**基于TensorFlow-AE的肝脏分割与淋巴瘤检测系统的设计与实现**

**姓 名 于恒健**

**学 院 软件学院**

**专 业 软件工程（国际班（英语））**

**指导教师 姜慧研 张宏**

**备 注**

**2017 年 6 月9日**

Design and Implementation of Liver Segmentation and Lymphoma Detection System Based on TensorFlow

by Yu Hengjian

|  |  |  |
| --- | --- | --- |
| Supervisor: | Professor | Jiang Huiyan |
| Associate Supervisor: | Engineer | Zhang Hong |

Northeastern University

June 2017

基于TensorFlow-AE的肝脏分割与淋巴瘤检测系统的设计与实现

|  |  |  |
| --- | --- | --- |
| 作者姓名： | 于恒健 | |
| 校内指导教师： | 姜慧研 | 教授 |
| 校外指导教师： | 张宏 | 工程师 |
| 单位名称： | 软件学院 | |
| 专业名称： | 软件工程 | |

东 北 大 学

2017年6月

**毕业设计（论文）任务书**

|  |
| --- |
| **毕业设计（论文）题目：** |
| 基于TensorFlow-AE的肝脏分割与淋巴瘤检测系统的设计与实现 |
| **基本内容：**  研究深度学习技术在医学图像处理方面的应用，熟练掌握与运用Python，MATLAB，C++等编程语言以及PyCharm，PDB等开发工具。认真分析淋巴癌PET-CT图像特征分析的需求，设计并实现PET图像预处理，实现线性谱聚类算法并对图像进行分割，应用深度卷积网络提取特征等基本功能，并能对所实现的部分进行测试和评价。  翻译一篇与毕设内容相关的外文资料，译文汉字字数不少于4000字。 |
| **毕业设计（论文）专题部分：**  **题目：**  **基本内容：** |
| **学生接受毕业设计（论文）题目日期**  **第 1 周**  **指导教师签字：jhy 校外导师签名**  **2017年 3 月 10 日** |

**Design and Implementation of Liver Segmentation and Lymphoma Detection System Based on TensorFlow**

**Abstract**

Feature extraction is an important prerequisite for image analysis and computer-aided diagnosis. It is hard to extract the hidden feature in medical images, which are complex and difficult to represent. With the rapid development of deep learning technology, it is important to use deep learning methods to proceed automatic medical image feature extraction.

Firstly, linear spectral clustering algorithm is applied in this paper to extract multi-scale superpixels. Deep convolutional neural network (DCNN) based on TensorFlow is integrated to extract features from superpixels. Secondly, the uptake of fludeoxyglucose (FDG) of liver is high in PET images, which impact on the false positive percentage of lymphoma diagnosis. Considering this issue, in order to reduce the false positive, a stacked autoencoder (SAE) is applied to identify the liver region and lymphoma area. The feature extraction and classification are implemented as web services by using Flask, which is a web microframework. The design of the system is based on MVC architecture in order to improve the portability of the system. Moreover, a distributed storage mechanism is implemented to maintain data integrity and improve the storage performance.

In order to evaluate the effect of feature extraction, this paper also implement a stacked autoencoder based on TensorFlow, which is used to classify the feature vectors extracted in the system. Totally 51,535 superpixels are used as train set and test set, half of them are the train set, and the rest of them are test set. In the case the of three-layered SAE, the liver recognition accuracy is 88.9%, and the lymphoma region accuracy is 96.2%. The result shows the system has good reliability and extensibility, which is of great significance for optimizing the recognition effect of medical images.

**Key words:** TensorFlow, CNN, Stacked Autoencoder, Feature Extraction, Medical Image Processing

基于TensorFlow-AE的肝脏分割与淋巴瘤检测系统的设计与实现

摘 要

特征提取是图像分析与计算机辅助诊断的重要前提，医学图像的复杂性较高，其特征表示与描述的难度较大，人工提取特征难以揭示医学图像内部所隐含的全部特点。随着深度学习技术的迅猛发展，基于深度学习的医学图像特征的全自动提取对挖掘医学图像更深层次特征具有重要的研究意义与应用价值。

本文首先基于线性光谱聚类算法提取多尺度的超像素；然后基于TensorFlow深度卷积神经网络（DCNN）从超像素中提取特征；由于肝脏对FDG的摄取值较高而影响PET图像的诊断，因此本文基于堆栈式自编码技术识别肝脏区域和淋巴瘤超像素区域，以降低由于肝脏带来的假阳性问题，提高诊断的准确率。本文将特征提取部分和基于堆栈式自编码技术识别部分以Web服务的方式实现于Flask轻量框架上，并实现了基于MVC的三层结构，提高了系统的可移植性。本系统基于数据库集群技术，构建了分布式存储子系统，以保持数据完整性并提高存储性能。

为了验证特征提取的效果，本文也使用TensorFlow实现了堆栈式自编码器，并使用多种堆栈式自编码器对本工程提取出的特征向量进行分类以评估特征提取的效果。测试中使用了51535块超像素充当测试集和训练集，选取1/2作为训练集，其余为测试集。在采用三层堆栈式自编码器的情况下，测试中肝脏和淋巴癌的准确率分别达到了88.9%和96.2%。测试结果表明，此系统具有良好的可扩展性和可靠性，对于优化医学图像的识别效果具有重要的意义。

**关键词：**TensorFlow，卷积神经网络，堆栈式自编码器，特征提取，医学图像处理

**Contents**

[**Abstract** I](#_Toc485025349)

[摘 要 II](#_Toc485025350)

[**Chapter 1 Introduction** 1](#_Toc485025351)

[1.1 Research Background 1](#_Toc485025352)

[1.2 Current Research Status 2](#_Toc485025353)

[1.3 Research Work 2](#_Toc485025354)

[1.4 Thesis Organization 3](#_Toc485025355)

[**Chapter 2 Key Technologies** 5](#_Toc485025356)

[2.1 TensorFlow 5](#_Toc485025357)

[2.1.2 System Architecture 5](#_Toc485025358)

[2.1.2 Tensors 5](#_Toc485025359)

[2.1.3 Computational Graphs and Nodes 6](#_Toc485025360)

[2.1.4 Session 6](#_Toc485025361)

[2.2 PyCharm 6](#_Toc485025362)

[2.2.1 Why Choose PyCharm 7](#_Toc485025363)

[2.2.2 Main Functionalities of PyCharm 8](#_Toc485025364)

[2.3 Summary 9](#_Toc485025365)

[**Chapter 3 Requirements Analysis** 11](#_Toc485025366)

[3.1 Overview 11](#_Toc485025367)

[3.2 Use Case Analysis 11](#_Toc485025368)

[3.2.1 Segmentation Module 11](#_Toc485025369)

[3.2.2 Feature Extraction Module 15](#_Toc485025370)

[3.2.3 Classification Module 16](#_Toc485025371)

[3.2.4 User Module 17](#_Toc485025372)

[3.3 Feasibility Analysis 18](#_Toc485025373)

[3.4 Summary 19](#_Toc485025374)

[**Chapter 4 System Design** 21](#_Toc485025375)

[4.1 Architecture Design 21](#_Toc485025376)

[4.1.1 System physical architecture 21](#_Toc485025377)

[4.1.2 System logic architecture 25](#_Toc485025378)

[4.2 Key Module Design 26](#_Toc485025379)

[4.2.1 Feature Extraction Module Design 26](#_Toc485025380)

[4.2.2 Classification Module Design 30](#_Toc485025381)

[4.2.3 Segmentation Module Design 34](#_Toc485025382)

[4.2.4 Login Module Design 36](#_Toc485025383)

[4.3 Database Design 37](#_Toc485025384)

[4.4 Summary 38](#_Toc485025385)

[**Chapter 5 System Implementation** 39](#_Toc485025386)

[5.1 Development Environment 39](#_Toc485025387)

[5.2 Implementation of Feature Extraction Module 39](#_Toc485025388)

[5.3 Implementation of Classification Module 43](#_Toc485025389)

[5.4 Implementation of Segmentation Module 45](#_Toc485025390)

[5.5 Implementation of Login Module 46](#_Toc485025391)

[5.6 Summary 47](#_Toc485025392)

[**Chapter 6 System Testing** 49](#_Toc485025393)

[6.1 Basic Test Methods Introduction 49](#_Toc485025394)

[6.2 System Key Module Testing 49](#_Toc485025395)

[6.3 Classification Results 51](#_Toc485025396)

[6.4 Summary 52](#_Toc485025397)

[**Chapter 7 Conclusion** 55](#_Toc485025398)

[7.1 Summary of the Work 55](#_Toc485025399)

[7.2 The Future Work 55](#_Toc485025400)

[**References** 57](#_Toc485025401)

[**Acknowledgements** 59](#_Toc485025402)

**Chapter 1 Introduction**

## 1.1 Research Background

Nowadays the deep learning methods are popular around the medical image process region. The deep leaning, contrast to the ordinary machine learning methods, has a lot of difference. For example, the deep leaning method always involves a complex weighted neural network such as AlexNet [1] and VGGNet [2]. They are usually widely used in image recognition area, image classification and segmentation. The neural network mentioned above got good results on the 2014 ILSVRC localization and classification subject. The common feature of neural networks is that they are usually very deep. There are 16 to 19 weighted layers to construct the whole neural network. As usual, these networks use an alternative procedure of convolutional layers and pool layers.

The traditional machine learning method can also handle this kind of medical image process work, but there are limitations. First it requires a group of experienced medical programmers in this field to analyze algorithm and procedure from the actual medical data or works. Some classical algorithms are used in the work such as SVM [3].To emphasize, this kind of method is dominated around this area for many years. The next procedure to optimize result is usually involved a set of high-complexity algorithm to handle the input or output of the procedure. Due to the subjective nature of the image, the way of manual processing in the medical image feature extraction area is likely to have unknown bugs, and the most important matter is that the process of writing algorithms is mainly depends on the past experience.

Thus, the deep learning method has advantage over the classical algorithm mechanism mentioned above. It is a kind of method of imitating the biological neural processing signal model, which means that the programming of the deep leaning does not require the explicit programming. Classical algorithms require a kind of implicit programming with train step in the constructed neural network. The deep leaning network can learn the feature automatically, avoiding the procedure of writing algorithm mentioned above to increase the effectiveness and speed of the medical image feature extracting.

## 1.2 Current Research Status

Archive [4] uses the deep neural network on three-dimensional CT image segmentation, the segmentation result performs well on large organs. But for the small size of organs and tissue, the result is not ideal, which is mainly due to the low volume of medical images and the fact that most of medical images are with the low resolution Therefore, increasing the resolution of the medical images should the first work required to improve the final recognition result. In a view of applying all the low-resolution images directly to the deep neural network, the appropriate pre-processing becomes a way to improve the accuracy of the deep neural network. The use of graph cutting method with deep neural network can improve the performance of the network in a certain context [5]. In a project of identifying lymphoma from PET-CT images, the archive [6] uses the LSC algorithm [7] to process the image, avoiding the neural network extracts some unnecessary deep logic and reduce the probability of excessive leaning by the deep neural network. However, relying on these methods makes the algorithm complex and is not easy to integrate. So, if used for a variety of range of medical system, it should be having some practical limitations.

## 1.3 Research Work

In this project, tensorflow is used to extract the feature of some processed segmented superpixels. The source data are derived from a local hospital.

Because the tensorflow framework requires structed data to build the deep neural network, the source image data must be processed to fit the function requirements of TensorFlow. Based on the library of numeric matrix mathematic processing and dicom format, the pre-processing can be implemented with scripts.

The linear spectral cluster algorithm is applied into the processed PET image, using the color parameter and the pixel number parameter to control the final superpixel number and the ratio of respecting the object edge and the superpixel shape. There should be a multi-scale segmentation to capture all kinds of organs and tissue.

By using VGG-19 net, the feature extraction should be easy to implement, and the extracted feature could be different level of the middle network output. The weights can be downloaded from web site.

The final check procedure of the project is classification methods. The autoencoders should be used to extract the hidden feature used for classification.

## 1.4 Thesis Organization

**Chapter 1** Introduction. This chapter includes the he Research background, current research status and research work.

**Chapter 2** Key Technologies. The first part of this chapter introduces the Tensorflow framework, system architecture of TensorFlow, tensors, graphs, nodes, session. The next part mainly introduces the PyCharm IDE.

**Chapter 3** Require Analysis. This chapter analyzes the use cases of each module, includes the segmentation module, feature extraction module, the classification module and the user module. Feasibility analysis is also included.

**Chapter 4** System Design. This chapter introduces the system physical architecture, logical architecture, and the key module design.

**Chapter 5** System Implementation. This chapter mainly introduces project development environment, the implementation of feature extraction module, classification module, segmentation module and the login module.

**Chapter 6** System Testing. The Basic test method and the test case are included in the previous part. The classification results are the second part of this chapter.

**Chapter 7** Conclusion. The conclusion of the whole project, from system requirement to the GUI design and the actual implementation of each algorithm. and the shortage need to be refined in the future work.

**Chapter 2 Key Technologies**

## 2.1 TensorFlow

TensorFlow is an open-source package which process numerical computation, especially tensors, utilizing with data flow graphs. The TensorFlow programs can be deployed with CPU or GPU as usual on producible server even in a home laptop for mobility.

### 2.1.2 System Architecture

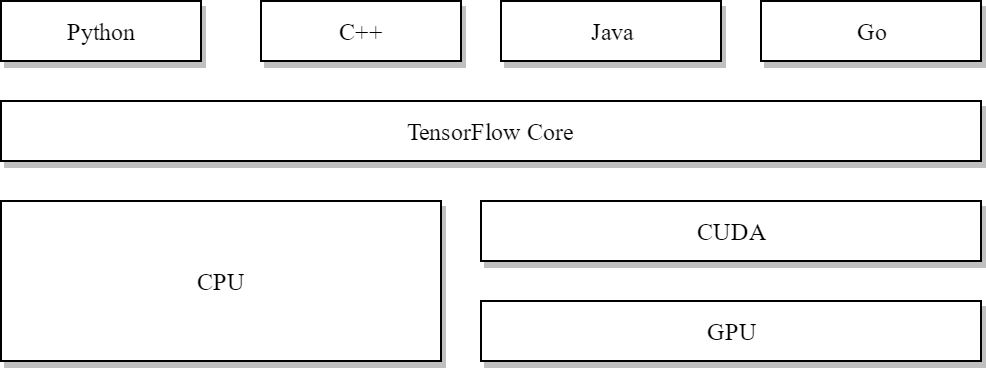


Figure 2.1 Main Architecture of TensorFlow

As shown in figure 2.1, the main architecture of TensorFlow consists of serval parts: front-end layer, core execution layer, and then the CUDA and lower hardware or CPU. Programmers can use any one of the front-end languages to define the calculation.

The design abstracts the hardware differences, making it possible to write a generic code similar to each other with serval supported front-end languages supported such as Python, C++, Java and Go. One caution that have to consider is that the front-end languages, also called APIs in languages, only the Python APIs are covered by API stability promises.

### 2.1.2 Tensors

The massive use of tensors in this framework is the reason why it’s named after TensorFlow. A Tensor is a sequential set of the same type of values. Not the same as vector which only has length, tensor has all the shape and datatype. There is also an import definition of the vector: the rank of tensor. The rank of tensor is the number of its dimensions. Because the tensor is relatively broadly defined, making it necessary to pose serval examples of different rank of tensors. Such as, a number of 5 is also a tensor of rank zero., and its shape is empty.

### 2.1.3 Computational Graphs and Nodes

A typical usage of computing values, specifically as tensors, is to build a computation graph that consists of many nodes. To simplify the usage the framework, a default graph is employed whatever computation you defined in the script. That means, if do not explicitly declare the graph, the framework is still workable. Nodes is another concept that consists the whole graph. It draws on the research result from the biomedical field on the structure of neurons. A node accepts a tensor as input and the apply operations stores in the node on it then produce the output, explicitly, a tensor. The type of nodes is defined by the actual usage of tensors: constant, variable and placeholder. A constant accepts no tensor as input but produce a constant tensor as output. The type of variable is like constant, but its output value is changed depends on the input or inputs, and its value can be changed at runtime.

### 2.1.4 Session

Above introduction of computational graphs is just an abstract concept that cannot be directly implemented onto hardware. The mainstream hardware frequently applied to this circumstance are nvidia graphic cards, and the highest interface it provides is CUDA library and C++ is required to execute or compile executable programs. Session contains the rules and strategy which how a graph is built for launching and the actually runtime status of the compiled graph. For instance, it is the responsibility that the session detects if it is under stand-alone mode or cluster mode. And if the program compiles at a cluster mode, the maintenance of the cluster should be under control of the session.

## 2.2 PyCharm

PyCharm is a powerful integrated development environment (IDE) which can be used in many circumstances from single-person development to agility team development, as we know, even it can do a job with multiple teams in close cooperation with hard task.

