0,74	512	16,9		
		0,74	512	12,0	7 khối u	Tim
		2	91	17,2		
5	Nam	0,78	512	19,8		Cơ hoành
		0,78	512	18,6	0 khối u	
		1,6	139	19,1		Tá tràng
6	Nam	0,78	512	18,8		
		0,78	512	14,3	20 khối u	Tim
		1,6	135	20,2		
7	Nam	0,78	512	24,9		
		0,78	512	15,2	0 khối u	Lá lách
		1,6	151	16,6		
8	Nữ	0,56	512	23,5		
		0,56	512	17,1	3 khối u (I, II, IV)	
		1,6	124	12,5		–

Tables A.1: Thông tin về tập dữ liệu *3D-IRCADb-01* (Tiếp).

STT	Giới tính	Kích thước voxel (mm)	Kích thước ảnh (pixel)	Kích thước gan (cm)	Bệnh lý gan	Trở ngại phân đoạn
9	Nam	0,87	512	20,6	1 khối u ở V/VIII	Dạ dày Ruột già
		0,87	512	17,0		
		2	111	18,1		
10	Nữ	0,73	512	18,4	8 khối u	Tá tràng Tuy
		0,73	512	15,5		
		1,6	122	14,8		
11	Nam	0,72	512	19,1	0 khối u	Dạ dày Tuy
		0,72	512	14,4		
		1,6	132	16,2		
12	Nữ	0,68	512	19,3	1 khối u VI	–
		0,68	512	17,7		
		1,0	260	11,0		
13	Nam	0,67	512	20,0	20 khối u	Tá tràng Tim
		0,67	512	12,9		
		1,6	122	18,1		
14	Nữ	0,72	512	22,4	0 khối u	Lá lách Tuy
		0,72	512	15,4		
		1,6	113	13,4		
15	Nữ	0,78	512	18,8	2 khối u (II, VIII)	Tuy
		0,78	512	17,7		
		1,6	125	14,7		
16	Nam	0,70	512	20,2	1 khối u (V)	Cơ
		0,70	512	17,7		
		1,6	155	20,2		
17	Nam	0,74	512	19,8	2 khối u (II, VIII)	Dạ dày Tim
		0,74	512	17,4		
		1,6	119	18,9		
18	Nữ	0,74	512	22,5	1 khối u (IV/V)	Thực quản Cơ
		0,74	512	15,1		
		2,5	74	18,6		
19	Nữ	0,70	512	19,5	46 khối u	Dạ dày Tá tràng Tuy
		0,70	512	16,5		
		4	124	14,2		
20	Nữ	0,81	512	20,0	0 khối u	Dạ dày
		0,81	512	16,6		
		2	225	16,8		

B

KẾ HOẠCH THỰC HIỆN LUẬN VĂN

Để có thể hoàn thành luận văn một cách tốt nhất, ngay từ những ngày đầu tiên, chúng tôi đã xây dựng kế hoạch cụ thể cho giai đoạn Đề cương luận văn cũng như giai đoạn Luận văn và đã thực hiện đúng theo kế hoạch đề ra. Đó là một trong những cơ sở quan trọng để chúng tôi có thể thực hiện thành công luận văn này. Chi tiết kế hoạch thực hiện luận văn được chúng tôi mô tả ở Figure B.1.

