上海交通大学学位论文

超声成像颈动脉斑块特征与临床症状的关系

姓　名：孙雨、夏翌、肖子聪、徐梓豪、张泽宇
学　号：519021911195, 519021910074, 519370910134, 519370910174, 519021910038
导　师：黄佩森
学　院：密西根学院
学科/专业名称：电子与计算机工程
申请学位层次：学士学位

2023 年 8 月

A Dissertation Submitted to
Shanghai Jiao Tong University for Bachelor Degree

RELATION OF CAROTID PLAQUE FEATURES DETECTED WITH ULTRASOUND IMAGING TO CLINICAL SYMPTOMS

Author: Yu Sun, Yi Xia, Zicong Xiao, Zihao Xu, Zeyu Zhang
Supervisor: Peisen Huang

UM-SJTU Joint Institute
Shanghai Jiao Tong University
Shanghai, P.R.China
August 1st, 2023

摘　要

颈动脉斑块是颈部血管中积聚的脂肪沉积物。作为中风的主要原因，需要及时发现和干预。超声成像作为一线检测工具，具有成本效益，但依赖于资深放射科医生的解读和解读。该项目旨在开发基于软件的方法，提取颈部超声图像的特征并预测临床症状。根据我们收集的文献，我们将该领域现有的基准与相关的评估指标进行了比较，并总结了他们实验中的主流方法。基于有限的 322 张超声图像，我们成功构建了一个基于机器学习的模型，该模型结合了用于图像输入的卷积神经网络和用于患者临床病史数据的决策树模型。最终测试准确率达到 80.56%，满足了主办方的要求。在此基础上，我们还设计了一个简单的网页应用，用于与用户进行交互，便于用户检查模型的输出结果。我们希望我们的设计能够在未来的颈动脉斑块自动诊断方面发挥作用。

关键词：颈动脉斑块，卷积神经网络，机器学习，分类

ABSTRACT

Carotid plaques are fatty deposits accumulating in neck blood vessels. As a predominant cause of stroke, it requires timely detection and intervention. Ultrasound imaging serves as the first-line detection tool, which is cost-effective but relies on senior radiologists to read and interpret. The project is to develop software-based methods, to extract features and predict clinical symptoms in neck ultrasound images. Based on the literature we collected, we compared the current existing benchmarks in this field with relevant evaluation metrics included, and summarized mainstream methods in their experiments. Based on limited 322 ultrasound images, we managed to build a machine-learning-based model, which incorporates convolutional neural network for image input and decision tree model for patients' clinical history data. The final test accuracy reaches 80.56%, which meets the requirement of our sponsor. Also, on this basis, we designed a simple web application for user interaction, so that users can check the output results of the model. We hope our design will make a difference in automatic diagnosis of carotid plaque in the future.

Key words: carotid plaques, convolutional neural network, machine learning, classification

Content

Chapter One Introduction

1.1 Background

Carotid artery diseases are one of the predominant causes of stroke [1]. These diseases occur when fatty deposits, known as carotid plaque, accumulate in the neck blood vessels. These vessels are responsible for delivering oxygen to the brain, making them essential for healthy brain function.

Accumulation of carotid plaque can cause blockages in vessels, either when reaching thresholds or when breaking off the artery wall. Blood supply will be cut off where the blockage takes place, further resulting in the lack of oxygen supplement.

One noteworthy characteristic of carotid plaque is its slow development. Individuals often experience few symptoms when the plaque is in early stages, and hardly realize its existence until they are caught in stokes. The potential risks of carotid plaque adds to the necessity of early detection and timely medical intervention.

Ultrasound imaging has emerged as the first-line detection tool for symptomatic or asymptomatic carotid plaques [2] considering its cost-effectiveness and non-invasive nature. However, ultrasound images rely on senior radiologists to read and interpret. The need for trained experts consumes labor source, and can cause delays in treatment.

To address the challenge, the project is initiated to develop software-based methods capable of extracting carotid plaque features from neck ultrasound images. These extracted features can then be utilized to explore their relationship with clinical symptoms. The ultimate goal in this project is to construct a robust model providing binary classification results, and categorizing ultrasound images as either symptomatic or asymptomatic.

The software-based approach aims to bring automation in the diagnostic process for carotid artery diseases. The work could potentially relieve the burden on radiological experts and improve the efficiency and accuracy in diagnosis. Furthermore, the function to distinguish plaques at different stages can help personalized the medical plan for different patients, and improve their medical experience.

1.2 Literature Review

To get a better understanding of the current stage of carotid plaque detection, a literature review was conducted on the topic of carotid plaque feature extraction and classification. We investigated into 5 papers in last decade and will summarize their methods and strengths. All those papers can be retrieved by using keyword "carotid plaque AND feature detection OR feature classification" on IEEE and CNKI database.

1.2.1 Overview

In all 5 papers, the mainstream method is a combination of image feature extraction and logistic regression for label classification. Also, several dimensionality reduction algorithms are applied in order to lower the computation cost and improve the interpretability of the model.

Authors	#Image	Accuracy	Sensitivity	Specificity	AUC
Huang, Z.[1]	384/164	85.4%	94.7%	64.0%	0.947
Golemati, S.[2]	150	N.A.	N.A.	N.A.	N.A.
Huang, X.[3]	136	89.0%	N.A.	N.A.	N.A.
Xu, M.[4]	94	85.1%	89.230%	75.860%	0.887
Luo, Y.[5]	120	96.6%	88.8%	98.7%	0.894

Tab. 1-0 Benchmark of Carotid Plaque Identification

In Table 1-0, size of dataset and available metrics of the experiment is shown for easier comparison. Notice that only in [1], train test split is performed. Therefore, both experimental results on train set and test set are recorded in Table 1-0 for [1].

1.2.2 Summary of Methods

Methods used in those papers can be generally categorized into 3 major steps: **feature extraction**, **feature selection/reduction** and **classification**. Several important methods will be elaborated in this minor section for future reference.

- **Feature extraction**:

  The first step is to extract as many significant features from the raw image dataset as possible.

  In [2], Curvelet transform was applied on ultrasound images to create handcraft features. As an extension to Fourier transform, for multi-dimensional signals, it not only preserves directional information, but also makes local angular properties is scale-dependent. Therefore, it is considered a robust transform algorithm.

  Another method includes using gray-level co-occurrence matrix[3], which is texture analysis algorithm describing periodic regular patterns of gray-level distribution on the image. It efficiently calculates the spatial correlativity of gray-level pixels on the image.

- **Feature selection/reduction**:

  Usually, feature extraction can be performed by existing software packages, which results in redundant features. For many online models or time-constrained models, the limitation on computation resource should be taken into serious consideration. Therefore, insignificant features should be eliminated, or the a dimension-reducing mapping should be applied on the original feature space in order to improve model's efficiency.

  A common choice for feature selection will be significance test[2, 3, 4]. By conducting significance test on statistics of a single variable/feature, it's easy to judge whether this statistics is significantly different in both groups based on p-value.

  Also, feature selection can be performed by finding the importance of each features, which can be finished with random forest algorithm[4]. During training, the decision tree will calculate how many impurity/entropy is decreased by introducing a specific feature. A feature with larger impurity reduction is considered more important.

Instead of selecting features from pre-extract features, it's possible to construct a function to map original features into a low dimensional space. In [3], Linear Discriminant Analysis (LDA) is applied to reduce the dimension of features. Compared to PCA, LDA performs better with the existence of labels because it aims to maximize the separation of two groups.

- **Classification**:

  The final step is to generate label predictions based on features.

  In [1, 4], logistic regression is used to classify the symptomatic and asymptomatic carotid MRI images. Though simple and common, it's considered very effective for simple classification tasks, and it's the foundation for many complex classifiers, like MLP (multilayer perceptron) classifier[3].

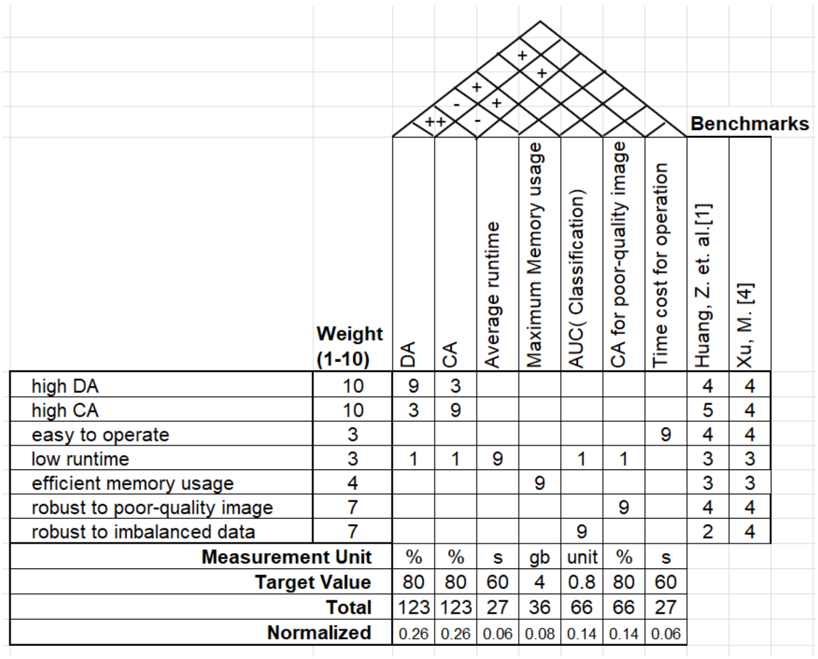
  MLP is a basic feedforward artificial neural network. At each layer, nonlinearity is introduced by using non-linear activition function. Therefore, compared to classification with linear boundary, it has better performance.

  Besides, Support Vector Machine (SVM)[5] is a powerful supervised learning algorithm. With the help of kernel function, features are mapped into hidden spaces for future linear separation. Compared to deep learning methods which usually requires large amount of data, it is powerful when the data size is small.

Chapter Two Requirements and Specifications

2.1 Customer Requirements

Based on discussions with instructor and investigations in works of relevant topics, we have listed 7 major demands of the sponsor as shown in the QFD chart below. For simplification, we denote detection accuracy as DA and classification accuracy as CA in the QFD chart. Among the customer requirements, the accuracy in terms of detecting the plaque and identifying symptomatic patients and robustness of the program are the major concerns here, while other demands including low runtime, being easy to operate and being memory efficient play relative minor roles. As shown here in this QFD chart, two benchmarks introduced in previous sections in general performed well in terms of the major demands, which sets a higher standard for our final outcome.