### 2.2.1 Why Choose PyCharm

In general, the use of Eclipse as a python IDE is still possible. But here are some reasons that Eclipse cannot be as powerful as PyCharm.

Eclipse was originally used to write java programs and then some collaborative working groups came out that if it is possible to use eclipse to write python programs. Fortunately, the main framework of eclipse is highly adjustable and equipped with the force that change the behavior of reacting with the user input by sub-classing the abstract classes. In that case, the researchers and the main works of the Eclipse foundation came out the possible solution of the editor behavior and the auto code completion, which is the essential part of a workable IDE accepted by many programmers. But that means there are much like between the original Eclipse and the Eclipse python version. Unfortunately, the original Eclipse IDE is a part of out-of-date designed, which is especially suitable for developing recent years ago.

The Eclipse provides the functionality of auto code completion and code review by use a plugin called pydev. Pydev is not in the original plugin package makes everyone wants use the pydev functionality have to go to the official website to download and install trough the unstable network. Especially in the network congesting or frequent occurrence of the network jitter in the region, means that the network infrastructure of network settings out of the problem may lead to eclipse plugin installation failed. Many people not only need to take the time to learn how to install pydev plug-ins through the network, but also because of above reasons wasted time on the network I/O.

By the way, the smart assistance is robust and available for all kinds of methods, functions, variables, constants, even in comments. But the recent eclipse only has the similar speed of providing a part of the functionality that PyCharm already has. That mean with more work to be done, especially with the intelligence name completion and error checking, JetBrains does better than the Eclipse Foundation.

The JetBrains Company is fully responsible for the development and the maintenance of the PyCharm IDE, and it is a commercial software. That means not only the software they charged, but the possible technical solutions they have are here available for customers to purchase. The community support, as we all know, sometimes it can be timely but most of cases it is not timely. That is the relative advantages of PyCharm relative to a community product.

### 2.2.2 Main Functionalities of PyCharm

Power save mode is a unique and powerful option in the file tab. It is a default closed feature after the user installation. The main use of this option is that it can be config to stop the battery-hungry behavior running as services at the background processes/threads. Generally speaking, the power save mode integrated in PyCharm is as powerful as other IDEs build by JetBrains and is useful if the developer really need to use IDE as a status of limited power supply. With this enabled, it can save a large quantity of CPU resources and memory occupation. The way to proceed this is maintain the auto-checking the syntax errors. Once you typed a character in the default editor of the main panel, the auto-checking functionality will run a full scan with your current editing file, in order to highlight the syntax errors which you probably made it with careless. But other battery consumed feature is not enabled at this mode, for example, if there are some if-clauses, it will not check the type or correctness by executing the block of code in any part of the if-clauses. Though without the auto executing feature may make the editor less intelligence, but it can be solved by occasionally by just switching the power save mode for a while, or just proceed to the functionality provided by the manual code inspecting, which is an option exists at code tab.

The integrated version control system provides a variety of choices the project can use in the VCS tab, such as git, CVS and SVN. In a pretty small project the standalone version control is suitable until it expands to the volume that anyone cannot realize or understand all the main functionalities it contains, or the modules it has. Then the integration with the version control systems like git can help this situation a lot. The most usable dialogs in the integrated system in git corresponds to the main keywords typed in the console mode. The update project dialog of git can be controlled by typing the arguments according to the format ruled by lots of Linux distributions, but also can be config by choosing the checkbox or button the IDE provides. A good review and processed information will help a lot.

Debug system is a widely-provided feature most IDE provide. Like an ordinary debugger, firstly developer has to set breakpoints in the code. These breakpoints will cause the insertion of a non-visibly code after the line you set the breakpoint, and with the right debug configuration, the internal python debug bridge (PDB) provides the main features what a debugger has.

Other useful functionalities like locate duplicates can inspect the code you wrote not the syntax error or possible simple logic mistake, but the block-wise duplication you done probably with the copy-paste strategies. The concurrency window will visualize the concurrent executing by the multiple threads/processes.

## 2.3 Summary

This chapter mainly introduces the tools of this project, which includes the tensorflow library in python programming language and the python IDE called PyCharm. The tensorflow library includes session, graph, tensor, node, and other component for high level abstract lazy computing, and there are many useful tools are integrated into the PyCharm. The next chapter is to introduce the use case analysis and system requirement.

**Chapter 3 Requirements Analysis**

## 3.1 Overview

This project is mainly focus on the feature extraction and classification from medical data. For the convenient of upgrading the system, it is essential to equip with the functionality that allows user to access the system from browser rather than a specific local GUI because the local GUI libraries are usually bumpy to optimize for upgrading.

Due to the scarcity of medical image data, the Feature extraction must be easy to build, with powerful and certified behavior when handling for small amount of data. Then the VGG-19 net is in my scope because its depth is clearly enough for processing the medical image data comparing build a deep convolutional net with personal efforts. What’s more, there are weights trained for natural images can be used directly with this.

The inspection of the feature extraction can be fulfilled by utilizing the classification. This project uses supervised learning, with the output feature from above proposed methods. The effectiveness of the model is usually measured using accuracy.

Last but not the least, since the service is open to the network, whatever what kind of network it is. In order to facilitate the decoupling, it is appropriate to take the unauthorized access under control. Therefore, the login and logout functionality is also required when building the system.

## 3.2 Use Case Analysis

The system is mainly focuses on the design of Linear Spectral Clustering module and Feature extraction module. The use case analysis of each main module is shown below.

### 3.2.1 Segmentation Module

The use case of linear spectral clustering (LSC) segmentation module is mainly designed for producing many small images called superpixel. The actual cases are shown in figure 3.1.

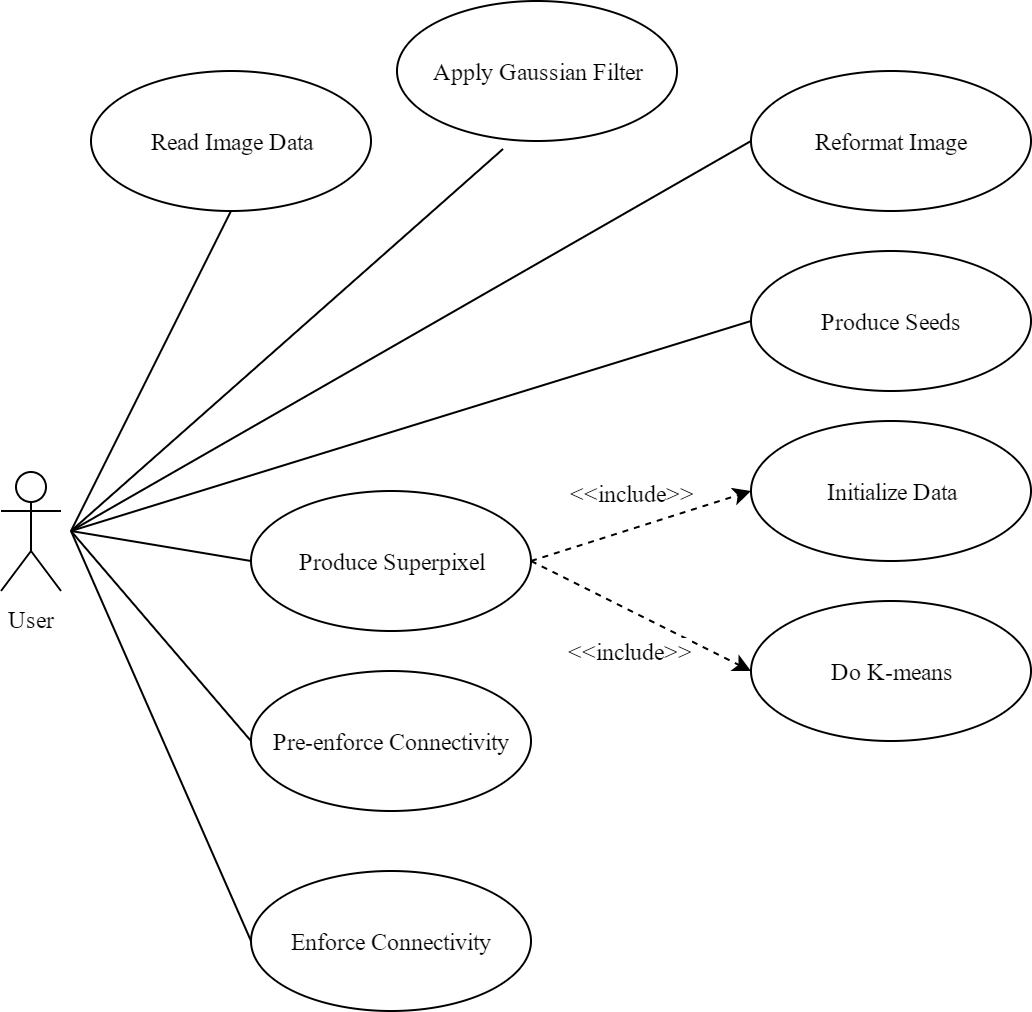


Figure 3.1 Use Case of Segmentation module

Table 3.1 shows the detailed description of Read Image Data. The image data can be loaded by internal Python image library such as PIL, or external 3rd-party libraries like OpenCV.

Table 3.1 Use case description of Read Image Data

| **Use Case No.** | 1 |
| --- | --- |
| **Brief Description** | Load Image through PIL (Python Image Library, imported as pillow) |
| **Pre-Condition** | The exact user logged in the system |
| **Post-Condition** | The user gets the matrix representation of one image |
| **Flow of Event** | 1. User select the path/location to be processed  2. Load the image through PIL  3. Record the detail of this processing |

Table 3.2 shows the detail of use case of Apply Gaussian Filter. The main idea of apply Gaussian filter is that the LSC is sensitive to inconsistence of color. Thus, operation on the filtered image is more stable in general.

Table 3.2 Use case description of Apply Gaussian Filter

| **Use Case No.** | 2 |
| --- | --- |
| **Brief Description** | Apply Gaussian Filter (3x3 Gaussian filter) to the image data |
| **Actors** | User |
| **Pre-Condition** | The image data have to be loaded without error |
| **Post-Condition** | The User got the filtered image |
| **Flow of Event** | 1. User select image to filter  2. Generate the pre-defined 3x3 Gaussian filter using function fspecial  3. Proceed the correlate operation between the image data and filter  4. Change the type filter data to integer-like matrix |

Table 3.3 shows the detailed description of Reformat Image. RGB color performs inconsistency when the spectral smoothly changes. The better choice is reformatting the RGB image to LAB format image.

Table 3.3 Use case description of Reformat Image

| **Use Case No.** | 3 |
| --- | --- |
| **Brief Description** | Change the image format from RGB to LAB |
| **Actors** | User |
| **Pre-Condition** | The image data should be filtered by Gaussian filter |
| **Post-Condition** | The user gets the reformatted LAB image data |
| **Flow of Event** | 1. User select image to transpose format  2. Calculate XYZ value of each pixel data in the image  3. Calculate LAB value of each pixel data from XYZ value |

Table 3.4 shows the detailed description of Produce Seeds. The number and initial position of seeds depends on the size of image (width, height) and the number of superpixel arguments specified.

Table 3.4 Use case description of Produce Seeds

| **Use Case No.** | 4 |
| --- | --- |
| **Brief Description** | Generate the initial seeds according to parameters set by user |
| **Actors** | User |
| **Pre-Condition** | The image data should be changed to LAB format |
| **Post-Condition** | User get the default seeds, control by argument settings |
| **Flow of Event** | 1. Calculate the initial superpixel width and height value  2. Allocate the x-axis value and y-axis value to seeds |

Table 3.5 shows the detailed description of Produce Superpixel. This is the most important procedure in the segmentation dataflow. A weighted k-means algorithm is applied to clustering high-dimensional image data. One thing that matters is that the total number of types of values are ten, include the cosine and sine of each channel of the image and the same procedure of row position and column position of each pixel. The weights used by the weighted k-means is generated by summing all the channels up.

Table 3.5 Use case description of Produce Superpixel

| **Use Case No.** | 5 |
| --- | --- |
| **Brief Description** | Generate Superpixel with the edge-detection and color-sensitive feature |
| **Actors** | User |
| **Pre-Condition** | The image data should be along with generated seeds description |
| **Post-Condition** | User get the primitive generated superpixel |
| **Flow of Event** | 1. Pre-treatment the superpixel program  2. Initialize the 10-dimension data from LAB data  3. Get weight from 10-dimension data  4. Do k-means cluster on the weighted 10-dimension data |

Table 3.6 shows the detailed description of Initialize Data. This is the essential procedure that map the original image pixel data to more image data. The initialize procedure is to generate the multi-dimensional data from the color data and position data of each pixel. This diagram shows the channels’ detail name and its meaning. The weight mentioned before is also generated from this case.

Table 3.6 Use case description of Initialize Data

| **Use Case No.** | 6 |
| --- | --- |
| **Brief Description** | Generate 10-dimension data and weights |
| **Actors** | User |
| **Pre-Condition** | The data of LAB image has to be extracted |
| **Post-Condition** | Get the high dimension data and weights |
| **Flow of Event** | 1. Get the L, A, B, X, Y channel data from the image  2. Apply trigonometric function to L, A, B, X, Y channels  3. Generate L1, L2, a1, a2, b1, b2, x1, x2, y1, y2 from LABXY channel  4. User get the high-dimensional data and weights |

Table 3.7 shows the detailed description of Do K-means. The iteration number of k-means can be fixed because the purpose of this project is handling a typical type of medical image. The initial value of the weighted k-means is decided by the default seeds, which is arithmetic average area of the image. The more iterations it proceeds, the better segmentation result is. Due to limitation time and memory space, the value of iteration number should be carefully chosen. The low value of iteration number can lead to the failure of next procedure. During the iteration, the procedure of adjusting the center of each cluster and the area of each cluster is changed once.

Table 3.7 Use case description of Do K-means

| **Use Case No.** | 7 |
| --- | --- |
| **Brief Description** | Do k-means algorithm on high dimension data |
| **Actors** | User |
| **Pre-Condition** | The high dimension data has to be extracted from original image |
| **Post-Condition** | User get the primary result of superpixel segmentation |
| **Flow of Event** | 1. Adjust the area of each superpixel  2. Adjust the center of each superpixel  3. Do step 1 until reach the iteration number  4. User get the generated label masks |

Table 3.8 shows the detailed description of Pre-enforce Connectivity. The main idea of pre-enforce connectivity is that the small superpixels could be useless and can be merged to neighbor superpixels in order to enhance the fluency of segmentation result. The condition of an image without the pre-enforce connectivity is analyzed in the next chapter.

Table 3.8 Use case description of Pre-enforce Connectivity

| **Use Case No.** | 8 |
| --- | --- |
| **Brief Description** | Merge small superpixels to the nearest superpixel |
| **Actors** | User |
| **Pre-Condition** | The superpixels must be generated |
| **Post-Condition** | User get the merged, connected superpixel |
| **Flow of Event** | 1. Initialize the searched region of each superpixel  2. Search the superpixel and then decide which super should be merged  3. Merge the small superpixels to the nearest neighbor superpixel |

Table 3.9 shows the detailed description of Enforce Connectivity. This is the last main procedure finishing the dataflow of segmentation.

Table 3.9 Use case description of Enforce Connectivity

| **Use Case No.** | 9 |
| --- | --- |
| **Brief Description** | Present successive superpixel numbers |
| **Actors** | User |
| **Pre-Condition** | The user ought to gain the image matrix and mask of pre-enforce |
| **Post-Condition** | User get the final result of linear spectral cluster algorithm |
| **Flow of Event** | 1. Calculate the high dimensional distance along each superpixel  2. Compare the relative distance and alter the index of each pixel block  3. Generate the final result |

### 3.2.2 Feature Extraction Module

The feature extraction is another main module in the project that utilize the VGG-19 net to extract feature vector from original superpixel images. The use case diagram can be realized as figure 3.2 below:

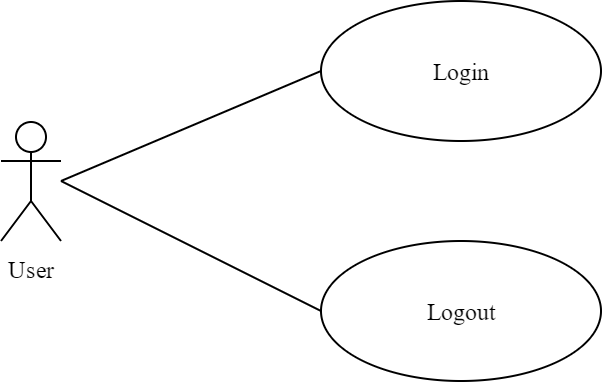


Figure 3.2 Use case of Feature Extraction module

Table 3.10 shows the detailed description of Upload Data. Upload data is available to use internal functionality of web browser.

