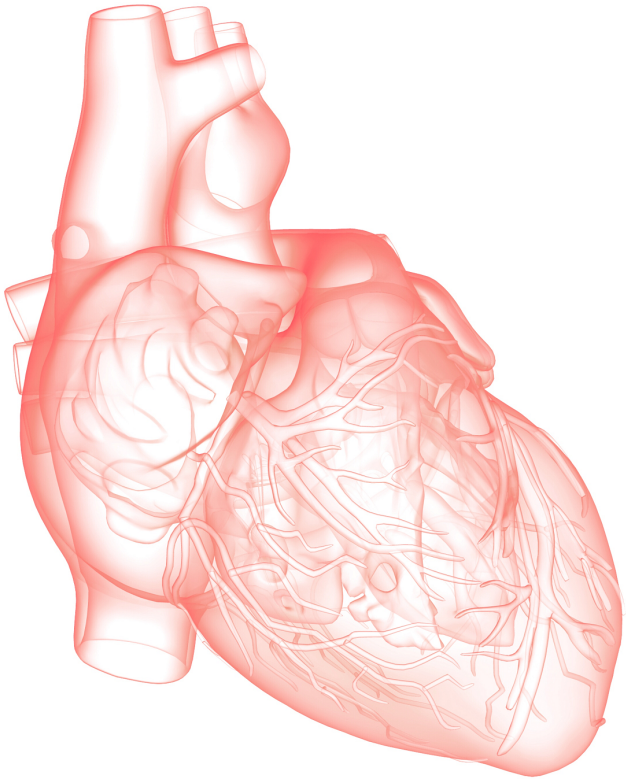

Pericardium Segmentation in Non-Contrast Cardiac CT Images Using Convolutional Neural Networks

Master's thesis in Master Programme Biomedical Engineering

BOLIN SHAO

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016:EX069/2016

Pericardium Segmentation in Non-Contrast Cardiac CT Images Using Convolutional Neural Networks

BOLIN SHAO

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2016

Pericardium Segmentation in Non-Contrast Cardiac CT Images Using Convolutional Neural Networks
BOLIN SHAO

Supervisor: Jennifer Alvén, Signals and Systems
Examiner: Fredrik Kahl, Signals and Systems

Master's Thesis 2016:EX069/2016
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46-(0)31 772 1000

Cover: Schematic diagram of a human heart

Typeset in LaTeX
Gothenburg, Sweden 2016

Pericardium Segmentation in Non-Contrast Cardiac CT Images Using Convolutional Neural Networks
BOLIN SHAO
Department of Signals and Systems
Chalmers University of Technology

Abstract

Recent studies show that the epicardial fat volume is an important indicator for many cardiovascular diseases, such as coronary atherosclerosis. The epicardial fat is the visceral fat that is located between the heart and the pericardium. The pericardium structure, i.e. the heart sack, is a thin layer that covers the heart and is barely visible in cardiac CT images.

This thesis proposes a method for automatic pericardium segmentation in non-contrast 3D CT images. The thesis can be divided into two main parts. The first part focuses on pericardium ground truth creation. Given a set of labelled contrast CT images, image registration methods are used to generate an estimated pericardium ground truth for unlabeled non-contrast CT images needed in the second part. Given this estimated labelling, the second part of the thesis concentrates on pericardium segmentation using machine learning algorithms. Convolutional neural networks (CNNs) are trained to classify the voxels in the non-contrast CT images. Multi-atlas techniques are combined with the generated probability map in order to define the regions of interest. Graph cuts is applied to obtain the final segmentation.

The results present an average dice coefficient larger than 0.95 on the test images, a comparable number to similar algorithms for contrast CT images.

Keywords: image segmentation, image registration, pericardium, machine learning, convolutional neural network (CNN), non-contrast CT, SCAPIS, epicardial fat.

Acknowledgements

This thesis would not be possible without the help from my supervisor Jennifer Alvén, who was extremely patient with all my questions and who kindly allowed me to use her computer to do parts of my heavy computations. Thanks to David Molnar, who carefully examined the results even when he was on parental leave. Thanks to Måns Larsson, who helped me understand how to start with the neural networks. Thanks to Courtney Keeler, who thoroughly checked my report and offered me advices on writing in English. Also, special thanks to my examiner Fredrik Kahl, who generously offered his office to me during my entire thesis.

Bolin Shao, Gothenburg, August 2016

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Epicardial fat is a visceral fat deposit, located between the heart and the pericardium, a double layered sac covering the heart. Recent studies have shown that the epicardial fat volume may be an indicator of many cardiovascular diseases, such as coronary atherosclerosis [29]. This requires an accurate pericardium segmentation, i.e. delineation of the pericardium in the image, as fat tissues are located both inside and outside pericardium.