									Benchmarks	
	Weight (1-10)	DA	CA	Average runtime	Maximum Memory usage	AUC(Classification)	CA for poor-quality image	Time cost for operation	Huang, Z. et. al.[1]	Xu, M. [4]
high DA	10	9	3						4	4
high CA	10	3	9						5	4
easy to operate	3							9	4	4
low runtime	3	1	1	9		1	1		3	3
efficient memory usage	4				9				3	3
robust to poor-quality image	7						9		4	4
robust to imbalanced data	7					9			2	4
Measurement Unit		%	%	s	gb	unit	%	s		
Target Value		80	80	60	4	0.8	80	60		
Total		123	123	27	36	66	66	27		
Normalized		0.26	0.26	0.06	0.08	0.14	0.14	0.06		

Fig. 2-1 The QFD Chart

2.2 Engineering Specifications

The following is Table 2-1 listing our specific engineering requirements, including engineering specifications and corresponding expected values. The DA and CA corresponds to the accuracy of the model. The Average runtime measures the speed of the model. AUC and CA for poor-quality image evaluates the robustness of the model. The maximum memory usage sets a standard for the memory consumption of the program, while time cost for operation measures the simplicity of understanding the interfaces of the model.

DA	**CA**	**Average Runtime**	**Maximum Memory Usage**
80%	80%	60 s	4 GB
AUC	**CA for Poor-Quality Image**		**Time Cost for Operation**
0.8	80%		60 s

Tab. 2-1 Specific Engineering Requirements

Chapter Three Concept Generation

3.1 Brainstorm

The requirement to accurately classify plaques of different patients, implies that we need to adopt ML(machine learning) or DL(deep learning) to train the model. Moreover, pre-trained network structures such as Resnet18, Resnet50 and VGG16 are preferred, since they have already been proved efficiency in millions of image classification tasks. And the limited quantity and quality of images should be further bridged by data enhancement and data preparation. The further analysis will be conducted by morphological analysis. Further more, we should also consider how to assemble medical information of patients with our trained model, to derive a more accurate platform. The methods, after we have searched and literature review, are stacking, bagging and boosting.

3.2 Morphological Analysis

3.2.1 General Function Structure

In this part, we conduct physical decomposition to identify functions of this project. Noticing that since our project is totally software-based, so that there are no energy in the function structure. Also, the signals in this project is all given by human including manually labelling images and hyper parameter tuning. Further more, we decompose the functions into sub-functions and configure function chains into complete functional model. The function structure is as follows: Figure 3-2 and Figure 3-3.

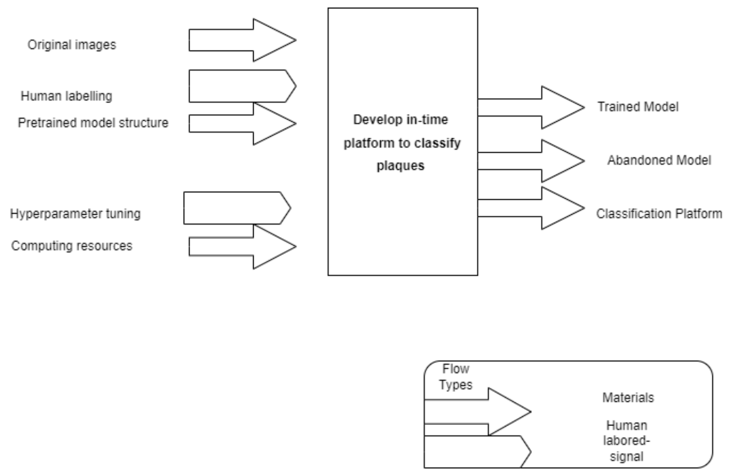


Fig. 3-2 General Function Structure

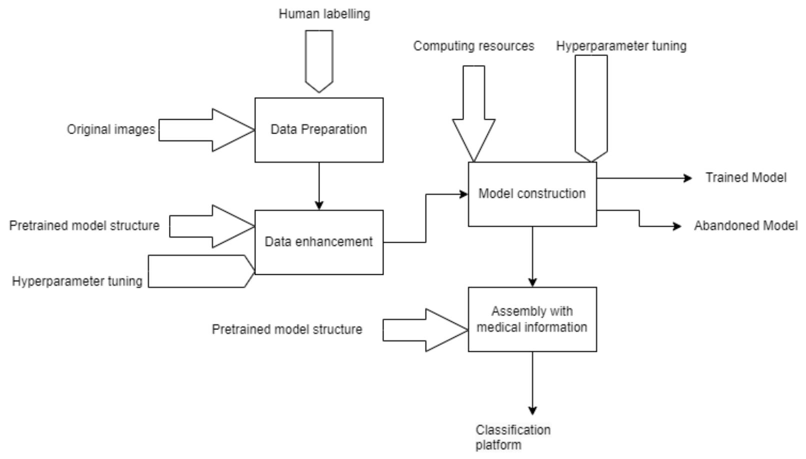
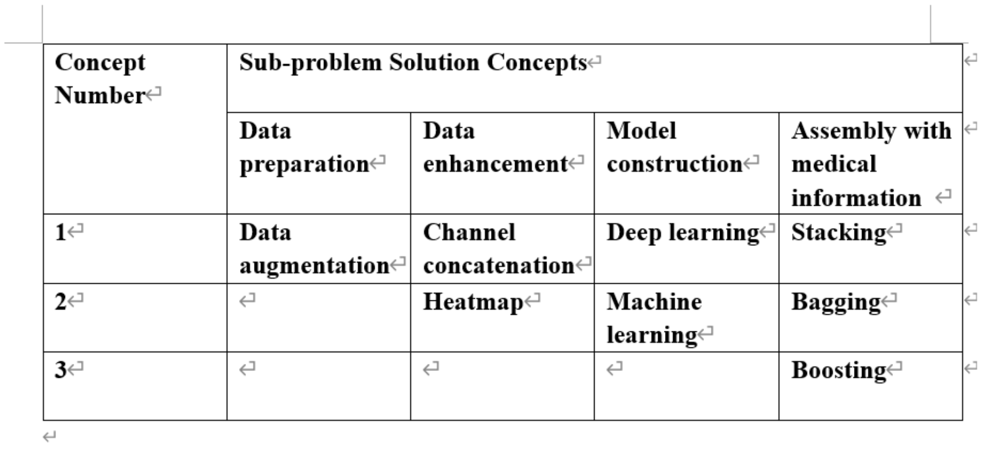


Fig. 3-3 Decomposed Function Structure in details

Thus we can divide our projects into multiple sub-functions: data preparation, data enhancement, model construction, assembly with medical information. Further solutions can be derived by brainstorm.The final sub-problem solution concepts we generated are as follows:

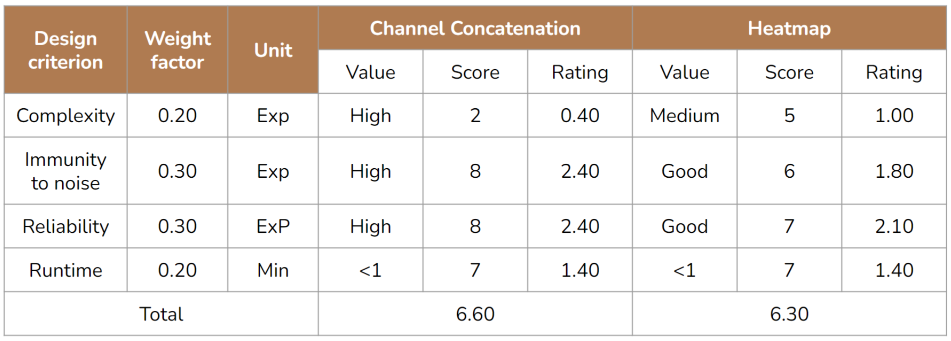


Concept Number	**Sub-problem Solution Concepts**			
	Data preparation	**Data enhancement**	**Model construction**	**Assembly with medical information**
1	**Data augmentation**	**Channel concatenation**	**Deep learning**	**Stacking**
2		**Heatmap**	**Machine learning**	**Bagging**
3				**Boosting**

Fig. 3-4 Sub-problem solution concepts

Chapter Four Concept Selection Process

After we have generated our concepts, we should further select them based on customer requirements and engineering specification. What's deserve noticing is that, the data preparation is usually done by data augmentation which is a assembly of flipping, rotating and so on. And we need to select desired concept for data enhancement, model construction and assembly with medical information. Among them, the data enhancement mainly contributes to DA(detection accuracy) and model construction mainly contributes to both DA and CA, while assembly with medical information contributes to CA. The selection weighted matrix is listed:



Design criterion	Weight factor	Unit	Channel Concatenation			Heatmap		
			Value	Score	Rating	Value	Score	Rating
Complexity	0.20	Exp	High	2	0.40	Medium	5	1.00
Immunity to noise	0.30	Exp	High	8	2.40	Good	6	1.80
Reliability	0.30	ExP	High	8	2.40	Good	7	2.10
Runtime	0.20	Min	<1	7	1.40	<1	7	1.40
Total			6.60			6.30		

Fig. 4-5 Selection Matrix: Data enhancement

For data enhancement, our selection criterion is mainly based on their ability to improve DA, complexity and run time, the reliability and immunity to noise are important factors for improving DA especially in poor quality images, so we give them more weight. As a result we select Channel concatenation as our data enhancement method.

Design criterion	Weight factor	Unit	DL			ML		
			Value	Score	Rating	Value	Score	Rating
Complexity	0.10	Exp	High	2	0.20	Medium	5	0.50
Reliability	0.60	Exp	High	8	4.80	Good	6	3.60
Training time	0.10	Hour	~10	2	0.20	~1	8	0.80
Runtime	0.20	Min	<1	7	1.40	<1	7	1.40
Total			6.60			6.30		

Fig. 4-6 Selection Matrix: Model construction

For Model construction, since this step is the key steps in terms of deciding CA and DA, we put more emphasis on reliability than all other criterion. And it is also the most time-consuming and computing resource-demanding step, so we give another 40% for these criterion. As a result, we select DL as our final model construction method.

Design criterion	Weight factor	Unit	Bagging			Boosting			Stacking		
			Value	Score	Rating	Value	Score	Rating	Value	Score	Rating
Complexity	0.20	Exp	Low	2	0.40	Medium	6	1.20	Medium	5	1.00
Reliability	0.60	Exp	Medium	6	3.60	High	8	4.80	High	8	4.80
Training time	0.20	Exp	Fast	8	1.60	Slow	3	0.60	Medium	5	1.00
Total			5.60			6.60			6.80		

Fig. 4-7 Selection Matrix: Assembly with medical information

For Assembly methods, while the run time in this step is negligible compared to model construction, we delete the criteria of run time for this step. As a result, we choose Stacking as our final assembly with medical information concept.

Apart from that, given the criterion, we concluded pros and cons of our top 5 concepts:

4.1 Advantages and Disadvantages of top 5 concepts

Considering that different process is almost individual, we tried different combinations and derived pros and cons below. The reason why we almost adopt DL is that there are some pre-trained networks in DL such as ResNet and VGG that has been feed to billions of images before, for us to choose.

4.1.1 Channel Concatenation+ DL+Stacking

- Advantage: high DA and CA, good performance in poor quality images.
- Disadvantage: high complexity, large memory usage, long run time and training time.