Table 3.10 Use case description of Upload Data

| **Use Case No.** | 10 |
| --- | --- |
| **Brief Description** | The user upload segmented superpixel data |
| **Actors** | User |
| **Pre-Condition** | The segmented superpixel data must be generated |
| **Post-Condition** | User upload data to system |
| **Flow of Event** | 1. User select path on browser  2. User click upload button |

Table 3.11 shows the detailed description of Extract Feature. As mentioned above, a VGG-19 net is applied to generate feature results.

Table 3.11 Use case description of Extract Feature

| **Use Case No.** | 11 |
| --- | --- |
| **Brief Description** | Get Feature from VGG-19 net |
| **Actors** | User |
| **Pre-Condition** | The data must be uploaded on to server side |
| **Post-Condition** | User get the feature of images |
| **Flow of Event** | 1. Load the VGG-19 net to graphic card memory  2. Calculate the feature vector by convolution and max-pool |

### 3.2.3 Classification Module

Compared with the modules above, there are still some non-trivial modules that play a vital rule on the whole data flow of medical image extraction., such as user module, maintenance module and classification module.

A summary-like use case diagram shown below as figure 3.3 describes these best.

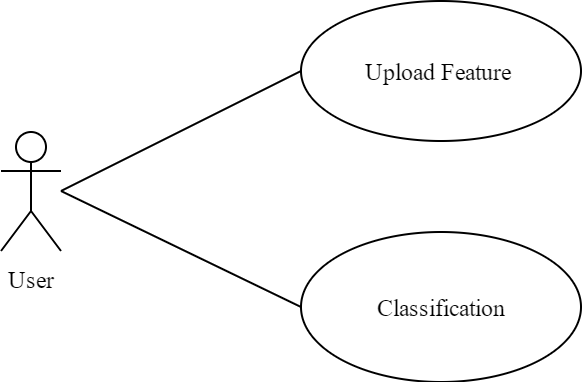


Figure 3.3 Use case of Classification

Table 3.13 shows the detailed description of Upload Feature.

Table 3.12 Use case description of Upload Feature

| **Use Case No.** | 12 |
| --- | --- |
| **Brief Description** | Upload generated feature to system preparing for classification |
| **Actors** | User |
| **Pre-Condition** | The data must be uploaded on to server side |
| **Post-Condition** | User post the data in the server |
| **Flow of Event** | 1. Upload the train data set 2. Upload the test data set |

The classification module aims to use labeled data and the extracted feature inspect the effectiveness of the feature extraction module. Table 3.13 shows the detailed description of Classification.

Table 3.13 Use case description of Classification

| **Use Case No.** | 13 |
| --- | --- |
| **Brief Description** | Do supervised classification using Stacked Auto-Encoder (SAE) |
| **Actors** | User |
| **Pre-Condition** | The data must be uploaded on to server side |
| **Post-Condition** | The user gets the accuracy of the train result |
| **Flow of Event** | 1. Train the SAE for classification 2. Extract hidden values from SAE, applying SoftMax classification |

### 3.2.4 User Module

Due to the simplicity of login use case, register case and logout case, the detail use case description here are descripted briefly.

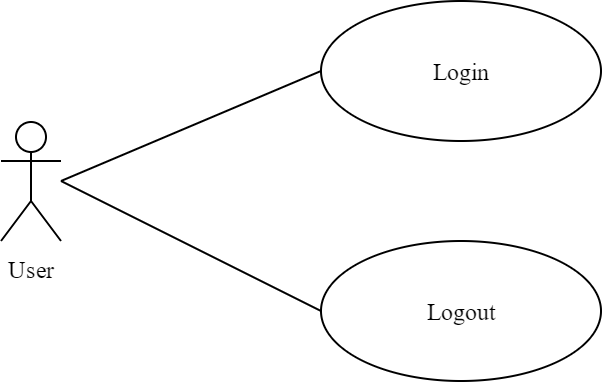


Figure 3.4 Use case of login module

The user login through a login page, Table 3.14 shows the detailed description of Login.

Table 3.14 Use case description of Login

| **Use Case No.** | 14 |
| --- | --- |
| **Brief Description** | User login |
| **Actors** | User |
| **Pre-Condition** | None |
| **Post-Condition** | The system logged the user in |
| **Flow of Event** | 1. User fill the username and password entry 2. User click the login button |

The user logout at the time the logout button is clicked. Table 3.15 shows the detailed description of Logout.

Table 3.15 Use case description of Login

| **Use Case No.** | 15 |
| --- | --- |
| **Brief Description** | User logout |
| **Actors** | User |
| **Pre-Condition** | User logged in the system |
| **Post-Condition** | User logged out |
| **Flow of Event** | 1. User click the logout button 2. The status of this user is labeled logout 3. Browser jump to login webpage |

One last thing has to attention is that the registration module is not designed yet because it is designed for web of local intranet/internet rather than world wide web. The account registration functionality can be achieved directly by administrator.

## 3.3 Feasibility Analysis

Python 3.5 is selected to be the all-stack develop language. Compared with Python 2, well known as Python 2k, the newly Python 3.x has improved serval regions from built-in functionality to external package namespace rule even in interpreter memory allocator. The reason why Python 3.5 is chosen is it can provide the max combability with all deployment environment and is novel enough to acquire new feature driven by internal functionality.

The linear spectral cluster segmentation algorithm is based on numpy, which is the actual standard in artificial intelligence research area when using pythonic implementation. Scientific calculation library such as scipy, matplotlib and TensorFlow are fully with the dependency of numpy. The utilizing of numpy enable the feature of seamless programming architecture, which means it is not necessary of adding proxy layer to integrate components with different algorithm implemented with function/method in a single script.

A deep convolutional neuron network such as VGG-19 with trained parameter data enables the stable feature extracting with fast speed. The interface can be implemented by the requirement context and is highly adjustable to operate, load, calculate and release.

Stacked Autoencoder can provide the stability initial value of classification network, which let the user check the accuracy of the feature extraction model.

## 3.4 Summary

This chapter describes the require analysis of user module, segmentation module, feature extraction module and classification module. The most important modules are segmentation module, feature extraction module. Then a detailed use case of each module is provided as a general introduction of the system. Due to the complexity of the entire system, the main algorithms are modeled encapsulated as modules. The segmentation module includes Read Image Data, Apply Gaussian Filter, Reformat Image, Produce Seeds, Produce Superpixel, Pre-enforce Connectivity, Enforce Connectivity cases. The feature extraction module includes Upload Data and Extract Feature cases. The end of this chapter analyzes the feasibility of the entire system. The next chapter will encapsulate the system design by using different software engineering methodology.

**Chapter 4 System Design**

In the previous chapter, the require analysis methodology generates the system requirements, and there are four main modules, user module, segmentation module, feature extraction module and classification module, compose the system. In this chapter, the project will cover all the architectural design and modules design and database design.

## 4.1 Architecture Design

### 4.1.1 System physical architecture

B/S architecture is applied to the system, which enables the agility to operate the system on any hardware environments. Because the feature extraction task requires high-level graphic cards even distributed graphic computing cluster to process, it is no use to the utilize the client’s processing resources from central process unit to graphic card. Modern styled browser such as chrome is enough to present powerful GUI to the user. Another advantage is that there is no need to install client software, which means the process of user interface is not subject to the network speed and, as a side effect, speeds the client reaction up. The actual physical architecture design available in figure 4.1.

WSGI is short of Web Server Gateway Interface. It defines how a server interact with the web app and how those web applications can be linked together to handle one single request. WSGI is also a standard document described in python language [8]. Thus, it is a low-level library between web applications/libraries to python programs. The specification of WSGI is that it has two ports on the different side, one is server side, and another is gateway side. The server side often describes the python program, and the gateway side usually filled with web server such as apache. The application is often designed as a set of callback function and the server side call the callback function/method python scripts described to handle the request.

One thing matters is that WSGI only defines a variety of rules, but not implement them. It is more like a contract. The WSGI compatible python application only need to obey its definition. Modern python web frameworks almost include its own implementation of it such as Flask. Actually, this project uses Flask rather than a single WSGI implementation.

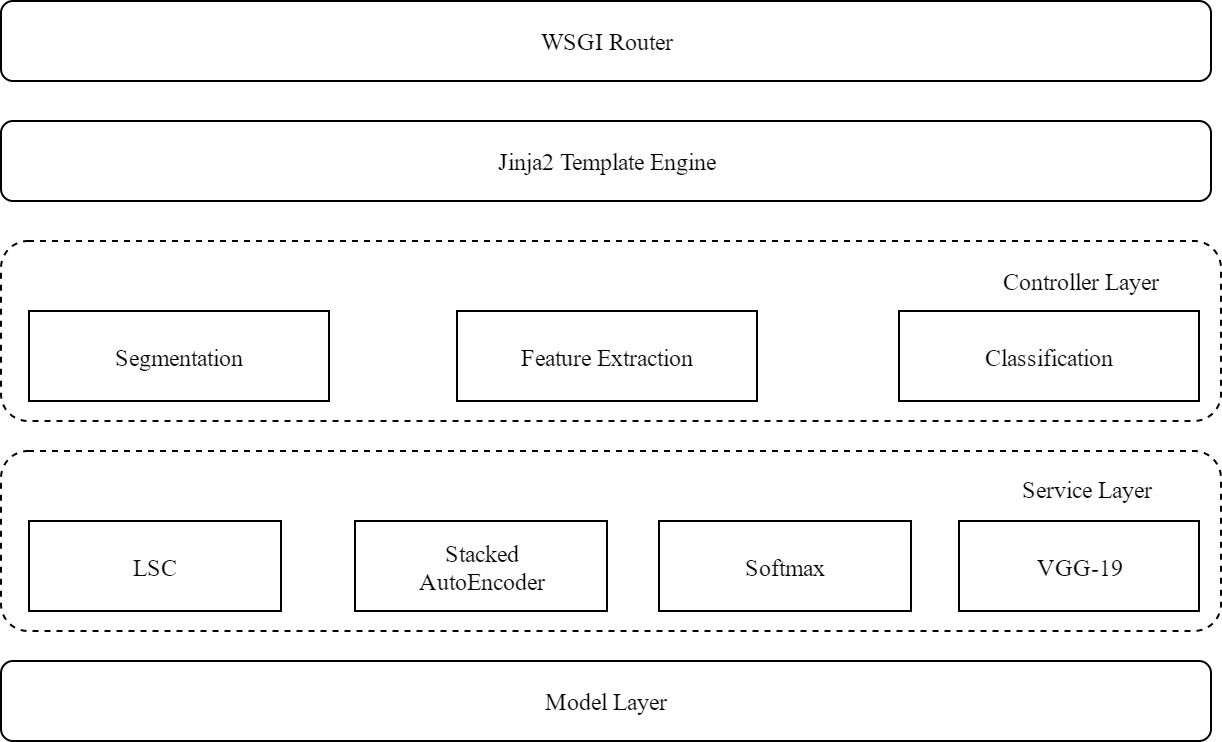


Figure 4.1 System physical architecture

Just behind the WSGI Router, a classical MVC based architecture is implemented in Flask framework. A three-layer architecture of controller layer, service layer and model layer consists the main project physical architecture. The controller layer is response of module-level business. The rule of controller is that each controller usually can only call the service layer component, and the invoke between controllers is rare. In other words, if a block of code is called by controller A and controller B, then this block should be encapsulated in service layer, and the controller A and controller B are here to call them twice in order to achieve code reuse. There are many controllers in controller layer and figure 4.1 only shows some important components in controller layer.

The service layer is a set of reusable functions that requires high computable resource or high memory resource, such as linear spectral clustering or VGG-19 convolutional net. Methods in this layer is low level component contrast to the controller layer. The invoking of one algorithm often requires a massive amount of physical memory. So, the controller component must consider to reuse it in order to reduce the memory usage. The major module in service layer such as LSC, SAE, VGG-19 are covered in the later chapters.

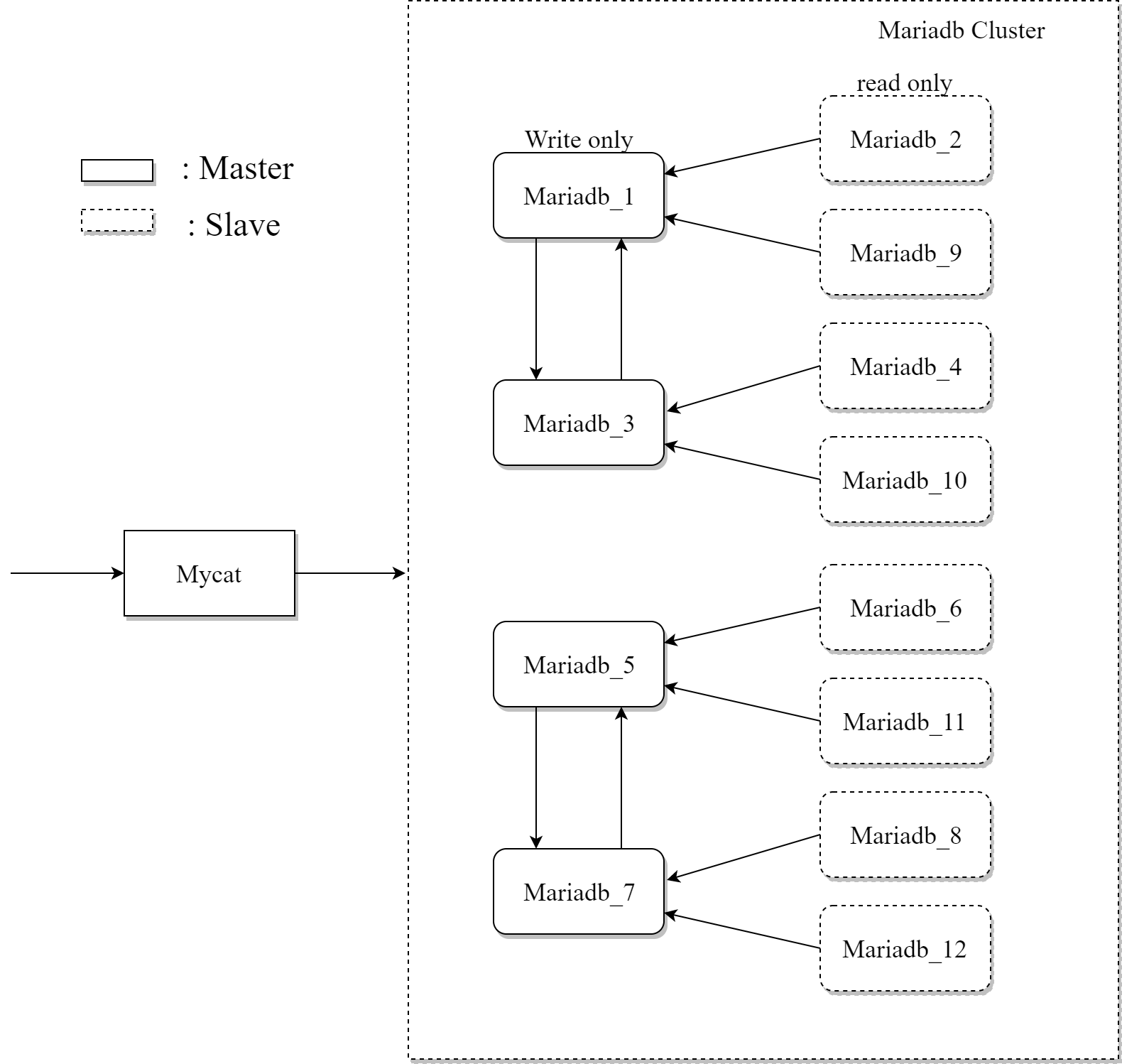


Figure 4.2 Module layer architecture design example

Module layer provides a lazy-loaded, or lazy-executed mechanism that related the hard disk speed limitations. The hard disk used in this project is often lower the speed of the algorithm run in the service layer. For example, the speed of generating feature vectors in memory is greater that the hard disk write speed. It is only specified on the random write speed, not the consistent write speed of the hard disk. Because the more written on the disk, the more likely the lower layer operating system managed to put the file in an inconsistent manner. And due to the physical motor derived hard disk’s long time consume when finding a random block on the disk die, the speed of random read/write is badly limited.

The lowest level of this project is model layer, it provides the database write / read functionality, encapsulates the database operation like migration, CURD operations and other file-specific resource use.

At the system design level, it occurs that the hard disks’ speed cannot catch the speed the CPU generate feature from patients’ PET image. Thus, it is considered to apply database cluster to provide a more robust database module layer. As shown in figure 4.2, the distributed database cluster is designed to solve this problem.

Designs involved in this architecture is the replacing MySQL database with MariaDB. MariaDB is technically supported by MariaDB Foundation. The compatibility is greatened, because of some reasons. First, all the table definition and table definition files all the same format., Not mention that they have identical client API and protocols used for connecting from client. And the most import thing is the client library is almost the same with MySQL’s library. Finally, MySQL is not completely open-sourced, making its open-source derived database, MariaDB, occupies a lot of market in this region.