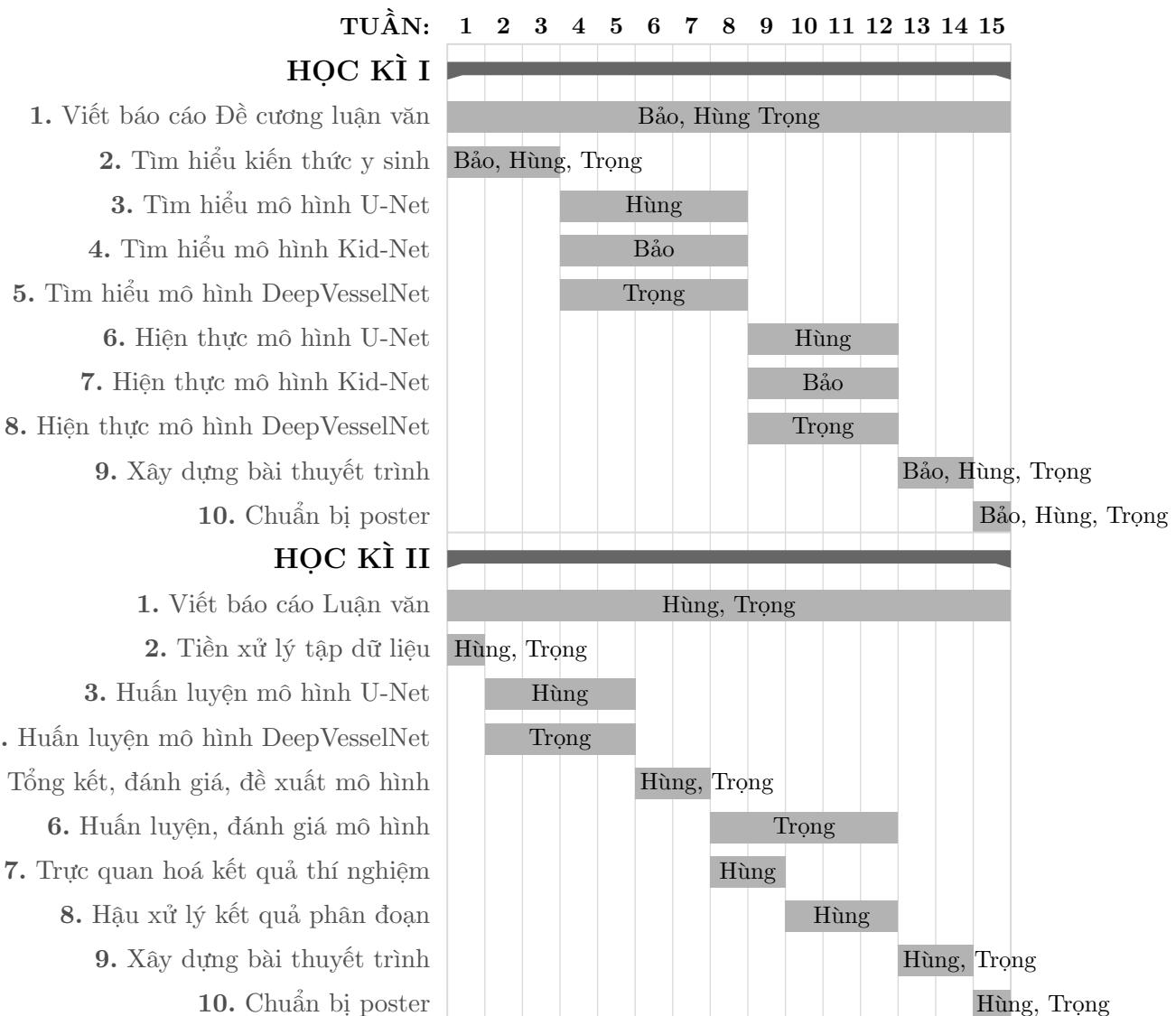


Figure B.1: Kế hoạch thực hiện luận văn.