4.1.2 Heatmap+ DL+Stacking

- Advantage: high CA, medium performance in poor quality images.
- Disadvantage: medium DA, medium complexity, large memory usage, long run time and training time.

4.1.3 Channel Concatenation+ DL+Boosting

- Advantage: high DA and high CA, good performance in poor quality images.
- Disadvantage: high complexity, long run time and even longer training time.

4.1.4 Heatmap+ DL+Boosting

- Advantage: high CA,medium complexity.

- Disadvantage: medium performance in poor quality images, medium DA, long run time and training time.

4.1.5 Channel Concatenation+ DL+Bagging

- Advantage: high DA, medium complexity, medium training time, good performance in poor quality images.
- Disadvantage: medium CA, long run time.

Chapter Five Selected Concept Description

5.1 Overview

The final detailed design encompasses a series of modules as depicted in Figure 5-8. The process begins with doctors directly using the original ultrasound image as input, regardless of image size. The image then undergoes several modules for accurate plaque delineation, image augmentation, plaque classification, and medical history analysis. The ultimate outcome is a diagnostic report involving the classification result, either symptomatic or asymptomatic, and corresponding confidence level, ranging from between 0 and 1.

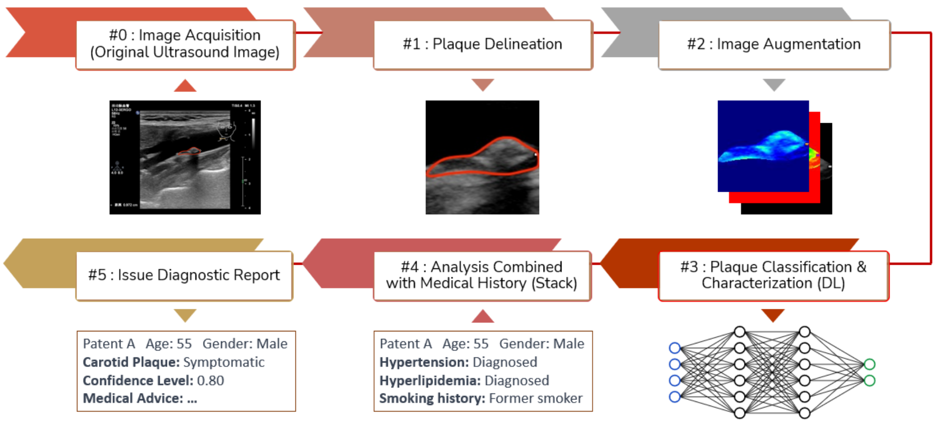

#0 : Image Acquisition (Original Ultrasound Image)
#1 : Plaque Delineation
#2 : Image Augmentation
#3 : Plaque Classification & Characterization (DL)
#4 : Analysis Combined with Medical History (Stack)
#5 : Issue Diagnostic Report
Patent A Age: 55 Gender: Male
Carotid Plaque: Symptomatic
Confidence Level: 0.80
Medical Advice: ...
Patent A Age: 55 Gender: Male
Hypertension: Diagnosed
Hyperlipidemia: Diagnosed
Smoking history: Former smoker

Fig. 5-8 Final detailed design (Concept diagram)

5.2 Engineering Design Analysis

5.2.1 Plaque Delineation

The ultrasound image received from the hospital is a gray-scale representation of carotid artery (Figure 5-9). The dark part represents the inner region of carotid artery, while the bright parts on either side represent the carotid artery wall. Additionally, the carotid plaque region of interest, attached to artery wall, has been manually outlined in red by the doctors. To assist in easier identification and segmentation of the plaque, the hospital also provides a corresponding white mask image (Figure 5-10). This mask image helps indicate the precise shape and location of the carotid plaque.

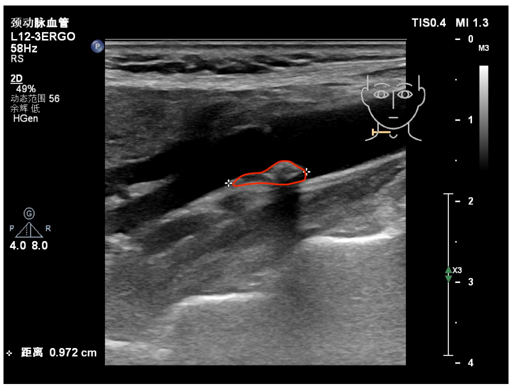


Fig. 5-9 Original ultrasound image

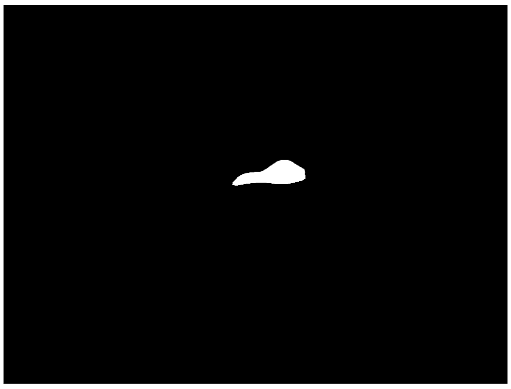


Fig. 5-10 White mask image

The first function of plaque delineation module is to actually trace the plaque region from the original image, while discarding any redundant parts. Bitwise AND operation is employed between the two images pixel by pixel. When the corresponding pixel in the mask image is black, represented as $(255)_{10} = (11111111)_2$, the output pixel remains black. On the other hand, when the corresponding pixel in the mask image is white, represented as $(0)_{10} = (00000000)_2$, the output pixel retains the value from the original image. Then only the plaque region remains visible on the new image, and the remaining areas are covered in black (Figure 5-11). The implementation utilizes *bitwise_and* function from *OpenCV* library.

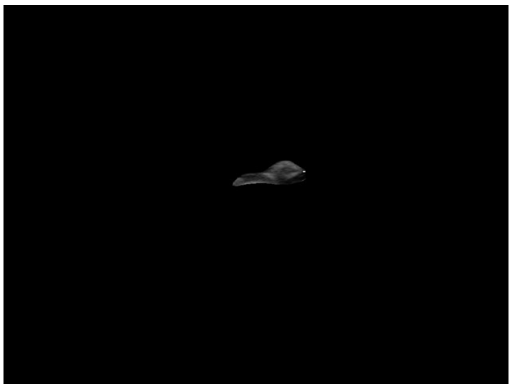


Fig. 5-11 Delineated image

Due to variations in carotid plaque sizes among patients, some plaques may only cover a small area in the new image, while others can occupy up to $1/4$ of the image. The subsequent function crops the image, using the smallest possible rectangle that contains the entire plaque. To maintain consistency and achieve a square shape for every output image (Figure 5-13), black padding is added around the cropped region. The padding ensures that each output image has the same dimensions, regardless of the size of the original plaque. The implementation utilizes *findContours* and *boundingRect* functions from *OpenCV* library.

5.2.2 Image Augmentation

Given the limited size of the image set, which is a common challenge in medical image resources, the image augmentation module plays a vital role. It helps expand the training set size and provides more diverse examples for the

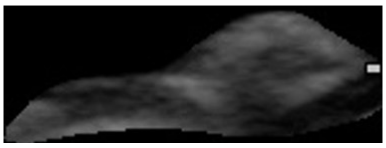


Fig. 5-12 Cropped image (rectangle)

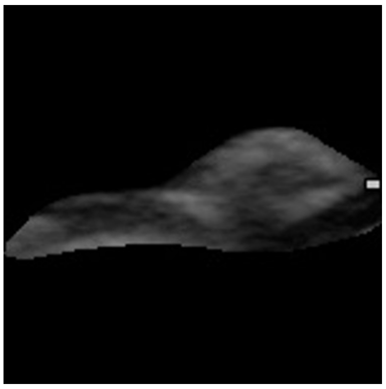


Fig. 5-13 Cropped image (square)

neural network to learn from, enhancing its ability to generalize and converge during the training process. It also helps extract features from cropped regions with variations in position, scale, and orientation. The diversity of data improves the model in better identifying symptomatic instances during the testing process.

1. **Random Linear Transform**

   Deep neural networks typically require a relatively large dataset to build effective models. A limited dataset, however, can lead to a risk of overfitting issues and produce biased models. Considering the difficulty in collecting more ultrasound images, a practical approach is to apply linear transformations to the original images, thereby creating additional samples. Transformation methods have been carefully chosen in final design, including horizontal flip, vertical flip, random rotation, random affine transformation, and sharpness adjustment. For each image in the training directory, the augmentation module retains the original sample and generates three additional augmented samples using these transformations (Figure 5-14). This process results in a total of 1000 images being created in the augmented training directory. The implementation relies on $transforms$ function from $torchvision$ package. The transformations only apply to training images as we want a single model to give only one result for a single patient during testing process.

2. **Contrast Ratio Enhancement**

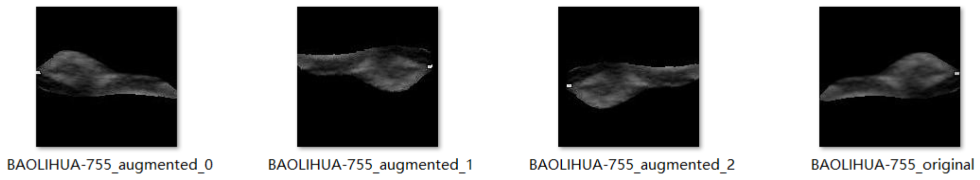


Fig. 5-14 Augmented images

Another issue that arises in the image set is dim images, where the plaque region becomes difficult to distinguish from the black background. To address this, the module incorporates a contrast enhancement process to improve the visibility and clarity of the plaque region in all images. By applying a consistent degree of contrast enhancement to all images, the module effectively enhances the differences in brightness between the plaque region and the background (Figure 5-15). The implementation is realized with help of $createCLAHE$ function from $OpenCV$ library.

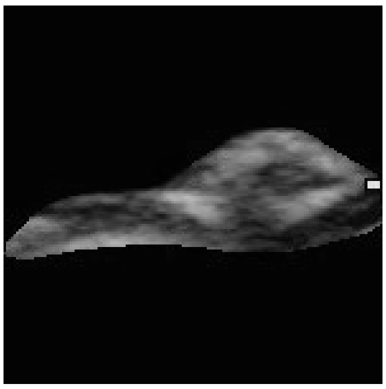


Fig. 5-15 Contrast-ratio-enhanced image

3. **3-Channel Transform (Optional)**

The pretrained convolutional neural network, introduced in the subsequent module, is fed with ImageNet dataset, containing thousands of 3-channel (RGB) images. The model can occasionally exhibit a preference for 3-channel images over gray-scale images. The module also provides two options to transform the ultrasound images into 3-channel ones. One is heatmap transformation, mapping the gray scale to a heatmap jet color scale

(Figure 5-16). Another is B-channel transformation, where the gray-scale image is read as a 3-channel image, and the output is retained in the B-channel only (Figure 5-17). The implementation relies on *Matplotlib* package.