As shown in the previous figure 4.2, totally 12 MariaDB database instances are involved in this database cluster. It has three advantages described below:

1. **Master-Slave backup mechanism.**

MariaDB itself supports master-slave mode, so through the configuration it can achieve automatic synchronization, which does not need engineers to program in the app level. Thus, the correctness and robustness is guaranteed by commercial company. In detail, the mariadb\_1, mariadb\_3, mariadb\_5, mariadb\_7 in figure 4.2 is master, and others are slave. The mariadb\_1 and mariadb\_3 are each of their master, other masters too. Once one master out of service, the other master will be the alternative choice. The slaves are with the similar mechanism to handle the unexpected crush.

1. **Read/Write Splitting**

Single process model is the simplest design for storing small pieces of data, and there is no problem of reading and writing confliction. But the single process in the processing capacity becomes a bottleneck. In order to improve the processing power, it is necessary to use multi-process, but it introduces the problem of reading and writing conflict. In figure 4.2, the masters mentioned above are write host, and the slaves are read host.

1. **Sharding (Fragmentation)**

Division number is a classic sharding methodology, which is suitable for the case of integer keys, each device to store the same size of the section. For example, let the id divide by 2, then the data whose remainder equals 0 goes to sharding cluster 0, other data goes to sharding cluster 1. In figure 4.2, it is clear that the mariadb\_1 to maradb\_4 and mariadb\_9 and mariadb\_10 is the cluster 0, others are cluster 1 mentioned just above. For convenience, Mycat is introduced as a database proxy, which is a database middleware. It has the ability of auto sharding without program it in application layer.

Mycat is a completely open-sourced, ready for enterprise application development of large database cluster support transaction. With the support of Mycat, the MariaDB can be considered a high-efficiency single database. This project applied the Mycat as a proxy to the database cluster.

### 4.1.2 System logic architecture

Different from system physical architecture, the system logic is quite simpler. Detail architecture shown at figure 4.3 below.

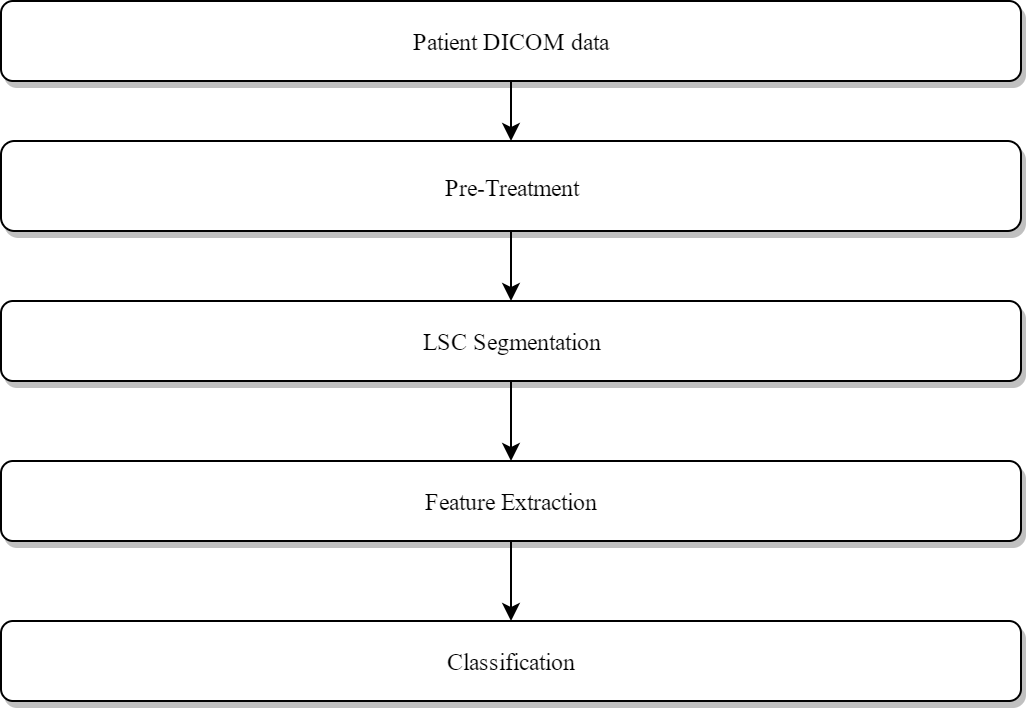


Figure 4.3 System logic architecture

The start point of the logic flow graph is that the patient’s DICOM data. DICOM is short of Digital Imaging and Communication in Medicine. It contains a key-value paired data include all the patients’ personal information and medical info help the diagnosis. Then my groupmate does the pre-treatment module to processes the DICOM data, which outputs the image matrix used for LSC segmentation module. Then the segmentation result is used to do feature extraction. Finally, for checking the effectiveness of feature extraction module, the classification module is introduced.

## 4.2 Key Module Design

According to the analysis of previous chapters, four modules are designed to compose the project as shown in figure 4.4.

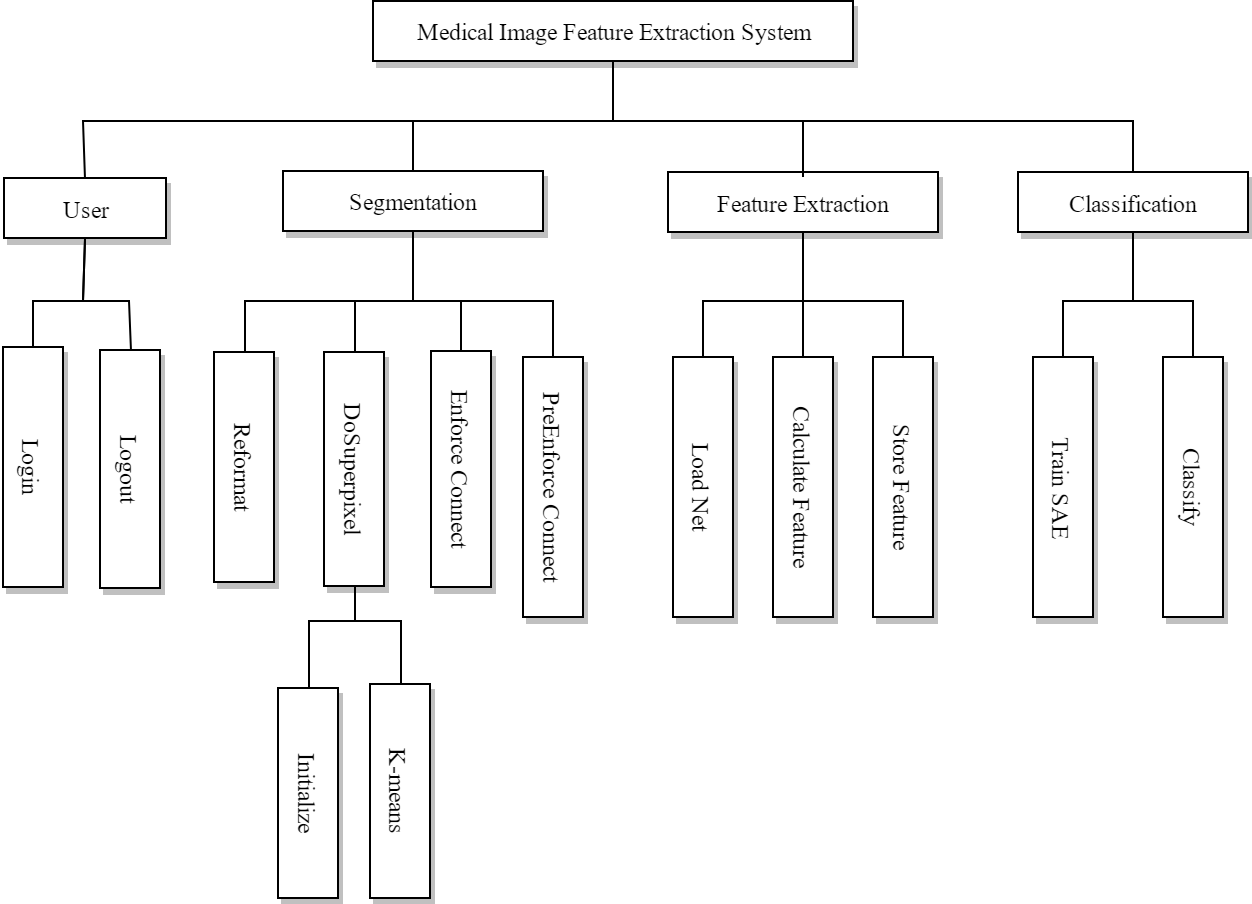


Figure 4.4 system function module diagram

### 4.2.1 Feature Extraction Module Design

**(1) Layers in VGG-19**

The Feature Extraction Module is mainly used VGG-19 convolutional neural network to extract feature. The weights are copied from groupmates. With the development of science and technology, especially in the region of parallel computing on graphics cards, even an ordinary game laptop with Nvidia GTX960M graphic card can almost put all the trained parameter in its gddr5 memory. Shown in figure 4.4, this project uses the first 17 layer of VGG-19 to extract meaningful feature.

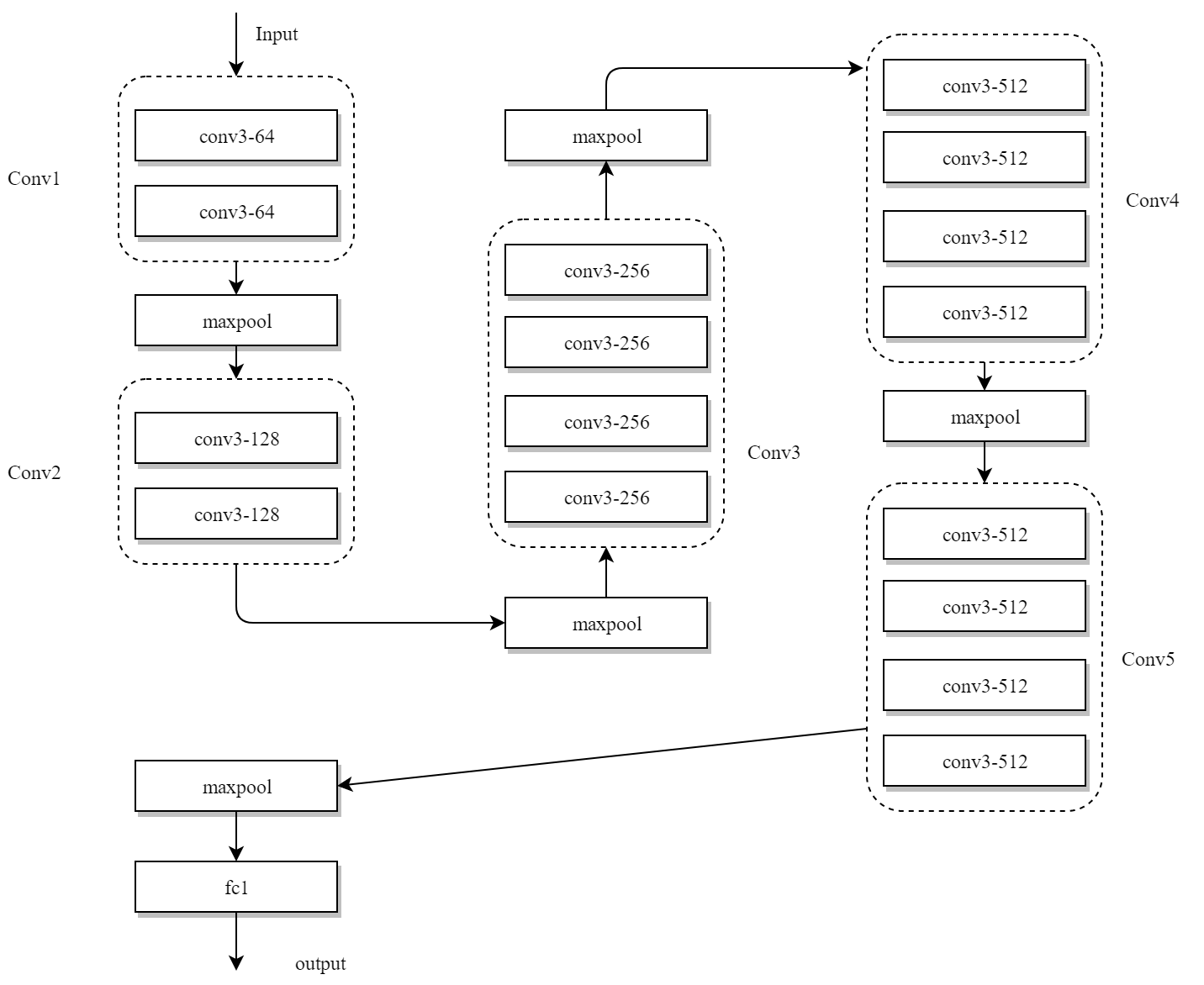


Figure 4.5 Convolution layers and maxpool layers in Feature extract module

As shown in figure 4.5, The conv3 means that the filter of this convolutional layer is 3x3 size, and the 64 in the image above is the output kernel number of this layer. As the data flows the output filter number of layer increases from 64 to 512. Then the paragraph below introduces the convolution. The strides, the step length from previous position to next position, at each dimension, are set to 1. That means, the slide window when calculate the convolution once covers one new image, one new line, one new column, and one new channel. A matrix of shapeimage, line, column, channel is the standard 2-D image convolutional shape presentation when used in convolution, it must be a 4-dimension matrix.

**(2) Convolution Mechanism**

Convolution has its meaning in mathematics, but the convolution used in convolutional neural network is a main process of convolutional layers. Based on some research in the biological, the convolutional neural networks take use of the matter that some weights are shared in the network, and the way to realize this is convolution [9]. The convolutional layers apply a specified filter to the image data. It just does a sequence of mathematical operation to generate a single number in the convolved feature map.

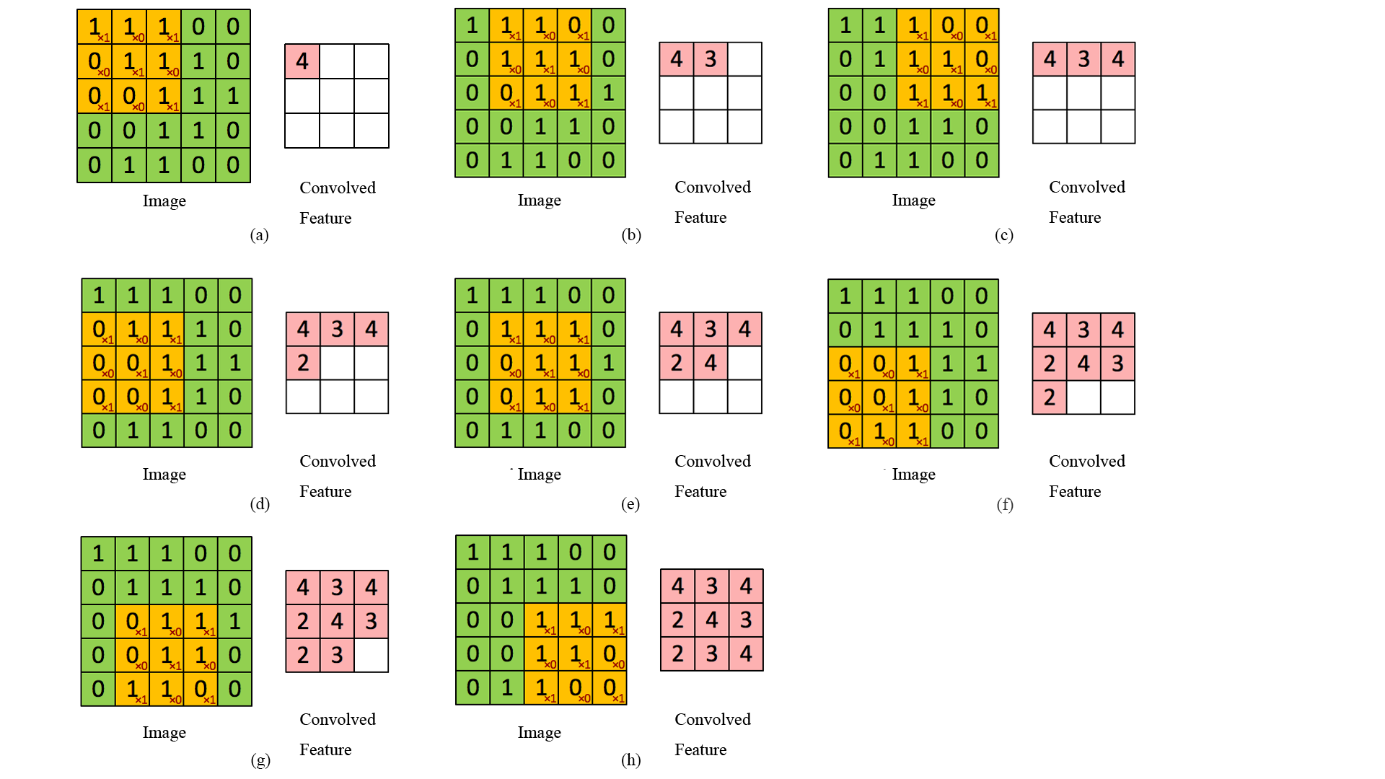


Figure 4.6 A convolution example

In figure 4.6, the Image is of shape five by five, and the value are 0 or 1. The filter is of shape three by three. The convolution in this area is slide the filter from the left to right, and then up and down. Each different position filter in the image produces a single output value in the output, that means, the convolved feature. In figure 4.5(a), the first position the filter finds is left most and the up most position, and then convolve the image data in this filter are to the filter data itself, then produce a single value of 4. As the filter slides to right side, it comes to (b) and (c) in the figure 4.5. Then repeat the action done above, but at a different horizontal level. Until all the horizontal levels are convolved, the output composed a full convolved feature is also a matrix. There is still possibility that the output is only a single number, that means the image size is equal to the filter, then the selected are between image data and filter is only one, then the output is only one number, in order words, a matrix of shape one by one.