BIBLIOGRAPHY

- [1] Stanford University. “Cs231n convolutional neural networks for visual recognition.” (2014), [Online]. Available: <http://cs231n.github.io/convolutional-networks/> (visited on 03/02/2019).
- [2] H. Le and A. Borji, “What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks?” *arXiv preprint arXiv:1705.07049*, 2017.
- [3] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [4] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
- [5] V. H. Tiếp. “Bài 13: Softmax regression.” (2017), [Online]. Available: <https://machinelearningcoban.com/2017/02/17/softmax/> (visited on 03/02/2019).
- [6] A. R. Weeks, *Fundamentals of electronic image processing*. SPIE Optical Engineering Press Bellingham, 1996, pp. 316–332.
- [7] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. “Thinning.” (2003), [Online]. Available: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm> (visited on 03/02/2019).
- [8] T.-C. Lee, R. L. Kashyap, and C.-N. Chu, “Building skeleton models via 3-d medial surface axis thinning algorithms,” *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994.
- [9] S. Chintala. “Pytorch release.” version 1.0.1. (2019), [Online]. Available: <https://github.com/pytorch/pytorch/releases/tag/v1.0.1> (visited on 03/02/2019).
- [10] A. Joshi and V. Geetha, “Sql injection detection using machine learning.,” Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference, 2014, pp. 1111–1115.
- [11] *Sqlmap*, <https://sqlmap.org/>, Accessed: 2022-09-30.
- [12] R. Rawat and S. Shrivastav, “Sql injection attack detection using svm.,” *International Journal of Computer Applications*, pp. 1–4, 2012.
- [13] F. Mereani and J. Howe, “Detecting cross-site scripting attacks using machine learning,” in Jan. 2018, pp. 200–210, ISBN: 978-3-319-74689-0. DOI: [10.1007/978-3-319-74690-6_20](https://doi.org/10.1007/978-3-319-74690-6_20).

- [14] A. Laughter, S. Omari, P. Szczurek, and J. Perry, “Detection of malicious http requests using header and url features,” in Jan. 2021, pp. 449–468, ISBN: 978-3-030-63088-1. DOI: [10.1007/978-3-030-63089-8_29](https://doi.org/10.1007/978-3-030-63089-8_29).
- [15] A. Alshammari and A. Aldribi, “Apply machine learning techniques to detect malicious network traffic in cloud computing,” *Journal of Big Data*, vol. 8, no. 1, p. 90, Jun. 2021, ISSN: 2196-1115. DOI: [10.1186/s40537-021-00475-1](https://doi.org/10.1186/s40537-021-00475-1). [Online]. Available: <https://doi.org/10.1186/s40537-021-00475-1>.
- [16] M. Alsaedi, F. A. Ghaleb, F. Saeed, J. Ahmad, and M. Alasli, “Cyber threat intelligence-based malicious url detection model using ensemble learning,” *Sensors*, vol. 22, no. 9, 2022, ISSN: 1424-8220. DOI: [10.3390/s22093373](https://doi.org/10.3390/s22093373). [Online]. Available: <https://www.mdpi.com/1424-8220/22/9/3373>.
- [17] L. N. M. Khoi, “Machine learning approaches to cyber threats detection.,” Jan. 2021.
- [18] IRCAD France. “3d-ircadb 01.” (2009), [Online]. Available: <https://www.ircad.fr/research/3d-ircadb-01/> (visited on 02/03/2019).
- [19] Medical Connections. “Hounsfield units.” (2010), [Online]. Available: <https://www.medicalconnections.co.uk/kb/Hounsfield-Units/> (visited on 03/02/2019).
- [20] D. V. Duy, L. T. Sach, and E. K. Karuppiah, “Medical images sequence normalization and argumentation: Improve liver tumor segmentation from small dataset,” 2017.
- [21] B. H. T. Nhat and P. H. Son, “Apply deep learning in tumor detection in ct images,” Engineer’s Thesis, Ho Chi Minh City University of Technology, 2019, pp. 50–51.

DANH MỤC TỪ KHOÁ

Danh sách dưới đây liệt kê các từ khoá được đề cập trong nội dung luận văn này theo thứ tự bảng chữ cái.

B	
Background	24, 25, 42, 49
Backward	15
Batchnorm	9, 50, 51
C	
Convolution	7, 8, 10, 40
D	
Dilation	12
E	
Erosion	12
F	
Filter	8
Foreground	24, 25, 42, 49
Forward	15
G	
Gradient	10, 15
H	
Hounsfield	21
I	
Inference	50
Intensity	21
L	
Learning rate	9, 50–53
M	
Left limit	41, 42
N	
Max-pooling	7
Mean	9
O	
Overfitting	7, 69
P	
Padding	8
Pooling	7
PyTorch	15, 44
R	
Receptive field	8
ReLU	10
Right limit	41, 42
S	
SGD	10
Shift-invariant	12, 13
Sigmoid	10
Skeleton	14, 68
Slicer	16, 46
Softmax	11
Stride	8
V	
VTK	16, 46