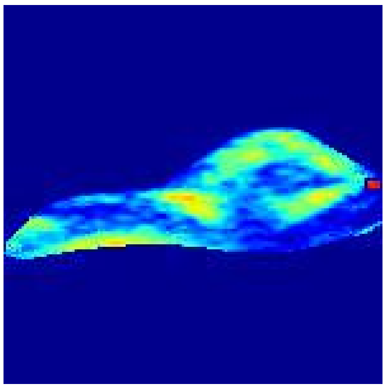


Fig. 5-16 Heatmap transformed image

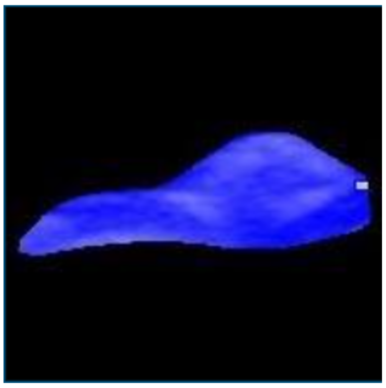


Fig. 5-17 Channel-B transformed image

5.2.3 Plaque Characterization & Classification

1. **Deep Learning & ResNet**

   In our final concept, deep learning methods was applied to tackle with the problem in medical ultrasound image characterization and classification. Deep learning has emerged as an important tool in the field of medical imaging, with particular significance in the classification of ultrasound images. Ultrasound technique generates large amount of data that can be challenging to interpret due to intrinsic noise and subtle differences among conditions, which may require a group of experienced senior radiologists to interpret. However, deep learning algorithms can excel in this complexity, extracting meaningful insights that might escape human detection. These algorithms use layers of artificial neurons to learn hierarchical feature representations. By automating this interpretation process, deep learning can increase the accuracy, speed, and consistency of ultrasound image classification, thus reducing the workload of radiologists, especially in the situation when time and human resources are constrained.

Specifically, Residual Network (ResNet) was chosen as the primary neural network. ResNet was a deep neural network structure proposed by Kaiming He, et al. in order to ease the difficulty in training very deep neural networks [6]. The most revolutional innovation in ResNet is the invention of "shortcut connection" (Figure 5-18), which means previous representations are skipping one or more layers to form a integrated mapping with future layers so that the can network is able to learn identity mappings between layers.

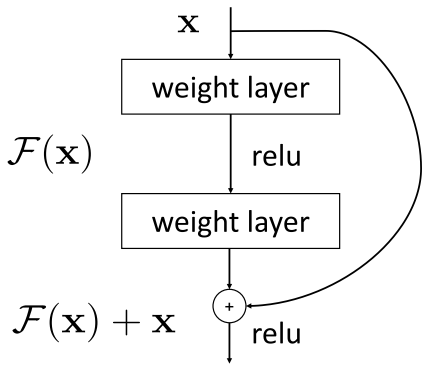

$\mathbf{x}$
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
+
relu

Fig. 5-18 Residual learning: shortcut connection

Suppose the underlying mapping to be learned is $\mathcal{H}(\mathbf{x})$, then the residual function can be characterized as $\mathcal{H}(\mathbf{x}) - \mathbf{x}$. By shifting the problem in this way, the difficulty of learning optimal mapping such as identity mapping is alleviated, as proven that the effect of training degradation phenomenon could be reduced in the experiment.

In our project, due to the subtleness of ultrasound imaging features, it's necessary to use deeper neural network to extract the underlying regular patterns embedded in ultrasound images, while not suffering from degradation phenomenon. In detail, global features of ultrasound images will be extracted, which serve as a component of the final image classifier.

2. **Local Echo Features Extraction & K-means**

Global features of the ultrasound images provides a general picture of the artery situation of the patient, while the echo features of the plaque focus more on the local properties based on gray values of ultrasound image of carotid plaque.

In ultrasound imaging field, objects with different characteristics will have different gray values on the ultrasound images. This is because harder objects will reflect more ultrasound waves than softer objects, such as lipid or fibrous tissues. Therefore, by investigating the proportion of the components of the plaque, we may find potential relationship between the echo features and risk probability of the plaque [7].

In order to quantify the proportion of the plaque components, we applied clustering algorithms to categorize gray values in Region of Interest (ROI), which was the area delineated by radiologists on the image. To put it simply, pixels with close gray values will be grouped into the same cluster (Figure 5-20). In our case, the final processing result will be a four-dimensional feature vector, which represents the proportion of pixels with four different levels of brightness to the total number of pixels. This echo feature, along with global feature derived from the previous step, will be fed into global classifier to generate the final prediction on the property of the carotid plaque.

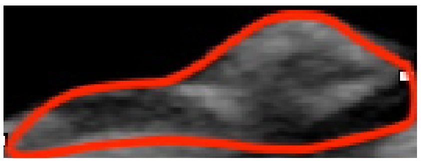


Fig. 5-19 Original ROI image

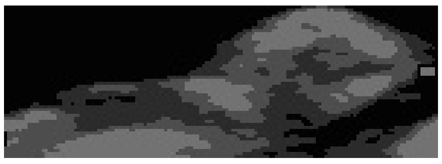


Fig. 5-20 Clustered ROI image

In order to cluster the pixels, K-means clustering algorithm was applied [8]. The general purpose of the algorithm was to partition n data-points into k clusters so that data-points within the same cluster are close to each other, while inter-cluster distance are as large as possible, based on a given metric. The pseudo-code of the algorithm is attached in the appendix.

3. **Feature Fusion and Classification**

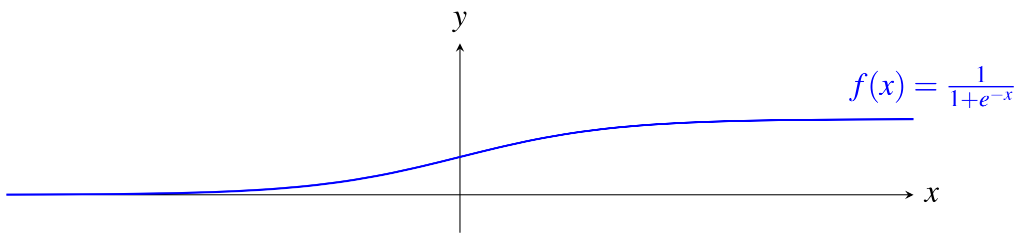


Fig. 5-21 Logistic function: "Sigmoid" function

Based on the derived global feature vector and local echo feature vector, we can simply concatenate them and form a new one as the input to the final binary classifier, which is based on logistic regression. Logistic regression is a crucial statistical model commonly employed in machine learning for binary classification tasks. It's essentially a predictive analysis algorithm, based on the concept of probability, used for forecasting the possibility of a categorical dependent variable. The fundamental concept behind logistic regression is the logistic function, which is an S-shaped curve that maps any real-valued number onto a value between 0 and 1, in our case, asymptomatic and symptomatic. This characteristic makes it particularly suitable for binary classification tasks, where the goal is to predict one of two possible outcomes, often denoted as 0 or 1. Its importance in binary classification arises from its ability to provide probabilities and classify new samples using continuous and categorical input data. Moreover, logistic regression is highly interpretable, as it allows for the estimation of odds ratios for each independent variable, making it an indispensable tool in fields such as medical diagnostics.

The adjustment of weight for each independent variable in the logistic model can be optimized using gradient descent with a specific loss function, which is used to characterize the deviation of predicted value with ground truth. For binary classification tasks, a common choice for loss function is binary cross-entropy function ℓ_{BCE}. The training procedure of a logistic model with stochastic gradient descent is attached in appendix for reference.

5.2.4 Integration of Clinical History Data

Based on image features, the previous model can generate a prediction score. Besides, what we have in hand is the clinical history data associated to each patients, which leads to another model. The clinical history data consists of 9 fields, which consists of gender, age, living habits and several medical history columns.

For this stage, we use XGBoost (eXtreme Gradient Boosting) algorithm [9] to build a classification model based on clinical history data. It works by combining the predictive power of numerous weak decision trees in a sequential manner, with each tree constructed to correct the prediction errors made by the previous tree, leading to the 'boosting' process. In each iteration, a new tree is added to minimize the loss function, which measures the difference between the actual and predicted results. The 'gradient' in gradient boosting refers to the use of gradient descent optimization method to minimize this loss. What sets XGBoost apart is its scalability and efficiency due to its unique system design, especially when the dataset contains multiple types of data, which is suitable for our case.

In the presence of multiple subordinate models, ensemble learning can be applied to assemble a possibly stronger model. In the situation of our project, among three main ensemble methods: bagging, boosting and stacking, stacking is adopted to integrate two models, since bagging often requires large amount of data and boosting is prone to overfitting.

The core idea behind stacking is to combine the predictive power of several base models and use another machine learning model, known as the meta-model, to make a final prediction. This meta-level model is trained to make effective use of the predictions from the base models, which are used as its inputs, to generate a final prediction. The primary advantage of stacking is that it allows the strength of each individual model to be combined, potentially leading to better performance and increased generalizability compared to any single model. Also, it can capture the non-linear relationships among the predictions of the base models, improving accuracy and robustness.

The final prediction score will be in the range of 0.0-1.0, with higher score representing higher probability of having symptomatic plaque. For the sake of

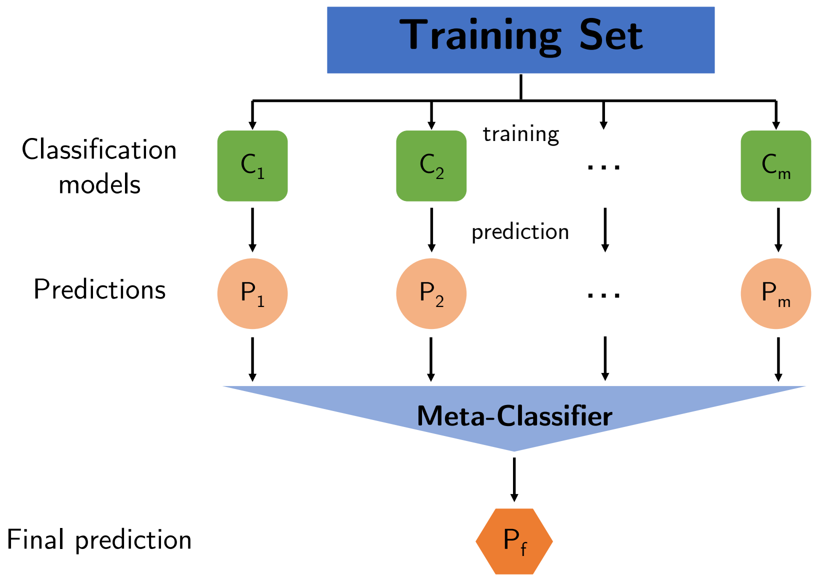


Fig. 5-22 Stacking

having a deterministic interpretation, a threshold of 0.5 was set to draw the boundary between asymptomatic and symptomatic plaques. Therefore, it's convenient for both doctors and patients to interpret the output of the model.