**(3) Maxpool Mechanism**

Similar to convolution, the maxpool is still a sample-based mechanism. The aim of the maxpool is to get a smaller representation of input, and reduce its complexity [10]. There is still has the conception of position. As shown in figure 4.7, the filter is a matrix of shape two by two, and the strides of the maxpool, however, different from the convolution operation. The strides of the maxpool often equals the shape of the filter matrix rather than 1.

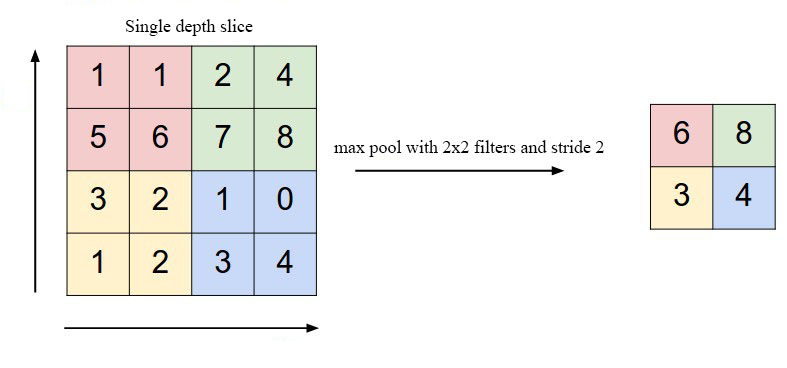


Figure 4.7 Maxpool example

Filled with convolution and maxpool mechanism, the only uncovered layer in figure 4.5 is the fc1 layer. It is a full-connect layer, the operation hidden in this layer is an ordinary matrix multiply. According to the rule of matrix multiply, the column number of the full-connect matrix is the output size of the whole module is 4096.

**(4) Sequence Diagram of Feature Extraction Module**

For further understanding of the feature extraction module the sequence diagram is provided below as figure 4.8:

The browser is the main actor of the upload operation. The user selects the actual path on the client machine. Then the internal mechanism ensures the file can be uploaded with multipart/formdata rather than application/x-www-form-urlencoded. Then the Extract Data Controller empties the previous user uploaded data, locate the actual path the weight matrix of VGG-19, and the ExtractDataService, described in the system architecture design, call the VGG-19Service to load the whole convolutional neural net. The VGG-19 net is resource consumable, that’s why load this net every time the request comes. In the actual situation, there is not enough memory to calculate the all image data in one loop, so the lazy load mechanism is applied in the ExtractDataService. The persistent-related save operation is also lazy save. When all the image done its feature extraction, a signal is return to the controller, and the controller returns a JSON message to the browser.

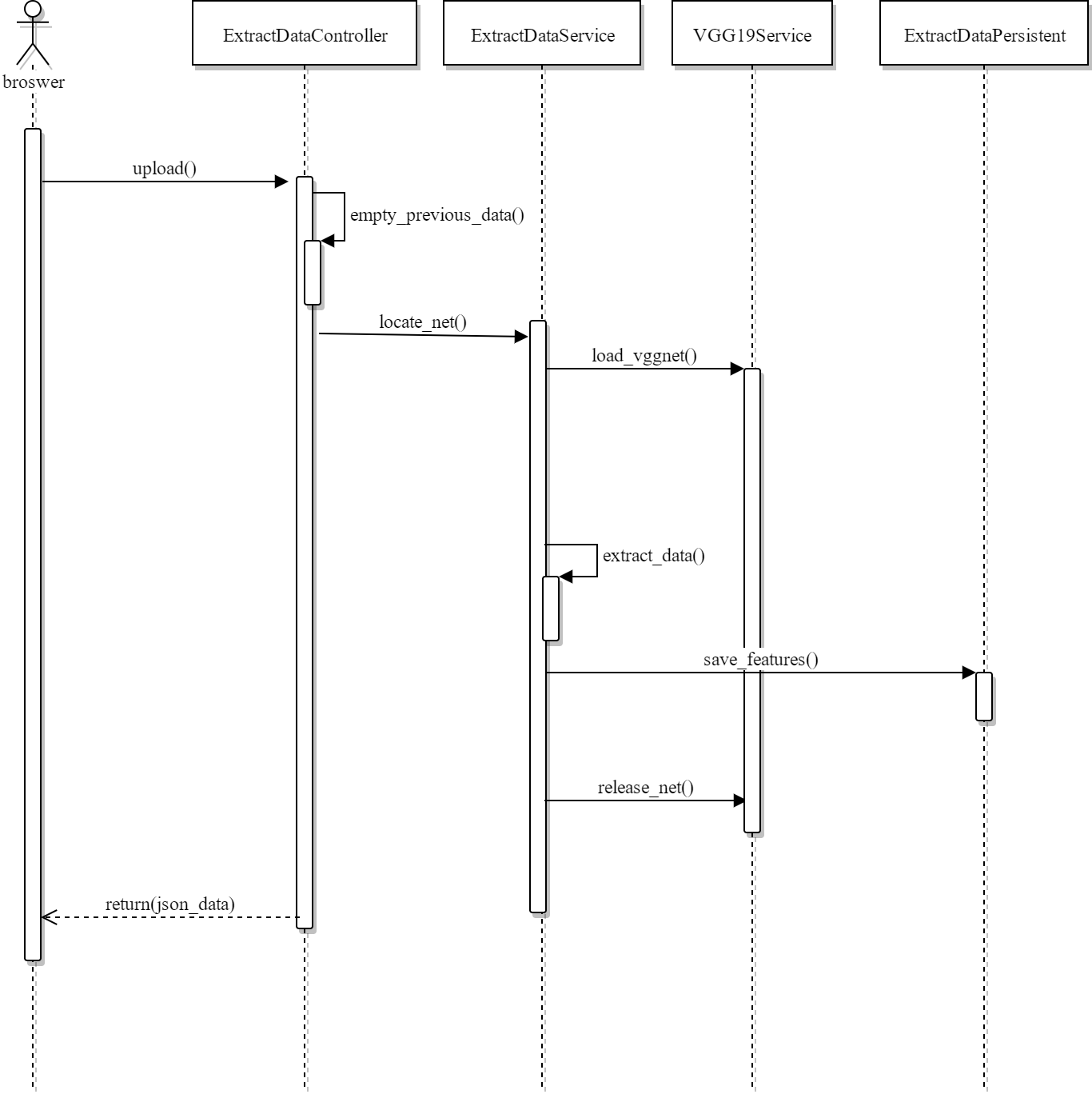


Figure 4.8 Sequence diagram of feature extraction module

### 4.2.2 Classification Module Design

Stacked AutoEncoder (SAE) is the main algorithm composed the classification module. First, a SAE contains serval autoencoders. A autoencoder, shown in figure 4.8, is a simple neural network that tries to learn a function . In a nutshell, the aim of a autoencoder is to trying to get a function that is an approximation to equal function [11]. But if some constrains are applied to the middle layer of the network, then the autoencoder may find some useful information from the input data.

In figure 4.8, the x1, x2…x6 are input data, and the +1 in layer L1 is biases, which is identical to put +1 in the input data rather than calculate the bias. The left-half image has may blank lines link the xs to the middle nodes in layer L2, which represents the weights of matrix W1. Imagine the input is a matrix, then the layer L2 equals to the input matrix multiply the weight matrix, finally, add the bias vector(matrix) to the previous result. It can be described by formula . But as mentioned above, the bias can be replaced by add the +1 in the input data by adding an extra row. Thus, the formula of calculating the L2 is change to . The same as the calculating values of layer L3 from the layer L2 data.

Since the model has been constructed, the weights need to be trained to generate the meaningful and usable hidden layer L2 values is on the schedule. The main idea of training the autoencoders weights is define the loss function J, which is a sum of the idea of keeping the autoencoder sparsity enough, then calculate the gradients of the loss function J. After get the partial gradients of each weight in weights matrix W1 and W2, the weight value in the matrix can be changed to meet the requirements of the loss function.

As important to the loss function, the number of hidden units S2 has to be less that the number of input data in layer L1. That is because the aim of the network is concerned on discovering concise representation of the input data. The exact implementation of the loss function and how to use the loss function to get partial gradient of each weight is covered at next chapter.

In the previous paragraph, it is introduced the single autoencoder. The output of one autoencoder is the hidden layer values in the layer L2 in figure 4.9. In most cases, one autoencoder is not good enough to extract the feature we want, so a multi-layer autoencoder is under consideration. If the output is treated as one autoencoder as the input data of another autoencoder [12], then in the logic meaning, the previous autoencoder on to another autoencoder like things do in figure 4.10.

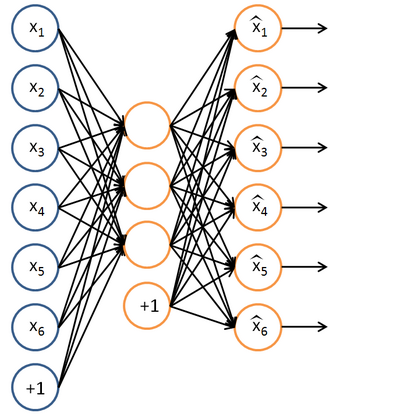


Figure 4.9 Autoencoder example

In figure 4.10, the hidden layer in the previous autoencoder are directly inputted into the second autoencoder to generate the output of the hidden layer.

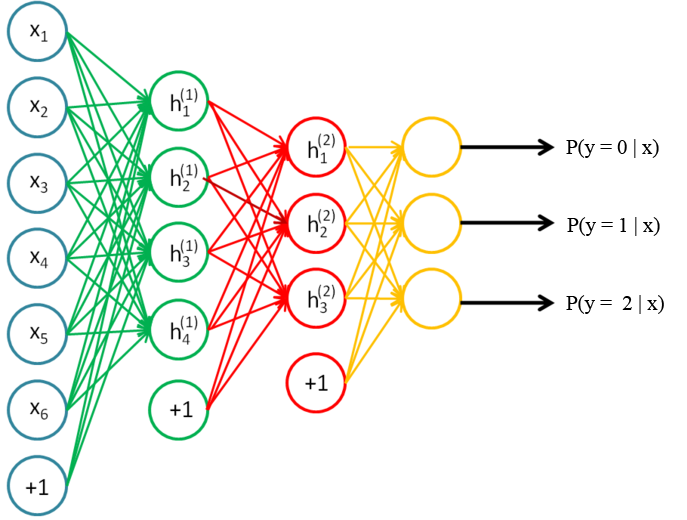


Figure 4.10 Stacked AutoEncoder example

Then a softmax layer is plugged into the last component of the autoencoders. The reason of introduce a softmax layer is to achieve supervised learning. The autoencoders are a kind of unsupervised learning. Without the softmax layer, this network cannot do the classification functionality. The layers of the stacked autoencoder is unlimited, but there is a limit of trainable network. Deep neural network may be easy to represent a number of complex functions, but is hard to train. For example, the gradient descent phenomenon makes the neuron trained to zero values eventually. So, if the network is able to converge, the net must be not too deep. Then the depth depends on actual experiment.

After introducing the stacked autoencoders, there is still a serious problem to deal with. Similar to problems faced in feature extraction module, there is no choice of putting all train data in one loop in order to get partial gradient of all the weight matrix. Contrasting to the lazy load in feature extraction module, the batch generation mechanism is widely used in the stack autoencoder training. The main idea of batch is generating randomly a part of training set rather use all of the training data to reduce the resource required.

The sequence diagram of classification module shown in diagram 4.11.

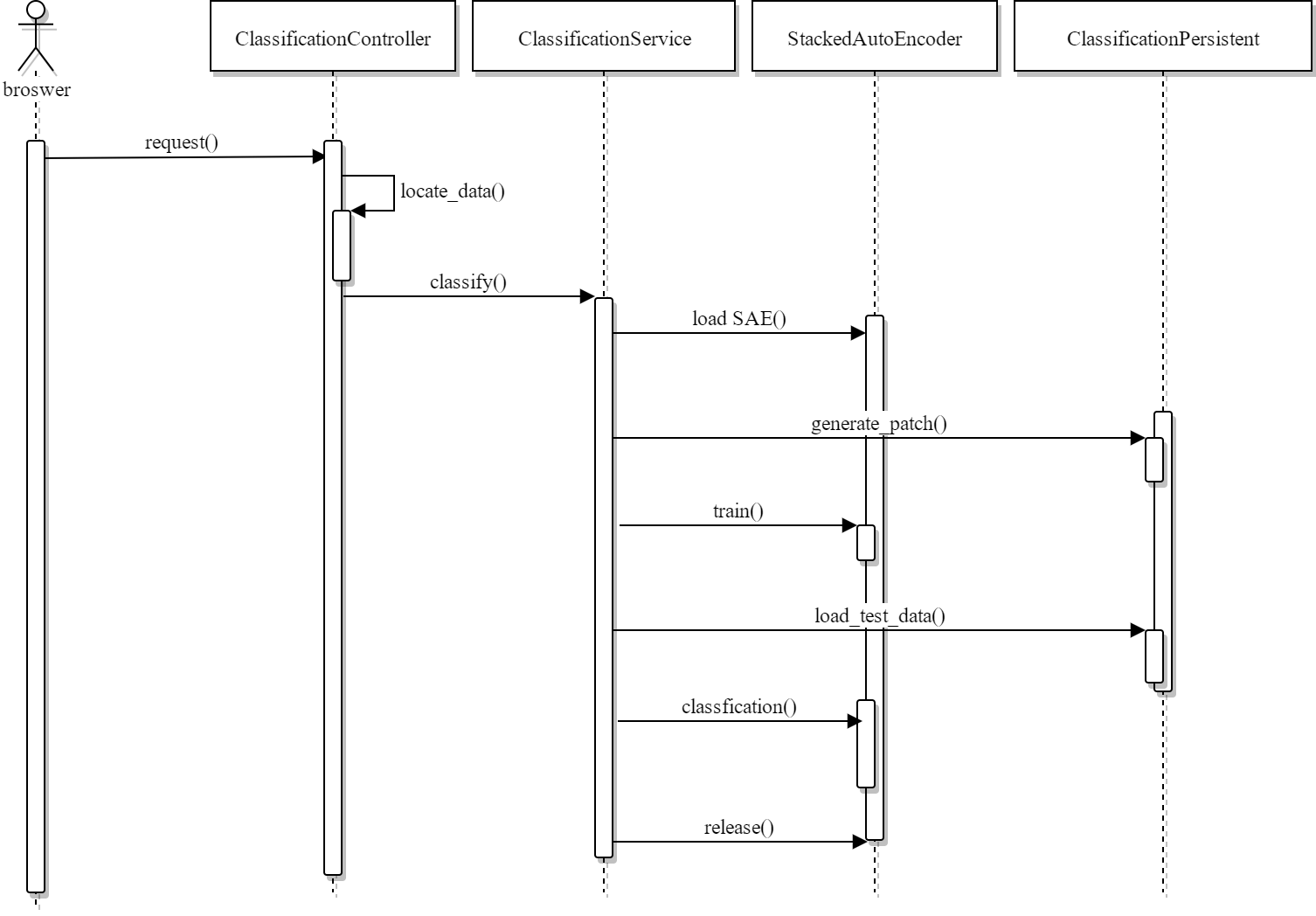


Figure 4.11 Sequence diagram of classification module

Since this module is to verify the effectiveness of the feature extraction module, the first thing that a controller do is to check if the train data and test data is loaded. If it meets the prerequisite this module asked for, the classification service will be called to load stacked autoencoder. Then the patch is generated to train the stacked autoencoder. After serval steps, the test data is loaded to evaluate the accuracy. Then the accuracy is encapsulated as json data returned to the browser.

The classification service call of the release net is designed because the stacked autoencoder is memory consumed. For further memory efficiency, the memory of each autoencoder is designed to release once the current layer is trained. By using this mechanism, the total memory usage is not to increase by the number of layers of the stacked autoencoder though the memory usage can be increased by the additional hidden features.