5.2.5 Graphical Interface

To enhance the user-friendliness of the design, we have incorporated a single-page web application for customers, which is shown in Fig. 5-23. The left half of the page is occupied by a form, where doctors/patients can effortlessly upload the ultrasound image and associated clinical history information to the server. Once uploaded, the front-end will wait until it receives the prediction score generated by back-end model. When the response is received successfully, the right half of the page will display the masked ultrasound image, where the position of the plaque will be highlighted with red color, and a brief diagnostic report will be generated in the right bottom block, which provides scores obtained from two sub-models (CNN and XGB) and the final stacking model. More importantly, a final prediction result on property of the carotid plaque will be displayed in different color, indicating symptomatic or asymptomatic. As claimed, this tool is only used for auxiliary diagnose, so it should not replace experienced professional doctor and radiologists. The aim of this application is to provide constructional insight to doctors' diagnostic procedure.

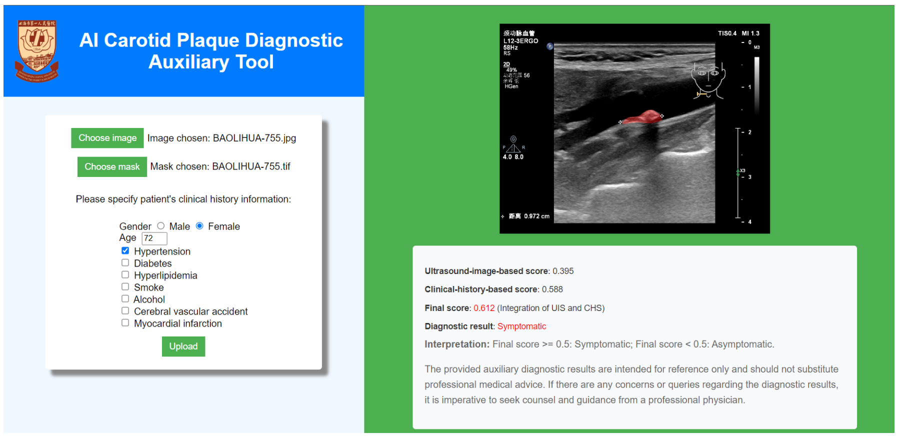


Image chosen: BAOLIHUA-755.jpg

Mask chosen: BAOLIHUA-755.tif

Please specify patient's clinical history information:

Gender Male [✓ Female]
Age 72
[✓ Hypertension]
Diabetes
Hyperlipidemia
Smoke
Alcohol
Cerebral vascular accident
Myocardial infarction

Ultrasound-image-based score: 0.395

Clinical-history-based score: 0.588

Final score: 0.612 (Integration of UIS and CHS)

Diagnostic result: Symptomatic

Interpretation: Final score >= 0.5: Symptomatic; Final score < 0.5: Asymptomatic.

The provided auxiliary diagnostic results are intended for reference only and should not substitute professional medical advice. If there are any concerns or queries regarding the diagnostic results, it is imperative to seek counsel and guidance from a professional physician.

Fig. 5-23 Graphical Interface

5.3 Implementing Plan

About our implementing plan, since our project is about programming without any physical assembling, we will mainly discuss about algorithms and implementations in this part.

About programming languages, we use Python because of its good flexibility and versatility and extensive library ecosystem in image processing field. Figure 5-24 is a chart showing our working flow. In the following we will explain algorithms in detail and list existed libraries we use stage by stage.

- **Augmentation**: In order to eliminate the noise of region outside the carotid plague, we invoke some functions in *OpenCV* library to crop the image and remove the irrelevant region as much as possible. The basic idea is finding the position of plague's contour first, then creating an external rectangle of plague and cropping eventually. Then we use contrast ratio enhancement to improve the visibility and clarity of the plague, which is also implemented based on *OpenCV* library.

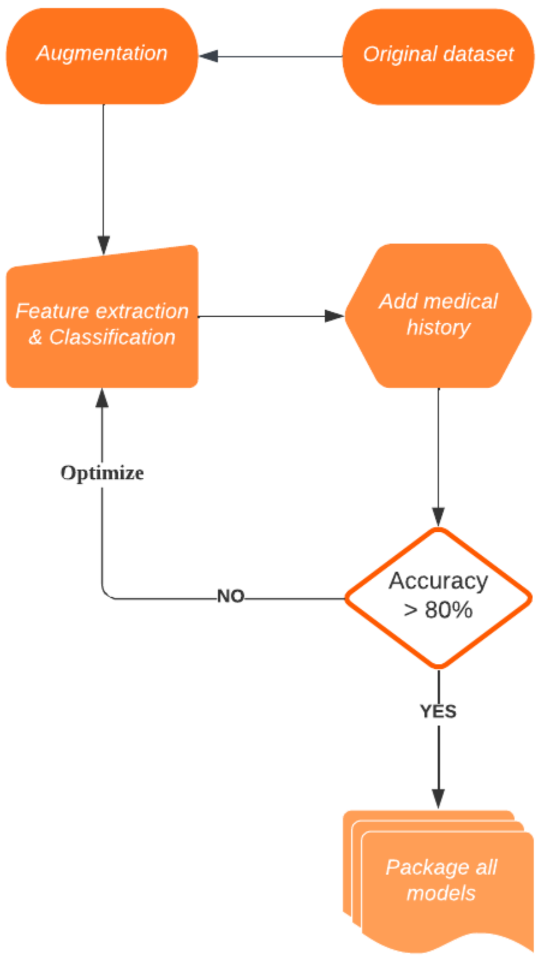


Fig. 5-24 Flow chart of the whole process

- **Feature extraction and classification**: Basically, we first introduce a pre-trained ResNet-18, then we replace its classification head so that it will generate a 10-dimension vector, which is an abstract representation for global feature of ultrasound image. As for local feature, we used K-Means for pixel value clustering. *scikit-learn* package is a powerful open source machine learning library for python, which provides many common algorithms used in machine learning field. We used the implementation of K-Means algorithm incorported in *scikit-learn*. Since the clustering result is unordered, which means the cluster center with higher pixel value is not guaranteed to appear at last. Therefore, we sort against the pixel value of total clusters, and generate the pixel proportion based on this adjusted order, which is shown in Fig. 5-25. Notice that the generated proportion (4-dimensional)

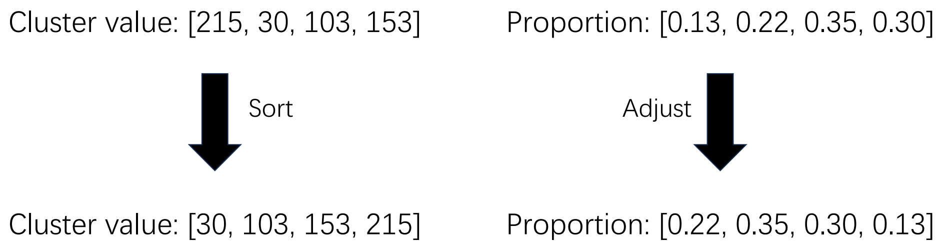

Cluster value: [215, 30, 103, 153]
Proportion: [0.13, 0.22, 0.35, 0.30]
Sort
Adjust
Cluster value: [30, 103, 153, 215]
Proportion: [0.22, 0.35, 0.30, 0.13]

Fig. 5-25 Sort clusters

is served as the local feature of the plaque. After the local feature is prepared, the input to the fusion model is now an image along with the local feature. The image is passed into ResNet-18, and generate a 10-dimensional vector, which is concatenated with the local feature, thus generating a 14-dimensional representation of the plaque. Finally, the representation will go through a logistic regression classifier to obatin the final prediction score.

We used *PyTorch* library, which is a popular open source library for deep learning in both industrial and academic field, to train this model.

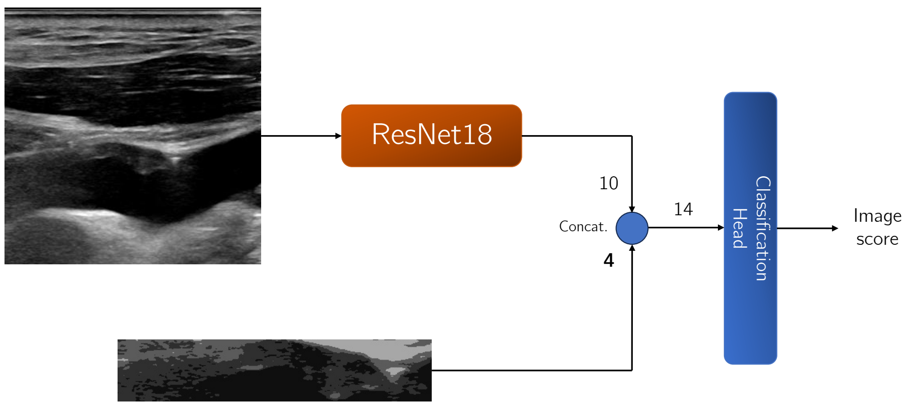


Fig. 5-26 Structure of the image classification model.

- **Integration of clinical history data**: In this part we introduce the clinical history information and use XGBoost classifier to obtain another prediction. An robust implementation of XGBoost can be found in Python *xgboost* library. Combined with the prediction obtained by the above ResNet-18 model and k-means clustering, we treat it as a new training set and feed them to a meta classifier to get the final prediction, which is the main idea of a ensemble classifier named stacking. Again, the meta-classifier is intrinsically a logistic regression which takes only a two-dimensional vector, which consists of scores from ultrasound image and clinical history data. We use the built-in logistic regression algorithm in *scikit-learn* library for meta-classifier training.
- **Implementation of web application**: In order to better put our model into the production environment and improve the user experience, we designed a simple web application for interaction. The application is based on request-

response model. In the world of computer networking, communication often occurs via the request-response model: when client (user) sends a request specifying its demand, the server will respond actively, based on the request. It's a fundamental part of how the internet works, and it's used in many different protocols and systems, such as HTTP (for web browsing), FTP (for file transfers), and many others. In our application, we used HTTP since HTTP is often faster for small files, and it's the most common protocol for web applications.

The technology stack used for this application is HTML5, CSS, JavaScript, and Python Flask framework. The basic function of them are:

- **HTML5**: Construct the basic layout of the web page.
- **CSS**: Define customized web-page pattern so that it looks more clear and elegant.
- **JavaScript**: Deal with several simple front-end service logic. Send asynchronous (non-blocking) request to back-end server.
- **Python Flask**: Specify the IP and port number of the application for user access. Process the request (picture and form data) sent by the front-end, and respond to the front-end.

The basic structure of our web application is shown in Fig. 5-27. The working procedure can be decomposed into 5 steps as shown in Fig 5-27:

1. **Request from user:** The client select the ultrasound image and mask image, as well as clinical data, indicated by several form control objects. When the "Upload" button is clicked, a POST request will be sent to the back-end server. Notice that the request is asynchronous, so that it will not block the front-end task. Taking advantage of the asynchronous request, a waiting animation based on CSS is designed. It is used to notice user your request has been received and is under processing.

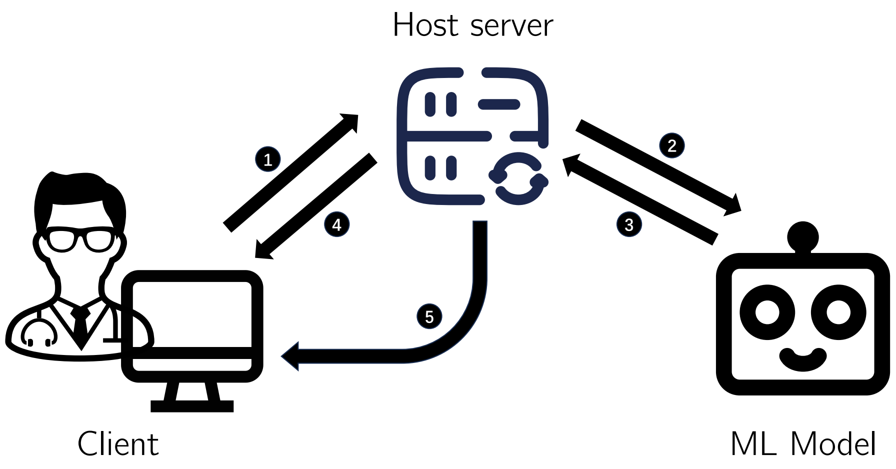


Fig. 5-27 Structure of web application.

2. **Command from server:** When back-end server receives the POST request sent by front-end, it preprocess the data collected and ready to communicate with the machine learning model on the server. The images will be dumped into a temporary folder on the server, and will then be cropped and clustered to generate the necessary input for machine learning models. Also, the form data will be converted into a one-hot vector.

3. **Model prediction:** We have 3 separate models on the server: CNN model, XGBoost model and logistic regression (stacking) model. Specifically, preprocessed image data will be fed into the CNN model, thus generating *cnn_score*. And the clinical one-hot vector will be the input of XGBoost model, generating *xgb_score*. Finally, the stacking model will take both *cnn_score* and *xgb_score*, generating the *final_score*. The prediction *result* will be generated accordingly, if *final_score* is greater or equal to 0.5, *result* will be "Symptomatic", otherwise it will be "Asymptomatic".

4. **Response from server:** After machine learning model finished its

job, the back-end server will package variables mentioned in the previous step into a JSON object for data transferring. Besides, the JSON object will include another field called *image_url*, which is the URL of the highlighted ultrasound image. The back-end server uses *OpenCV* library to locate the plaque position based on the mask, and increase the pixel value in its R channel and is then saved on the server. The URL path is assigned to *image_url*, then the JSON data is sent back to front-end for display. A user-friendly design noteworthy is that when the prediction result is symptomatic, front-end will render the font color as red (otherwise green). This aids those who have problem seeing things clearly, especially for the elders. Instead, they can get a feeling from colors, since striking colors such as red will serve as a warning to them.

5. **Exception handling:** Sometimes, the workflow cannot go successfully as expected, because exceptions may happen due to several reasons. In order to promote better user experience, the server tries to catch all possible exceptions as much as possible, and returns error message and HTTP status code to the front-end for display. Main sources of exceptions are:

   (a) Missing images from front-end.
   (b) Image IO exceptions (ultrasound image or mask image).
   (c) Missing required form data (age and gender).
   (d) Sophisticated exception during machine learning model inference.

   The error data will be received by front-end as the abnormal response to the asynchronous request. Accordingly, when receiving error data, a warning dialog box will be displayed in the center of the page, guiding users to do legal operations.

So far, the runtime of the application is based on the localhost server, which is only for development and test, rather than production. There are several web techniques for web applications to deploy on the public network, such as containers like Docker and Kubernetes, or cloud services like Amazon Web Services (AWS) and Microsoft Azure. However, these methods are usually not very easy to configure, and the vast majority of cloud services are not free. Since application deployment is not the focus of our project, we decide to implement an alternative plan: using Network Address Translation Traversal (NAT Traversal). Under normal circumstances, devices under private network cannot be visited by public network directly due to security concerns. In order to solve this problem, NAT Traversal is used to allow private network accessed by public.

NAT Traversal usually relies on an intermediate server to forward the request between 2 devices. The intermediate server is exposed in the public network. First, device A in the private network will establish a connection with the intermediate server, and its IP and port will be recorded. Later, when devices in the public network wants to communicate with device A, it will send a request to the intermediate server, and its request will be forwarded to device A. Similarly, the response from device A will be forwarded.

In our project, we use Ngrok, which is a free NAT Traversal service. We first launch the web application on localhost, and tell Ngrok the port number which our service is listening. Then Ngrok will provide us a public URL. Therefore, our service can be accessed by all users who know this URL.

About budget distribution, we spend total 2000 Yuan visiting hospital and patient and building and maintaining online platform, which are in the range of our initial budget.

5.4 Validation Results

The functionality of our design will validated in two main aspects: reliability and efficiency. The reliability is evaluated by common metrics used in classification tasks, including accuracy, precision, sensitivity, specificity, and AUC score. The general evaluation procedure is performed by comparing the predicted labels with ground truth labels. The efficiency will basically focus on the runtime efficiency of our deep learning model during inference in terms of both time cost and memory usage.

5.4.1 Reliability

To evaluate the reliability of our model, the initial dataset was randomly split into training set and test set in a ratio of 9 to 1. The training set will be used to train our model, while the test set will be made inference on to verify the reliability of our model. The detailed model training settings will be elaborated in the appendix.

The metrics used to evaluate our design is shown in Table 5-27 and Figure 5-28.

Set: #Images	Acc. (%)	Prec. (%)	Sens. (%)	Spec. (%)	AUC
Training set: 286	93.71	97.52	91.81	96.52	0.99
Test set: 36	80.56	95.00	76.00	90.91	0.84

Tab. 5-27 Evaluation metrics on both training set and test set.

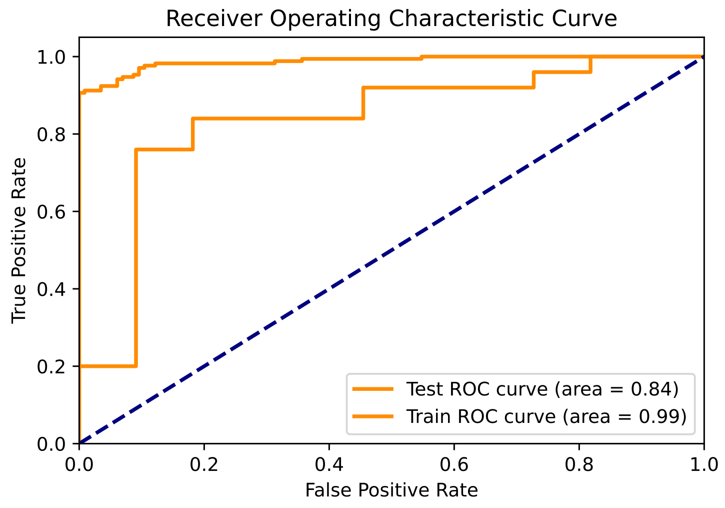


Fig. 5-28 ROC Curve and AUC score.

The expression of each metric can be found in the appendix.

As shown in Table 5-27, the test accuracy reaches 80.56%, which satisfies the basic requirement of our customer. Besides, the precision of our model is high, which means when our model gives a prediction that a plaque is malignant, there's high probability it's actually malignant.

5.4.2 Efficiency

As machine learning models grow increasingly complex, understanding their efficiency in varying operating environments becomes paramount. Evaluating the performance of our model is not a one-dimensional task; it necessitates an examination that reflects real-world scenarios and challenges. This section focuses on a comprehensive analysis of our model's efficiency, exploring how it behaves in both local and online environments. By rigorously testing the model under these distinct circumstances, we can ensure that it not only meets the desired performance metrics but also adapts seamlessly to different deployment settings. The insights gained from this testing procedure provide a multifaceted view of the model's capabilities.

- **Local Environment:**

  The most computationally intensive part in our design is the deep learning sub-model, since our designed model has 11181672 parameters. To ensure the model won't pose a huge challenge to computer's performance, we need to assess the time cost and memory usage during the inference stage.

  The machine used for test is equipped with NVIDIA GeForce RTX 3060 Laptop GPU (6GB), 12th Gen Intel(R) Core(TM) i7-12700H and 16GB RAM. The RAM usage during inference stage on test set is shown in Figure 5-29, which is measured with assistance of Python *psutil* library.

  The total inference time is only 0.8805 s, for 36 test images in total. Therefore, our model is efficient in both time and memory, and is applicable in a general production environment.

- **Online Platform:**

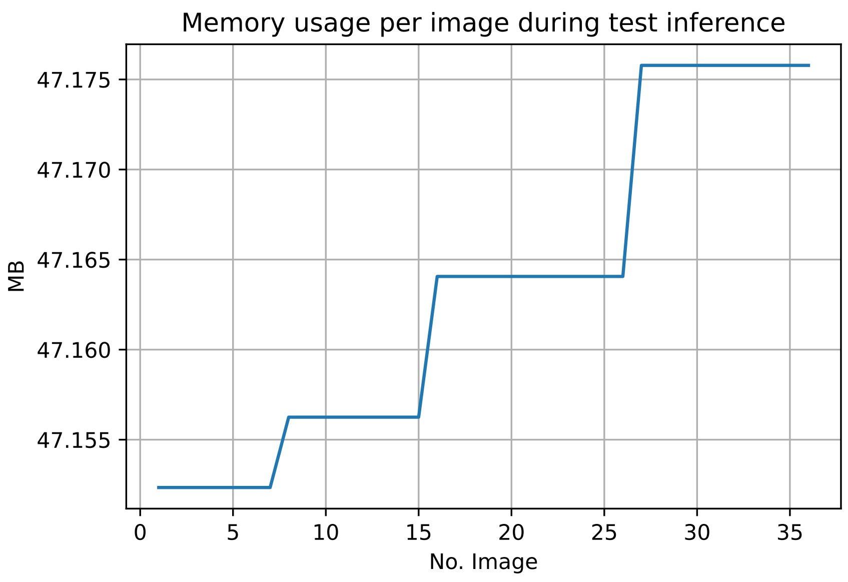


Fig. 5-29 Memory usage per image during test inference.

The validation of the efficiency of our machine learning model on the online platform required a methodical and rigorous approach. Recognizing that concurrent user requests are a common scenario in real-world applications, we aimed to simulate this environment to assess the model's responsiveness and stability. By issuing 10 concurrent requests and repeating this process 250 times, we were able to gauge how the model performed under sustained load, closely mimicking the behavior it might encounter in a production setting.

We paid particular attention to metrics such as the average response time, the distribution of response times, and the number of outlier responses that deviated significantly from the mean. These statistics offered a granular view of how the model coped with simultaneous requests, informing our understanding of its scalability and resilience.

The distribution plot and box plot of the response time are shown in Fig. 5-30 and Fig. 5-31.