### 4.2.3 Segmentation Module Design

As the third important module in this project, the segmentation module is introduced in the third part of this chapter. The main algorithm of segmentation module is linear spectral cluster.

The linear spectral cluster algorithm has the time complexity O(n), which is suitable for clustering huge image data. Some kind of cluster algorithm based on some artificial intelligence or complex recursive mechanisms that have the time complexity greater than O(n), making the time consuming unacceptable. But the actual result is ideal. Another kind of segmentation algorithm is fast and require much less resources such as physical memory, but the final result of the cluster is not full of edge-detective. But from archive [7] it realized with some pre-conditions, the loss function of one high-resource algorithm is mathematically equals to one fast algorithm. That means can imitate the behavior of it to catch the performance of the fast algorithm.

In this algorithm, k-means clustering [13] is the main method to realize. But before using the image data to do the clustering, the algorithm use the mapped high-dimensional data instead the value of the RGB value of the image with weighted recursive to optimize the final result.

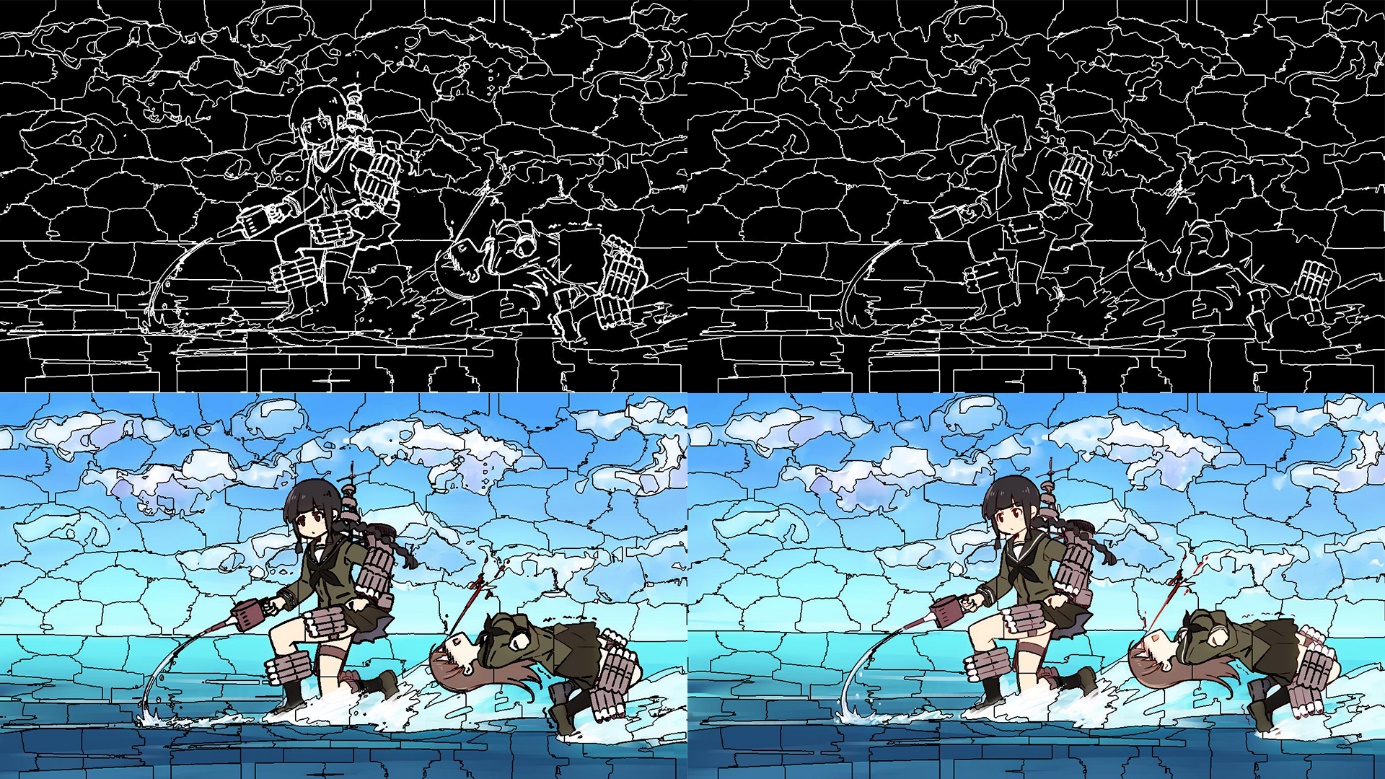


Figure 4.12 Example effect contrast of merging tiny superpixels. The left side is the result without merging, right side is with the procedure.

After the main procedure it produced, the linear spectral cluster algorithm also introduces an enforce connectivity mechanism to refine the result. The main idea of it is search every superpixel generated by the previous k-means step. Since the superpixel number required is fixed before the procedure starts, the average mean size of the superpixel is fixed either because the image size is numeric. It loops through all the superpixel, if a superpixel size is less than one quarter of the mean size, the superpixel is to be merged to the neighbor of the nearest superpixel. Actual effect of merging shown in figure 4.12. The left side of the figure is the handling without the enforcement, and right side of the figure is the filtered result. It is easy to find the left side is a kind of over segmentation, and there are many small superpixels that should not be included in the final result.

The sequence diagram of segmentation module is modelized below as figure 4.13:

The browser uploads the data to server, and the segmentation call the segmentation service for reformatting it to LAB value. Then the k-means cluster algorithm is applied to generate primary label values. After the clustering, the pre-enforce connectivity merges the small blocks. Finally, the labels are consistent with the enforce connectivity.

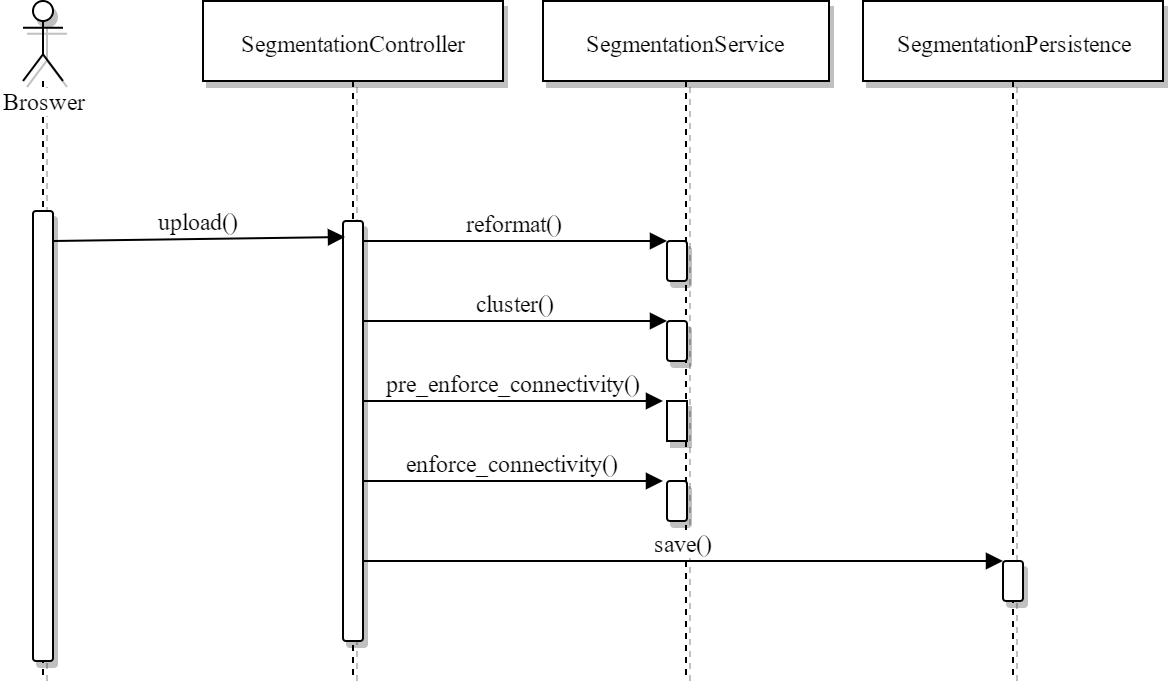


Figure 4.13 sequence diagram of segmentation module

### 4.2.4 Login Module Design

The login module checks the password the user provides. Detail diagram shown as figure 4.14:

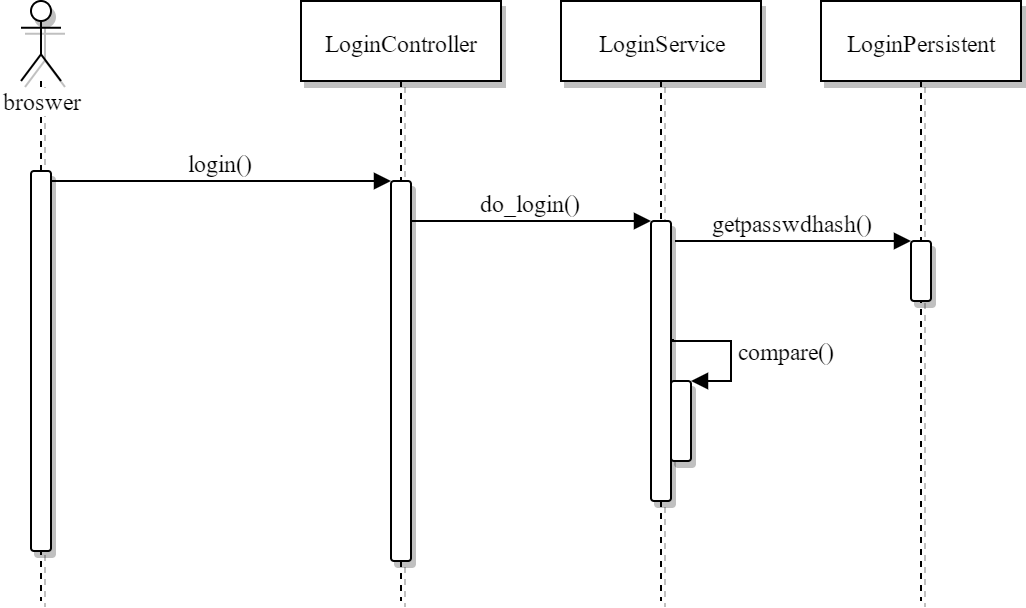


Figure 4.14 Sequence diagram of login module

The only thing login module does is to check the if the user’s password equals to the right hash. If the user provides the correct password, the system logs the user in. If not, a json data contains the failed information is returned to browser.

## 4.3 Database Design

MariaDB is used in this system. Mysqldb, which is the traditional database connection library, is not used because it not supports python 3.x officially. Instead, a pure-python connector called pymysql is used as the database connector. The E-R diagram of the system is shown as figure 4.15.

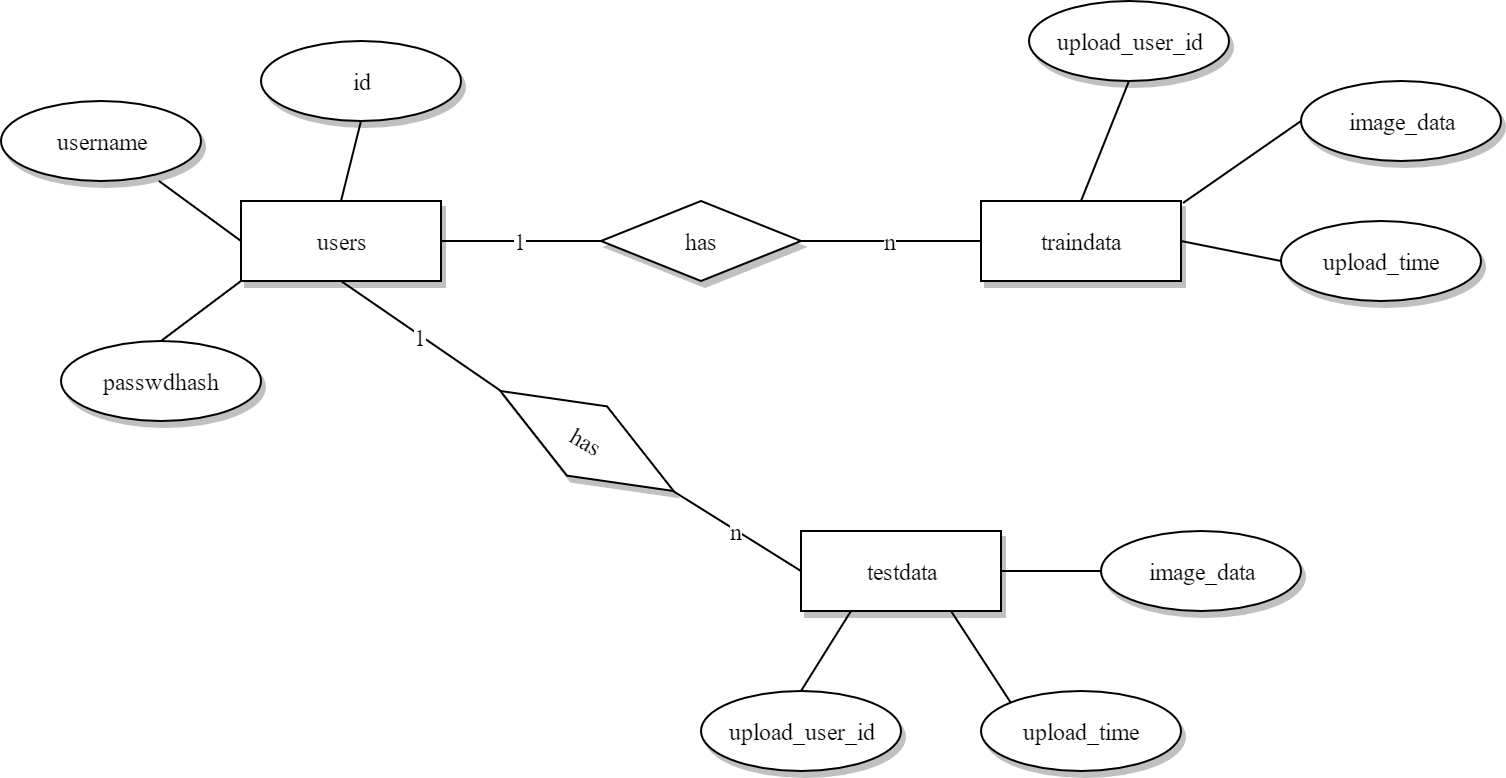


Figure 4.15 E-R diagram

The detail structure of table users is as table 4.1 shows:

Table 4.1 Detail structure of table users

| **Column Name** | **Data Type** | **Primary Key** | **Default** |
| --- | --- | --- | --- |
| id | INT | Yes |  |
| username | VARCHAR(30) | No |  |
| passwdhash | VARCHAR(129) | No |  |

The detail structure of table traindata is as table 4.2 shows:

Table 4.2 Detail structure of table traindata

| **Column Name** | **Data Type** | **Primary Key** | **Default** |
| --- | --- | --- | --- |
| upload\_user\_id | INT | Yes |  |
| image\_data | VARCHAR(65535) | No |  |
| upload\_time | DATATIME | No |  |

The traindata table is used for storing the train image data. Each row in the traindata contains a data of a base64 encoded binary image. The testdata is the same structure as the traindata table.

The detail structure of table test data is as table 4.3 shows:

Table 4.3 Detail structure of table testdata

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Primary Key** | **Default** |
| upload\_user\_id | INT | Yes |  |
| image\_data | VARCHAR(65535) | No |  |
| upload\_time | DATATIME | No |  |

## 4.4 Summary

In this chapter, the system physical architecture is described with the technology required to compose the web services. Logical architecture describes how the patients’ image data is processed. Then, the main four modules are described with their sequence diagrams. After that, the database design is shown as tables, which are the main idea of the model layer in the system physical structure. The next chapter introduces the system implementation.

**Chapter 5 System Implementation**

In the previous chapter, the system design of the project is introduced, and the we will introduce the actual implementation of each module described in the chapter before.

## 5.1 Development Environment

A python web library called flask is used as the main web framework. Flask is a light web library with the werkzeug and Jinja2 template engine [14]. The typical initial usage of flask is only serval lines of codes in a single script, which makes it easy when developing.

TensorFlow is integrated into the Flask application, which is introduced in chapter two, enabling the feature of receiving train data from web. PyCharm is described in the chapter two, which as an integrated python development environment, managing the integration of the frameworks.

Git is a free version control system, which is open-source, and has the feature of branch, distributed developing. Distributed is the new centralized, as the git philosophy, the distributed agility is the main advantage in group developing.

Python 3.5 is used as the main interpreter because of the TensorFlow compatibility on Windows.

## 5.2 Implementation of Feature Extraction Module

The loading of VGG-19 net is a part of the feature extraction module. The flow diagram of the implementation is shown as figure 5.1.