The 95% confidence interval for the mean response time was calculated to be (1.024, 1.066) seconds, which is acceptable in most application scenar-

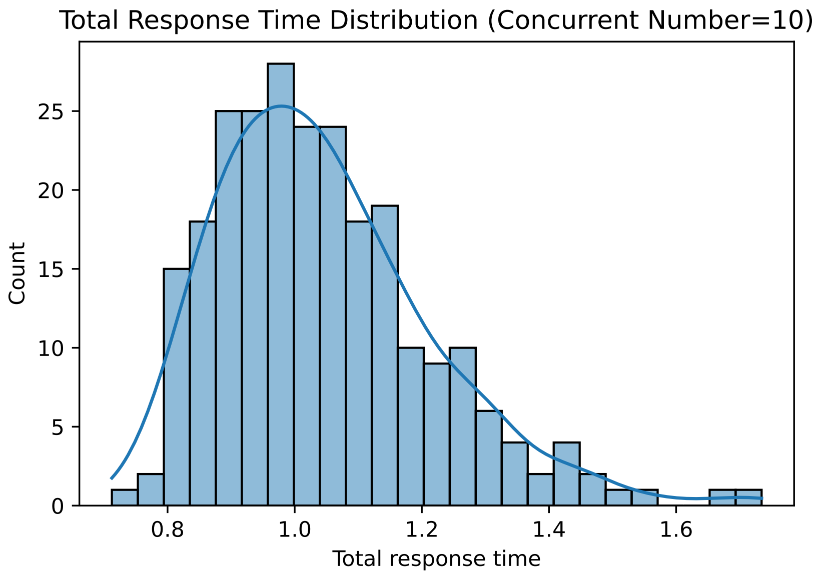


Fig. 5-30 Total response time distribution

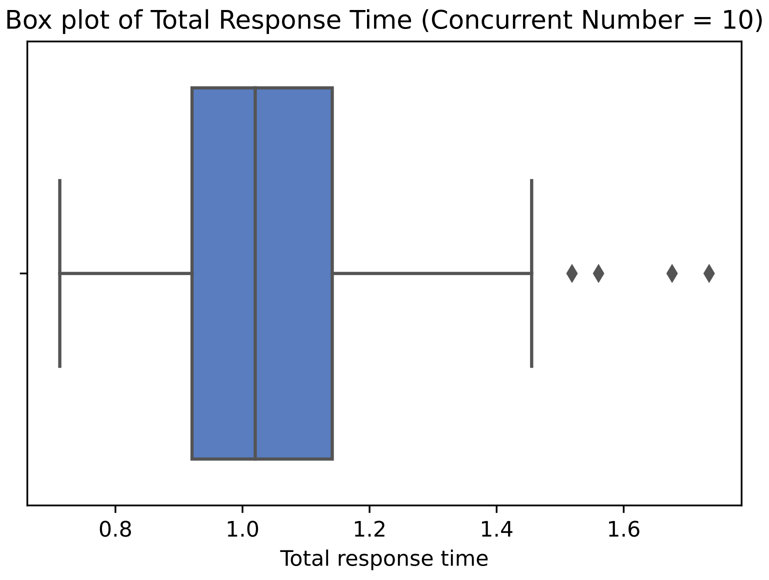


Fig. 5-31 Box plot of total response time

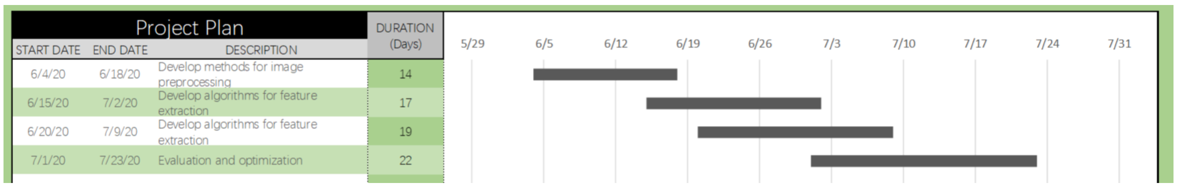
ios.

Chapter Six Project Plan

We divide our project into following steps: literature survey (June 1 - June 10), sample preparation (June 5 - June 13), trait characterization (June 11 - June 16), model prototype construction (June 13 - June 20), parameter adjustment (June 20 - July 5), dataset expansion (July 2 - July 7), verification and comparison (July 5 - July 10). In addition, we have multiple milestones: listed in Table 6-31. In addition, a Gantt chart is plotted to show our progress of completing the project, which is listed in Figure 6-32.

	Due date	Content
Milestone1	June 18	Develop methods for image preprocessing (denoising and segmentation)
Milestone2	July 2	Develop algorithms for feature extraction
Milestone3	July 9	Develop models for image classification
Milestone4	July 23	Evaluate the performance of models and optimize algorithms and model parameters

Tab. 6-31 Milestone arrangement



START DATE	END DATE	DESCRIPTION	DURATION (Days)
6/4/20	6/18/20	Develop methods for image preprocessing	14
6/15/20	7/2/20	Develop algorithms for feature extraction	17
6/20/20	7/9/20	Develop algorithms for feature extraction	19
7/1/20	7/23/20	Evaluation and optimization	22

Fig. 6-32 Gantt chart

And we distribute our workload in Table 6-32.

Finally, because the current data size (around 350), we don't need extra money to spend on computing resource. So our expense is mainly spent in terms of communication with hospital and patients to get access to urgent need in this area. And we will also try to build a platform to help doctors identify the malignant plague. Here's our budget plan: Table 6-32, 2000 Yuan in total.

Content	People in charge
Data preparation	Zicong Xiao and Zihao Xu
Model construction	Yu Sun, Yi Xia and Zeyu Zhang
Parameter tuning	Zicong Xiao and Zihao Xu

Tab. 6-32 Workload distribution

Content	Expected expense (Yuan)
Visit hospital and patient	1000
Build and maintain online platform	1000

Tab. 6-32 Budget arrangement

Chapter Seven Discussion

In this project, we eventually meet the 80% accuracy as we initially planed. However, during the development process there are still several aspects of the project that we could have improved. We think that our project plan needs further revision in terms of time allocation. We should also enhance the communication with the hospital for valuable medical advice on the interpretation of the ultrasound images we received.

To begin with, We spent a large proportion of time on data preprocessing and enhancement. The limited number of images we received meant that data augmentation is necessary if we want to reach a higher accuracy. However, our efforts did not significantly improve our accuracy and in some cases it even lowered the accuracy. Our highest accuracy is reached without much assistance of the augmented images we created. We believe that this may be related to the quality of the image. It is also possible that we fail to try enough data augmentation techniques to find the efficient way of removing the noise and extract the useful features. Performing more literature review on the possible data augmentation techniques in relevant fields before trying them may be an more effective method as trying some of these methods like the 3-Channel transform turned out to be time-consuming. Meanwhile, we think that the communication between us and the hospital was not sufficient. As none of our team members are experts of carotid plaques, professional knowledge is needed in this area to offer guidance on what might be the valuable features for classifying carotid plaques, which we can follow in the data augmentation stage.

In addition, after the feature fusion model trained on the ultrasound images was proposed, we improved the final accuracy of the model by more than 5 percent via hyper-parameter tuning. Due to the tight schedule we were unable to make further adjustments to these parameters. We think that we should allocate more time for this part during the planning stage.

The final test accuracy of our model exceeds 80%, which clearly shows the strong correlation between the carotid plaque and stoke symptoms. However, one notable weakness of our model is the limited size of data size it is trained on. Compared with latest researches on similar problems, the 322 images we received limits the robustness and the accuracy of our final model.

Chapter Eight Conclusion

To address the severe problem of locating and identifying malignant plague, we are going to design a software-based approach to extract the plague feature and ultimately automatic diagnose platform. According to the investigation, the major customer demands are accuracy of detecting plague, identifying symptomatic patients and robustness of program. Utilizing resnet18 and clustering, we manage to train a model that extracts both the global and the local features. Our model performs logistic regression to give a prediction based on the ultrasound images. In addition, an additional model is trained on the clinical history using XGBoost. The prediction of two models are combined using stacking to yield the final prediction. Our final model manages to reach 80% accuracy with a limited size of data set, which shows that ultrasound images of carotid plaque and the clinical history of the patients can be used to predict stoke symptoms. The requirements on memory and speed are also met based on our tests. Moreover, we design a graphical interface to make its application more user-friendly. In the future, the hospital will be able to collect more images to be used in training, which we believe could further improve its accuracy and robustness.

Chapter Nine Future Works

In this project, we have reached 80% accuracy of classification, which is satisfactory by now. However, there are latest GPT model such as BiomedGPT model that can reach approximately 90% accuracy[10]. It is achieved by training on thousands of images and a multi-modal model is presented. Although we don't have such massive database. We can still construct an online image submission platform for hospitals, which is expected to collect images over time.

Apart from that, given that the incompatibility of data enhancement, augmentation and our model, this could be due to multiple reasons:

- the quality of images
- the limitation of our model

We notice that the gray scale and contrast ratio is not eminent in our data set, improving the performance of image capture method in hospital may help to solve this problem.

In addition, we have tried some linear enhancement method such as cropping, flipping, rotation and some non-linear enhancement method such as heatmap, channel concatenation. These methods' effect are largely influenced by the image quality and some useless feature such as image edges. If we can adopt some method that is much more resistant to the low quality and maintains only important features, the accuracy may be raised. Wang and Gong proposed Keep Augment algorithm that is said to adopt "information-preserving strategy" that derives "more faithful training examples"[11]. See Fig. 9-33:KeepAugment improves existing data augmentation by always keeping the important regions (measured using saliency map) of the image untouched during augmentation. This is achieved by either avoiding to cut important regions (see KeepCutout), or pasting important regions on top of the transformed images

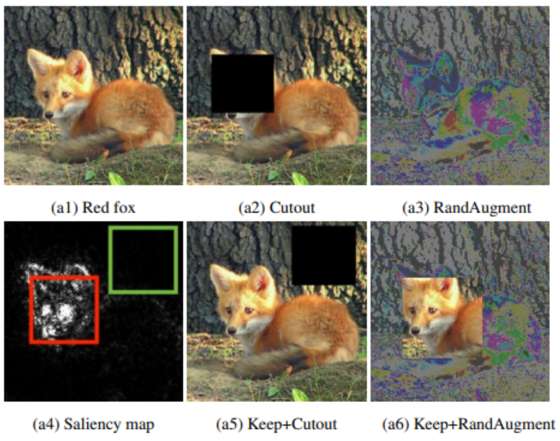


Fig. 9-33 Memory usage per image during test inference.

From the image processing aspect, a labeled and cropped data set may take month, because of cropping image manually is a considerably time-consuming work. If we can apply some auto-crop method to help doctors to process these images, it will save enormous human work. As clustering based image segmentation methods are prevailing methods, future works can be based on hierarchical and partitional based clustering methods, such as: Divisive clustering[12], fuzzy c-means (FCM) [13], and mountain method [14].

Above is our recommendations for future works, based on the problem we have encountered. As a rule of thumb, there can be subsequent problems emerging in the procedure of fixing these problems. We hope the after workers can patiently try these methods, and by applying more advanced algorithm, fulfilling more difficult criterion.

References

[1] Zhe Huang, Xue-Qing Cheng, Hong-Yun Liu, Xiao-Jun Bi, Ya-Ni Liu, Wen-Zhi Lv, Li Xiong, and You-Bin Deng. Relation of carotid plaque features detected with ultrasonography-based radiomics to clinical symptoms. *Translational Stroke Research*, 13, 11 2021.

[2] Spyretta Golemati, Amalia Yanni, Nikos N. Tsiaparas, Symeon Lechareas, Ioannis Vlachos, Demosthenes Cokkinos, Miltiadis Krokidis, Konstantina Nikita, Despina Perrea, and Achilles Chatziioannou. Curvelettransform − based texture analysis of carotid b-mode ultrasound images in asymptomatic men with moderate and severe stenoses: A preliminary clinical study. *Ultrasound in Medicine Biology*, 48, 10 2021.

[3] Xiao-Wei Huang. Assessment of the risk of atherosclerotic plaques base on ultrasound images. Master's thesis, Southern Medical University, 2015.

[4] Meng-Miao Xu. Detect neovascularization of carotid plaque on unenhanced mri by texture features. Master's thesis, Jiangsu University, 2020.

[5] Yong-Kang Luo. Carotid plaque texture feature extraction in 3-d ultrasound images and its application to atorvastatin evaluation. Master's thesis, Huazhong University of Science and Technology, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] X. Xu, L. Huang, R. Wu, W. Zhang, G. Ding, L. Liu, M. Chi, and J. Xie. Multi-feature fusion method for identifying carotid artery vulnerable plaque. *IRBM*, 43(4):272–278, 2022.

[8] E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

[9] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016.

[10] Kai Zhang, Jun Yu, Zhiling Yan, Yixin Liu, Eashan Adhikarla, Sunyang Fu, Xun Chen, Chen Chen, Yuyin Zhou, Xiang Li, Lifang He, Brian D. Davison, Quanzheng Li, Yong Chen, Hongfang Liu, and Lichao Sun. Biomedgpt: A unified and generalist biomedical generative pre-trained transformer for vision, language, and multimodal tasks, 2023.

[11] Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Keepaugment: A simple information-preserving data augmentation approach, 2020.

[12] Marie Chavent, Yves Lechevallier, and Olivier Briant. Divclus-t: A monothetic divisive hierarchical clustering method. *Computational Statistics Data Analysis*, 52(2):687–701, 2007.

[13] James Bezdek. Cluster validity with fuzzy sets. *Journal of Cybernetics*, 3, 07 1973.

[14] R.R. Yager and D.P. Filev. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1279–1284, 1994.

Symbols and Marks Appendix 1

TP (True Positive): Symptomatic people correctly identified as symptomatic
FP (False Positive): Asymptomatic people incorrectly identified as symptomatic
TN (True Negative): Asymptomatic people correctly identified as asymptomatic
FN (False Negative): Symptomatic people incorrectly identified as asymptomatic

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{9.1}$$

$$Precision = \frac{TP}{TP+FP} \tag{9.2}$$

$$Sensitivity(\text{also } TPR) = \frac{TP}{TP+FN} \tag{9.3}$$

$$Specificity = \frac{TN}{TN+FP} \tag{9.4}$$

$$FPR = \frac{FP}{FP+TN} \tag{9.5}$$

Pseudocode of Algorithms Appendix 2

Pseudo-code of K-means Algorithm

Algorithm 1 K-means algorithm

```
1:  procedure Kmeans($X, K$)
2:      Initialize $K$ centroids $C = \{c_1, c_2, ..., c_K\}$
3:      while centroids $C$ change do
4:          for each point $x_i$ in $X$ do
5:              $d(x_i, c_j) \leftarrow$ distance between $x_i$ and $c_j$
6:              $c^*(x_i) \leftarrow \arg\min_{c_j \in C} d(x_i, c_j)$ ▷ Assign point $x_i$ to nearest centroid
7:          end for
8:          for each centroid $c_j$ in $C$ do
9:              $c_j \leftarrow \frac{1}{|c^*(c_j)|} \sum_{x_i \in c^*(c_j)} x_i$ ▷ Update centroid $c_j$ to mean of assigned points
10:         end for
11:     end while
12:     return clusters $c^*(x_i)$ for all $x_i$
13: end procedure
```

Logistic Regression

Binary Cross-Entropy Loss Function

$$\begin{aligned}\ell_{BCE}(y_i, \hat{y}_i) &= -y_i \log(\hat{p}_i) - (1 - y_i) \log(1 - \hat{p}_i) \\ &= -y_i \log(\sigma(\theta^T x_i) - (1 - y_i) \log(1 - \sigma(\theta^T x_i)) \\ \frac{\partial}{\partial \theta_j} \ell_{BCE}(y_i, \hat{y}_i) &= (-y_i + \sigma(\theta^T x_i)) x_{i,j}\end{aligned}$$

y_i is the ground truth label; $\hat{y}_i$ is the predicted label; $\hat{p}_i$ is the logit; x_i is the data-point; $\sigma(\cdot)$ is the sigmoid function; θ is the weight vector for input data-point.

Logistic Regression with SGD

Algorithm 2 Logistic Regression with Stochastic Gradient Descent

```
1: procedure LogisticRegressionSGD($X, y, epochs, \eta$)
2:     Initialize weights $\theta$ with zeros or small random numbers
3:     for $epoch \in 1$ to $epochs$ do
4:         Randomly shuffle elements in $X$
5:         for each $(x_i, y_i)$ in $(X, y)$ do
6:             $z_i \leftarrow \theta \cdot x_i$                                  ▷ Compute weighted input
7:             $\hat{y}_i \leftarrow \frac{1}{1+e^{-z_i}}$                         ▷ Apply sigmoid function
8:             $error_i \leftarrow \frac{\partial}{\partial\theta}\ell_{BCE}(y_i, \hat{y}_i)$       ▷ Compute error
9:             $\theta \leftarrow \theta - \eta \cdot error_i$                     ▷ Update weights
10:        end for
11:    end for
12:    return weights $\theta$
13: end procedure
```

Training Settings

The dataset (322 images) is divided into 286 training images and 36 test images. The input raw images are resized to 224×224 and then normalized. The optimizer used for training is ADAM with a learning rate = 0.001. Later, the model is trained with a batch size of 20, with total epoch of 26. The loss and accuracy plots are shown in Fig. 9-34 and Fig. 9-35.

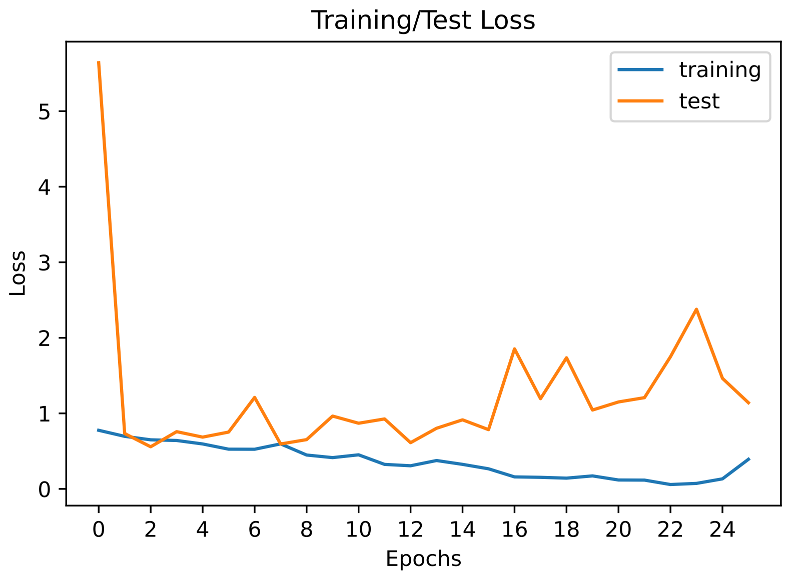


Fig. 9-34 Training/Test Loss.

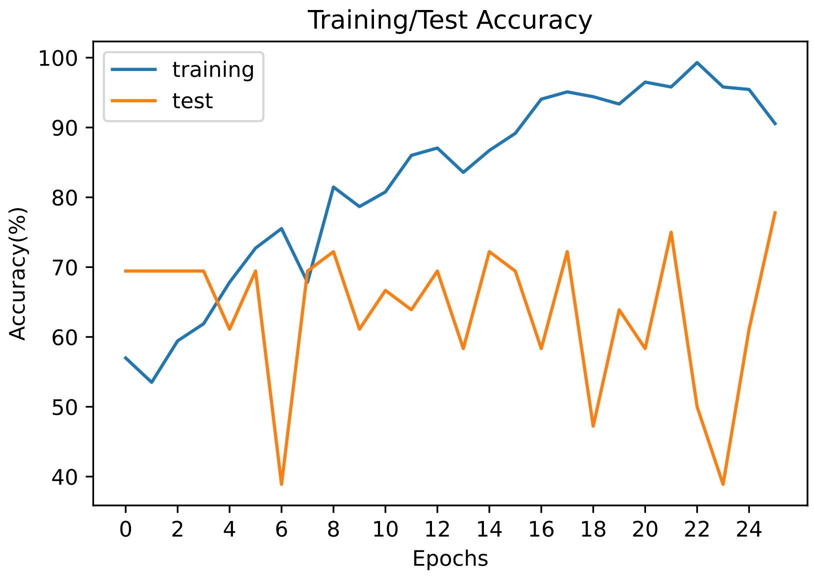


Fig. 9-35 Training/Test Accuracy.

Engineering Changes Notice

- **WAS**:

  In the image preprocessing stage, we first crop the image set based on the contour of carotid plague in order to eliminate the interruptions of other uninterested regions. Then we augment images by random linear transform, including horizontal flip, vertical flip, random rotation, random affine transformation, and sharpness adjustment. We triple our dateset based on these transform methods. Besides, we try contrast ratio enhancement and 3-channel transform to help the feature extraction process.

- **IS**:

  After testing, we find that although we augment images expand dataset size using random linear transform, the final classification accuracy does not improve compared with simply cropping images and then splitting them into training and test set. One possible reason may be that the random linear transform process generates some chaotic and useless features, which can not be learnt by deep learning model in the training process. In addition, the contrast ratio enhancement is able to improve the performance a little bit while 3-channel transform does not make obvious help in classification task. Therefore, in our actual prototyped part we remain the process of contrast ratio enhancement and abandon random linear transform, dateset expansion and 3-channel transform part.

Acknowledgements

We’d like to acknowledge the assistance from Prof. Sung-Liang Chen and Prof. Peisen Huang, UM-SJTU Joint Institute, in guiding us to complete the design on right track. We’d also like to extend gratitude to Dr. Caixia Jia, SHG, who provides the image set.