First step of this program is to load the weights from file. The scipy.io.loadmat library is used to load the weights because the file in this project is matlab compatible format. The default loadmat function use example is matlab style, only string type of filename is required:

data = loadmat(data\_path)

The data\_path is an argument from the outer variable domain. Then the layers are should be declared in tensorflow interface to compose the whole net. But the whole layers are defined as a python tuple of string:

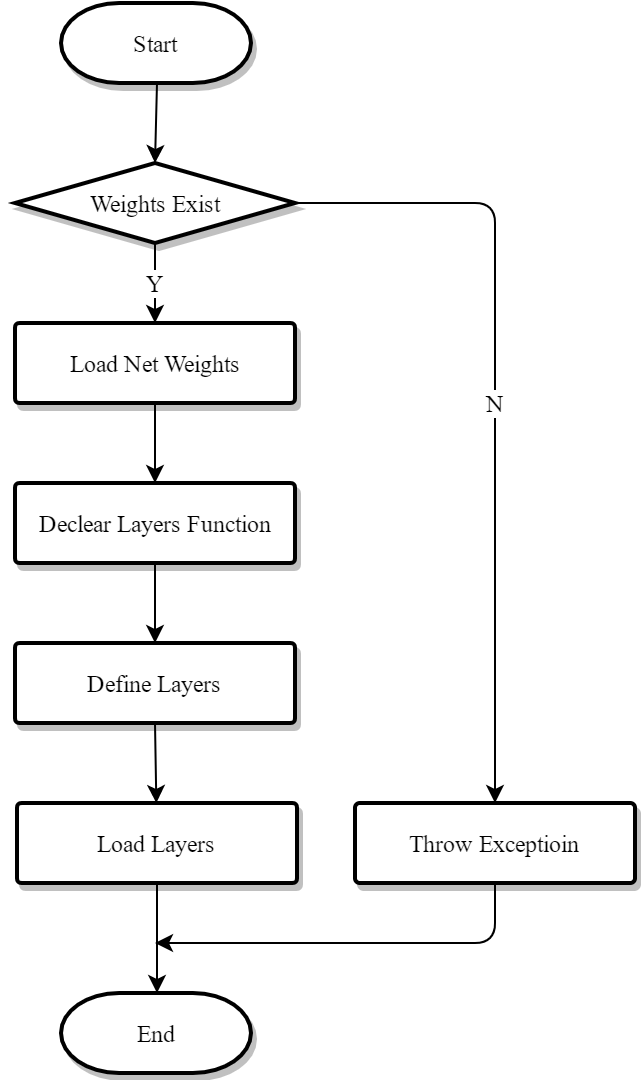


Figure 5.1 flow diagram of VGG-19 net

layers = (

'conv1\_1', 'relu1\_1',

'conv1\_2', 'relu1\_2',

'pool1',

'conv2\_1', 'relu2\_1',

'conv2\_2', 'relu2\_2',

'pool2',

'conv3\_1', 'relu3\_1', 'conv3\_2', 'relu3\_2', 'conv3\_3', 'relu3\_3', 'conv3\_4', 'relu3\_4',

'pool3',

'conv4\_1', 'relu4\_1', 'conv4\_2', 'relu4\_2', 'conv4\_3', 'relu4\_3', 'conv4\_4', 'relu4\_4',

'pool4',

'conv5\_1', 'relu5\_1', 'conv5\_2', 'relu5\_2', 'conv5\_3', 'relu5\_3', 'conv5\_4', 'relu5\_4',

'pool5',

'fc1',

'relu\_fc1'

)

As mentioned above, the convolutional layers are named by conv such as conv1, means the first layer’s convolutional kernel. Pool means the maxpool layer, Fc means the full connect layer. Relu means the ReLu function. Due to the deep recursive definition, the extraction using slice/index operation in python can be litter weird:

for i, name in enumerate(layers):

kind = name[:4]

if kind == 'conv':

kernels, bias = weights[i][0][0][0][0]

kernels = np.transpose(kernels, [1, 0, 2, 3])

bias = bias.reshape(-1)

current = \_conv\_layer(current, kernels, bias, name=name)

elif kind == 'relu':

current = tf.nn.relu(current, name=name)

elif kind == 'pool':

current = \_pool\_layer(current, name=name)

The enumerate is a built-in function in python. Function enumeratereturns an enumerate object. The iterable should support iteration in python. The return value yields a tuple contains the index and a value from the iterable parameter. The kind is used to extract and store the type of each layer. Slice operation of parameter 4 defines to get the contents from start to the 3rd object, the 4th object is not included in the iterable slice result. For example, the forth slice of conv1\_1 gets the conv, which is the actual type of the layers’ definition. If the kind string is equals to the conv, the kernels, bias variable can be extract from the loaded weights by the slice operation, whose parameter is the index generated by the function enumerate mentioned above. The reason why there are 4 zero slice is that the deep recursive definition of the weights. The next step is to transpose the kernel matrix because the matlab version of weights is column-row style or Perl style, but the python, like arrays in C, is row-column style or C style, that means the numpy.transpose function should be used to transpose the 4-dimensional matrix. Then the bias variable should be transposed to fit the tensorflow operation to shape of none, shapeof none means the any shape of ndims equals to 1 is fit for this shape pattern such as shape of slice thirty-two. And the ndims is the number of dimensions of the matrix in numpy representation. Variable current above is used every loop because what variable current stores is not only the output of the previous layer, but the input of the next layer. The other branch of the if-else clause is much simpler, such as clause whose condition is the kind equals to relu or equals to pool. Finally, when the if-else clause is completely not fitted to the kind string, the default else clause is used.

The declaration of the convolutional layer and pool layer is encapsulated as function below:

def \_conv\_layer(input, weights, bias, name):

conv = tf.nn.conv2d(input, tf.constant(weights), strides=(1, 1, 1, 1), padding='SAME')

return tf.nn.bias\_add(conv, bias, name=name)

def \_pool\_layer(input, name):

return tf.nn.max\_pool(input, ksize=[1, 2, 2, 1], strides=(1, 2, 2, 1), padding='SAME', name=name)

The tf.nn.conv2d is a function provided by the tensorflow which let the project programmer can focus on the actual layer design of the network rather than the implementation of the convolutional procedure. The conv2d computes a 2-D convolution, but the input of variable input and filter arguments are 4-D required. The detail implementation of the conv2d is transparent to python interpreter because it is a C-extension library which implemented by C++ language in order to improve the speed. But from the function comment can know is it calculates the convolution of the input and filter by flattens the filter argument to a two-dimension matrix as the shape of shape filter\_row, filter\_column, channels\_in and channels\_out, and the image as shape of batch\_num, row, column filter\_row, filter\_column, and channels\_in. According to the matrix multiply rule, if the image is left-multiplied with the filter, it will produce a matrix of shape of batch\_num, row, column and channels\_out, actually is the definition of the convolution mention in the previous chapter. As the comment say the default NHWC format of the calculation shows below:

output[b, i, j, k] = sum\_{di, dj, q} input[b, strides[1] \* i + di, strides[2] \* j + dj, q] \* filter[di, dj, q, k]

The precondition of the calculation is if the 3rd slice of strides is equals to the zero slice of the variable strides. In usual, the strides of shape one, stride, stride and one is similar to the conv2d function in tensorflow, the max\_poolfunction also requires 4-dimensional image matrix as the first argument, but due to the feature of the maxpool, the second argument is the kernel size rather than the kernel itself.

Finally, the call of the VGG-19 net constructs the main procedure of the feature extraction module:

with tf.Session() as sess:

for i in range(0, images.shape[0], batch\_size):

batch\_xs = images[i: i + batch\_size]

batch\_xs = np.stack([batch\_xs, batch\_xs, batch\_xs], axis=3) - mean\_pixel

result = sess.run(nets['fc1'], feed\_dict={input\_images: batch\_xs})

result = result.reshape(batch\_xs.shape[0], 4096)

The with-as clause is a special grammar in python programming language, which enables the interpreter-level memory control and safe release mechanism. As introduced above in chapter 2, the session is as special context in tensorflow programming. In the tensorflow session, the value of tensors can be evaluated by physical device such as graphic card. The batch\_xs variable is a batch, and numpy.stack is used because the original image data is single channel, but the VGG-19 net requires RGB channel data. So, the function is mainly join an array into new axis. tf.Session.run function is the evaluate function that evaluate a tensor from a tensorflow graph. In this situation, it evaluates the fc1 layer output, which is a vector of shape of 4096.

## 5.3 Implementation of Classification Module

The classification use SAE to build proper weights that classify the results. The flow diagram of classification module shown below as figure 5.2:

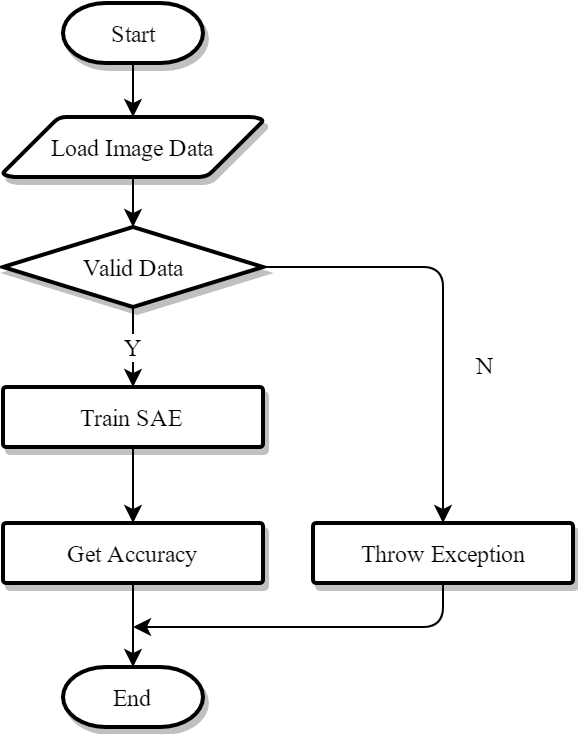


Figure 5.2 Flow Diagram of Classification Module

The loading of train data and test data is divided in two steps, one is load the file uploaded onto the system, another is generating the batches, as the pre-treatment procedure.

images, labels = persistents.classification\_persistent(test\_folder)

test\_images, test\_labels = \_gen\_batches(images, labels, [-1, 0, 0])

sas = SAE.train\_stacked\_autoencoders(in\_data=train\_images, stack\_shape=[4096, 512, 64, 6])

The test\_images, test\_labels are the generated tests batch. The stacked autoencoders are utilized in the program. The implementation of the stacked autoencoders are implemented here：

for i in range(len(stack\_shape) - 1):

train\_data = in\_data if i == 0 else stacked\_autoencoders[i - 1].transform(train\_data)

for epoch in range(training\_epochs):

for k in range(train\_data.shape[0] // batch\_size):

batch\_xs = get\_random\_block\_from\_data(train\_data, batch\_size)

stacked\_autoencoders[i].partial\_fit(batch\_xs)

return stacked\_autoencoders

The stack shape is a list of tuples, which describes the shape of the stacked autoencoder. The number of input layers and the number of hidden layer is the only two parameters required. If only one autoencoders composes the stacked autoencoder, the length of the tuple should be 2. But the length of the stack is two doesn’t mean the length of the tuple is length of 4, because the output/hidden layer number of previous layer is equals to the input of next layer. So, the length of stack shape list is of length 3. Then the train data obtained by the train\_data is changed each loop. The actual hidden layer weights will be loaded into a full connect network below:

x = tf.placeholder(dtype=tf.float32, shape=[None, 4096])

y\_ = tf.placeholder(dtype=tf.float32, shape=[None, 6])

w1 = tf.Variable(sas[0].getWeights(), dtype=tf.float32)

b1 = tf.Variable(sas[0].getBiases(), dtype=tf.float32)

w2 = tf.Variable(sas[1].getWeights(), dtype=tf.float32)

b2 = tf.Variable(sas[1].getBiases(), dtype=tf.float32)

w3 = tf.Variable(sas[2].getWeights(), dtype=tf.float32)

b3 = tf.Variable(sas[2].getBiases(), dtype=tf.float32)

[\_.close\_session() for \_ in sas]

h1 = tf.nn.softplus(tf.matmul(x, w1) + b1)

h2 = tf.nn.softplus(tf.matmul(h1, w2) + b2)

y = tf.nn.softplus(tf.matmul(h2, w3) + b3)

loss = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(labels=y\_, logits=y))

train\_step = tf.train.AdamOptimizer(learning\_rate=1e-4).minimize(loss)

correct\_prediction = tf.equal(tf.arg\_max(y, 1), tf.arg\_max(y\_, 1))

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

The placeholder of x and y\_ is the actual image/label placeholder of the network. Placeholder is a mechanism of lazy evaluation, which enables the calculation of huge data. The shape of placeholder can be pre-defined by the shape argument, which is optional, and the type of shape is iterable. The variable sas is the list of autoencoders, which are stacked together. The zero sliceof sas means the first stacked autoencoder, and the second autoencoder is one slice of sas. It is easy to analyze that there are three autoencoders in the stack. The h1, h2 variable is the middle trainable layer output. Finally, with the y, the final prediction, and the loss are defined, a complete train graph is finished. The correct\_prediction and accuracy are equipped with the evaluation step usage.

By the way, for a mirco-scope detail inspection, the loss function definition of a single autoencoder is defined as below:

self.cost = (tf.reduce\_sum(tf.pow(tf.subtract(self.reconstruction, self.x), 2.0))) + (self.lambda\_ / 2) \* (tf.reduce\_sum(tf.square(self.weights['w1'])) + tf.reduce\_sum(tf.square(self.weights['w2'])))

def partial\_fit(self, X):

cost, \_ = self.sess.run([self.cost, self.optimizer], feed\_dict={self.x: X})

return cost

The self means the object-oriented programming feature is used. Different from the regular mathematic calculation, the tensorflow library is used to keep compatibility with tensors. The next component of this snippets is a class method definition, which enables a batch of training, which is called partial\_fit in the program.

## 5.4 Implementation of Segmentation Module

The flow diagram of LSC Segmentation Module show below:

The main operation uses a massive of the internal/3rd-party library like numpy, scipy, scikit, but the main idea that accelerates the process of images is vectorizer operation. So, for brief an example of victoried operation is posed.

vpow = np.vectorize(lambda \_: \_ \* \_)

L1\_pow = vpow(L1[min\_x: max\_x, min\_y: max\_y] - centerL1[i])

L2\_pow = vpow(L2[min\_x: max\_x, min\_y: max\_y] - centerL2[i])

a1\_pow = vpow(a1[min\_x: max\_x, min\_y: max\_y] - centera1[i])

a2\_pow = vpow(a2[min\_x: max\_x, min\_y: max\_y] - centera2[i])

b1\_pow = vpow(b1[min\_x: max\_x, min\_y: max\_y] - centerb1[i])

b2\_pow = vpow(b2[min\_x: max\_x, min\_y: max\_y] - centerb2[i])

x1\_pow = vpow(x1[min\_x: max\_x, min\_y: max\_y] - centerx1[i])

x2\_pow = vpow(x2[min\_x: max\_x, min\_y: max\_y] - centerx2[i])

y1\_pow = vpow(y1[min\_x: max\_x, min\_y: max\_y] - centery1[i])

y2\_pow = vpow(y2[min\_x: max\_x, min\_y: max\_y] - centery2[i])

Codes of lines above is an example calculation of the kernel operation, weighted k-means, which is a kind of sub-matrix calculation. The slice of numpy array such as a1 in the coder can be slice over multi-dimension, here is two, to avoid the time consuming for loop.

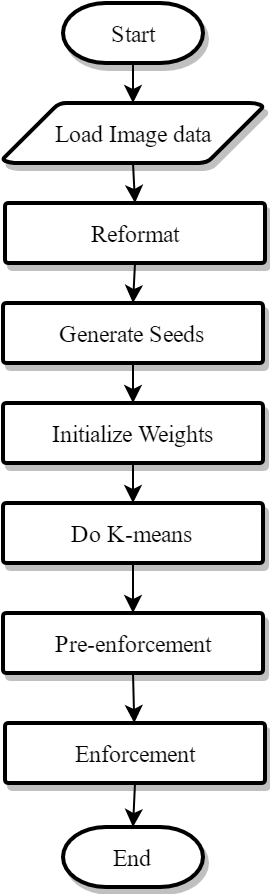


Figure 5.3 Flow Diagram of Segmentation Module

## 5.5 Implementation of Login Module

The login module checks the username and password user provided. Example implementation shown below:

db = persistents.get\_connection()

cursor = db.cursor()

cursor.execute("""SELECT `password` FROM `users` WHERE `username`=%s;""", username)

result = cursor.fetchone()

if result:

return \_get\_sha512\_hexdigest(password) == result[0]

else:

return False

Since the code is simple, there is no reason to explain each part of the coder. The front-end interface is as followed:

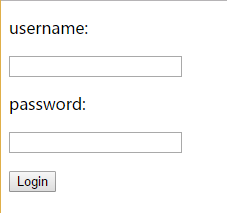


Figure 5.4 GUI Design of Login

The username and password is the only text user need to input to login the system.

## 5.6 Summary

This chapter introduces the module implementation of the feature extract module, classification module, login module and the segmentation module. The used functions and the flow diagram mainly composed the content of this chapter. Since the complexity of the project, the next chapter it to introduce the test case of the project.

**Chapter 6 System Testing**

The previous chapter introduces the flow diagram and the actual implementation of the This chapter is the conclusion of system testing and the classification module result, which is the main visible result can analyze.

## 6.1 Basic Test Methods Introduction

In case of the confuse of the testing, black box testing and white box testing are briefly introduced. Black box testing is often used in the actual production area and specific region when the source code is not known, such as from other organization/company, or the test group only proceed a quick testing over a huge component system. The main idea of black box testing is to test the all kinds of input, test the output/behavior of the system, suitable for a simple function. The shortage of the black box testing is that it is easy to ignore the internal bug caused by complex algorithm/logical design. Then the white box testing is introduced to prevent the shortage of unknown the internal infrastructure of the library/system. The white box testing, on the contrast, is an essential procedure in production system because the internal bugs is easy to find by inspecting the coder and system design documentation.

## 6.2 System Key Module Testing

For the simplicity of back-end functionality testing, the main modules tests are integrated into one paragraph. The test case of the all modules shown in table 6.1. Except the login module is excluded from the test case description, the other modules are implemented a test case in the test case table below as segmentation module, feature extraction module and classification module. The segmentation test is not proceeded through GUI due to the segmentation procedure is too long to put it into the Flask web framework. The test case 2 and 3 are proceeded through GUI. The classification module test only uploads the test data because the train data is uploaded in the previous test case.

| Table 6.1 Test case description of main modules | | | | | |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Operating** | **Input Data** | **Expected Results** | **Actual Results** | **Test Result** |
| 1 | Segmentation | Fused PET Data | Segmented super pixels | Multiscale superpixels | Same as expected |
| 2 | Feature extraction | Multiscale superpixels | Feature vectors | Visible feature maps | Same as expected |
| 3 | Classification | Feature maps | Accuracy percentage | Accuracy floats | Same as expected |

The brief test cases describe the overall system test case processed in the system testing.

Black box testing is mainly used in the GUI testing because it is easy to figure out the output/status of the web page, which is a complement of previous system implementation chapter.

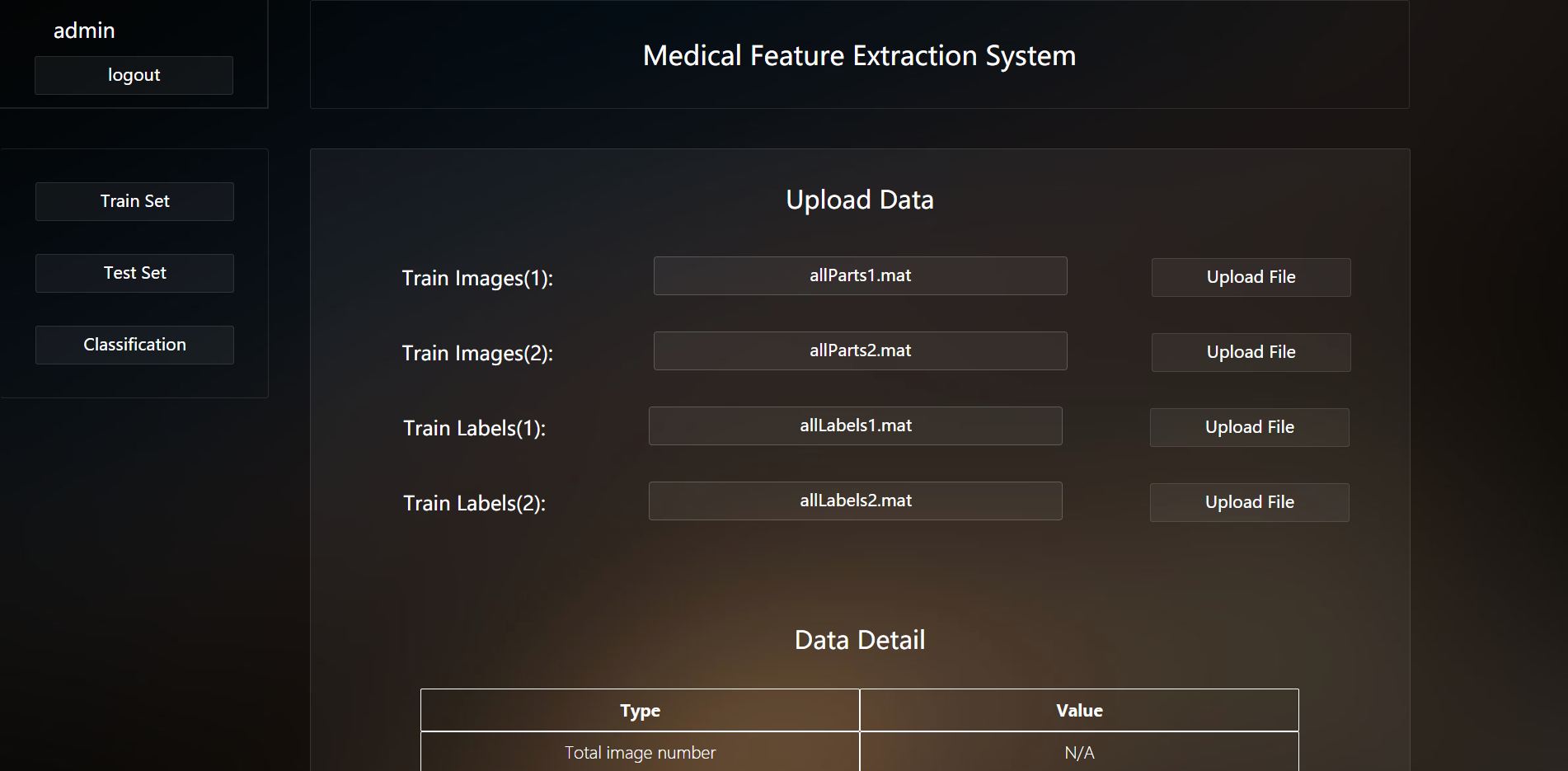


Figure 6.1 Actual Front-end Graphic User Interface Example

As shown in the figure 6.1 the main modules are displayed as the sidebar buttons, one module per button. And the kernel module, feature extraction module result shown in figure 6.2. The full version of feature map download is under developmental phase. But with the random generator, in python programming language, the random chose visible version of feature map is shown as simple way of generated static file in the server.

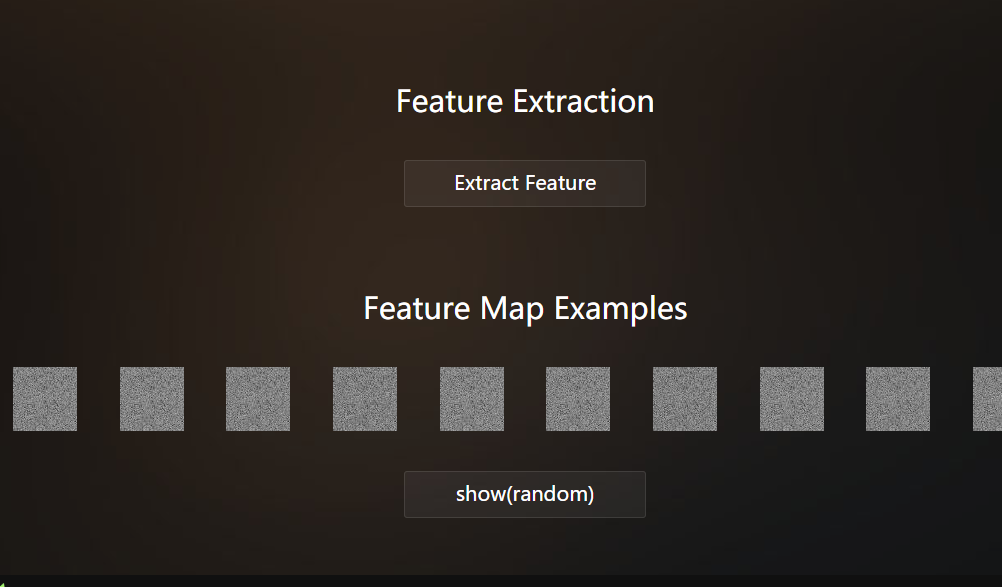


Figure 6.2 Feature Extraction Result

## 6.3 Classification Results

As the final module’s GUI test result, the single 6.3 paragraph is used to emphasize.

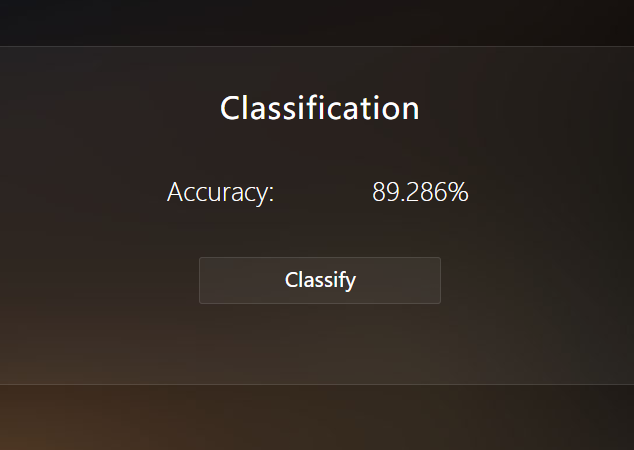


Figure 6.3 Classification Result Example

The all segmentation as the train set except only 1024 background images superpixel because the liver image and lymphoma image number are much less than the background image num. The number of liver, lymphoma, background superpixels is 574, 898, 25593, respectively. The test set superpixel number is 252,138 and 1024, respectively. The classification evaluates the accuracy shown below as figure 6.3.

For the accurate result, the final classification takes three experiments’ average. The accuracy table show as table 6.2. In each row, the AE means autoencoders. 2AE represents two autoencoder with size of [4096, 256, 6]. 3AE represents three autoencoder which size is [4096, 512, 64, 6].

Table 6.2 Final Classification Result

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Liver** | **Lymphoma** | **Background** |
| Accuracy(2AE) | 88.1% | 90.8% | 79.4% |
| Accuracy(3AE) | 88.9% | 96.2% | 84.5% |

As shown in the figure 6.4, the three autoencoders are with the line style of solid, the dashed line is the accuracy of two autoencoders. The reason why there is no one autoencoder or four autoencoders is that one autoencoder’s accuracy is zero, indicated the autoencoder cannot describe the feature in one layer, and when my laptop interpreting the four level autoencoders version of SAE, the memory is out of use. So, there is only two lines in the figure 6.4.

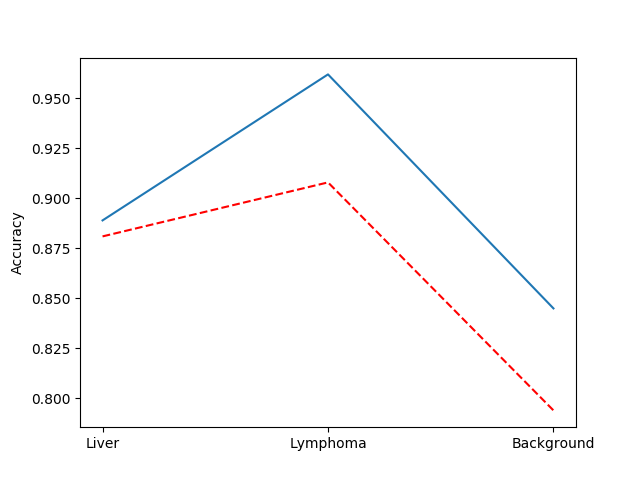


Figure 6.4 Classification Result, solid lines are the 3AE type, and the dashed are 2AE type

## 6.4 Summary

This chapter introduces the main test mythology of the system test, and through the black-box testing, the main test cases are divided into three groups which represents the three-main module in the system design except the login module. The GUI test is also included in this chapter. As a result of classification, the classification result is also included in this chapter. Next chapter the summary is to be introduced.

**Chapter 7 Conclusion**

In the previous chapter, we introduce the testing technology and the graphic user interface test. The classification results revealed the effectiveness of the feature extraction module. In this chapter, summary the total work and the future work direction.

## 7.1 Summary of the Work

This project mainly introduces the system used for medical feature extracting, which are mainly focus on the segmented result of the linear spectral clustering algorithm. Then the system is mainly designed as a B/S model for future compatibility

This project using linear spectral clustering algorithm as the segmentation solution to the image data that processed by the groupmates. The data source drives from a hospital in mainland China.

The VGG-19 net is used to extract the feature of each segmentation result from the previous step. Only 17 of 19 weighted nets are used to extract a feature vector of length 4096. The feature vector is to be generate as feature maps.

Stacked Autoencoders are applied in the classification module to analyze the effectiveness of the feature extraction module, different types of feature extraction module are used to generate the currently accuracy.

## 7.2 The Future Work

We are mainly focus on the shortage of the project. The PyCharm is a commercial IDE which requires a lot of money to buy the license. Though it is powerful, the project development is still under risk of code stealing.

Processed well on the LSC algorithm, but we have found some shortage in achieving the python version of this algorithm. The algorithm uses pre-enforcement and enforcement step to merge small superpixels to the near superpixels. But it may be confusing if we are actually focus on a small part of the image, which will require the further process to extract the small are of the target image block. This project is not an end-to-end program, making it harder to use if the user has not software engineering experience.

**References**

1. Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.
2. Agrawal A, Lu J, Antol S, et al. VQA: Visual Question Answering[J]. International Journal of Computer Vision, 2015, 123(1):1-28.
3. Jiang Y, Li Z, Zhang L, et al. An Improved SVM Classifier for Medical Image Classification[C]// Rough Sets and Intelligent Systems Paradigms, International Conference, RSEISP 2007, Warsaw, Poland, June 28-30, 2007, Proceedings. DBLP, 2007:764-773.
4. Zhou, Xiangrong, et al. First trial and evaluation of anatomical structure segmentations in 3D CT images based only on deep learning[J], Medical Imaging and Information Sciences,2016,33(3):69-74.
5. Lu F, Wu F, Hu P, et al. Automatic 3D liver location and segmentation via convolutional neural network and graph cut[J]. International Journal of Computer Assisted Radiology & Surgery, 2017, 12(2):171-182.
6. Bi L, Kim J, Kumar A, et al. Automatic detection and classification of regions of FDG uptake in whole-body PET-CT lymphoma studies[J]. Computerized Medical Imaging & Graphics the Official Journal of the Computerized Medical Imaging Society, 2016.
7. Li Z, Chen J. Superpixel segmentation using Linear Spectral Clustering[C]// Computer Vision and Pattern Recognition. IEEE, 2015:1356-1363.
8. Phillip J. Eby. Python web server gateway interface v1.0[EB/OL]. https://www.python.org/dev/peps/pep-0333/
9. Convolutional Neural Networks (CNNs/ConvNets). [EB/OL]. http://cs231n.github.io/convolutional-networks/
10. Jay Ricco. What is max pooling in convolutional neural networks. [EB/OL]. https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks
11. Autoencoders and Sparsity. [EB/OL]. http://ufldl.stanford.edu/wiki/index.php/Autoencoders\_and\_Sparsity
12. Stacked Autoencoders. [EB/OL]. http://ufldl.stanford.edu/wiki/index.php/Stacked\_Autoencoders
13. k-means clustering[EB/OL]. https://en.wikipedia.org/wiki/K-means\_clustering
14. Flask web development, on drop at a time[EB/OL]. http://flask.pocoo.org/

**Acknowledgements**

First of all, I would like to express my gratitude to my parents and all my relatives. It is my parents gave me the strong soul to overcome the difficulties these months in doing the experiment and the economic support. My cousin, OsirisY, who gave me useful opinions on the word/paragraph format, and guided me on the principle of the paper layout. My brother Shi and his wife, Ms. Dong, invited me to talk when I was mental breakdown, let my experiment was able to continue. My roommate Eric, helped me a lot in the paper content and the experimental module design. Most importantly, the front-end project technology and CSS style. My roommate Zhou Shuai helped me a lot in database design. Thanks God for giving me the brain that could barely finish the graduation work

Second, I am here to express my sincere thanks to the free diagram tools in draw.io, which is helpful when constructing simple diagram including UML and all kinds of design diagrams. The Google Drive also did a significate contribution to my paper’s version control. And the video games provided by steam, especially nekopara series, which gave the main power of finishing my project.

Last and special, my thanks would go to my professor Jiang, and the classmate Lee, who gave me a lot of help on my project, including the project approval and technical support. The related work of medical feature extraction system is greatly appreciated by all the person who helped me during these months.

Finally, I would like to have grateful thanks to the professor and the classmates who helped me again!