The widely used computed tomography (CT) images can be categorized into contrast CT and non-contrast CT. Although the quality of contrast CT images are in general better than non-contrast CT images, non-contrast CT images are much easier to acquire and no intrusive injection is required. The injection of contrast agent can sometimes cause adverse effects. Reactions vary from minor to severe, in some cases even death. Also the non-contrast CT requires less dose and has no nephrotoxicity. In general, non-contrast CT is still safer than contrast CT. In addition, for historical reasons, non-contrast CT is still widely used in some countries, for example the US, as a way of scoring calcium content in the vessels [30].

1.2 Aim

Conventionally, the pericardium segmentation can be done manually by professionals. However, the workload of manually delineating pericardium is huge, about 10 hours for one patient. Thus, an automated method is demanded. This thesis focuses on automatic pericardium segmentation in non-contrast CT images.

1.3 Data

The data used in this project comes from the Swedish CArdioPulmonary bioImage Study (SCAPIS), a nationwide project, which collects medical images of people aged 50 to 64 years old [31]. The data includes 27 contrast CT and the corresponding non-contrast CT images, together with manually delineated pericardium segmentations in the contrast CT images. The manual delineations were created by professionals in previous projects. Thus there are no manual delineations for the non-contrast CT images.

The contrast images in this project are of sizes varying from $512 \times 512 \times 342$ to $512 \times 512 \times 458$ voxels with a voxel dimension of $0.3320\text{mm} \times 0.3320\text{mm} \times 0.3000\text{mm}$ and the non-contrast images are of sizes from $512 \times 512 \times 65$ to $512 \times 512 \times 94$ voxels with a voxel dimension of $0.3320\text{mm} \times 0.3320\text{mm} \times 1.5000\text{mm}$, a significantly larger scale in the third direction.

Figure 1.1 shows an example of one slice of a 3D contrast volume image, the manually delineated segmentation of the contrast image and non-contrast volume image. It can be observed that the pericardium structure is barely visible as a thin line in the contrast image, and even fainter in the non-contrast image.

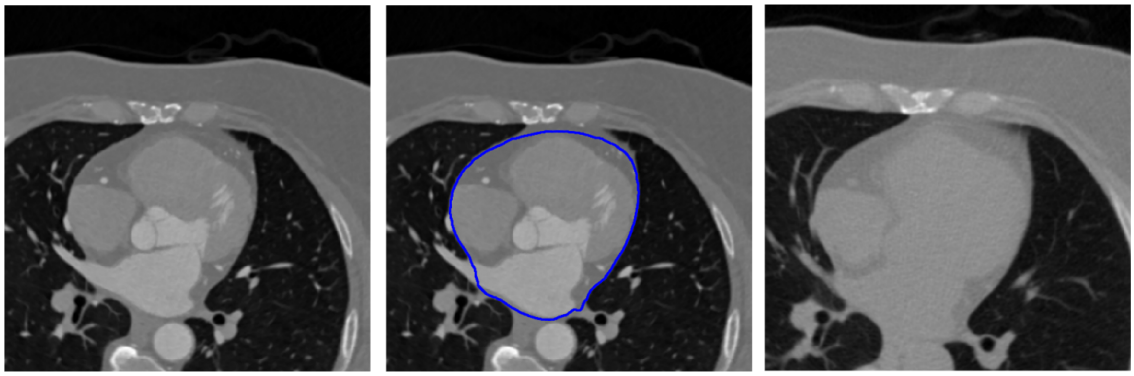

Figure 1.1: An example of contrast and non-contrast CT images. Left: contrast CT. Middle: contrast CT with pericardium segmentation. Right: non-contrast CT.

1.4 Our approach

To achieve the aim, machine learning algorithms have great advantages. The image segmentation task is indeed a classification problem, i.e. to tell if the voxel belongs to the foreground or the background, which is exactly what neural networks excel at. Given recent studies [28], a convolutional neural network will be trained to distinguish pericardium from other structures. Post processing techniques, such as multi-atlas and graph cuts, are used to refine the results. However, ground truth has to be acquired prior to the training.

In a common CT image acquisition procedure, the non-contrast image will first be taken, and then the contrast agent will be injected to the patient. After 10 to 15 minutes, the contrast image can be taken. Thus, the contrast and non-contrast images not only differ from intensities as Figure 1.1 shows, but also in placement. To achieve accurate segmentations, image registration methods are used to register the contrast images to the non-contrast images. In this thesis, both feature-based and intensity-based registration methods were applied.

In summary, this thesis consists of two major parts. The first part deals with the missing manual segmentation for the non-contrast images. The second part uses the segmentation results from the previous parts to train a convolutional neural network in order to automatically segment unseen non-contrast CT images.

1.5 Related work

Previous work by Alexander Norlen et al. [22] presents an efficient approach to automatically segment pericardium and compute epicardial fat volume in contrast CT images. The method used random forest as a classifier and achieved a dice score of around 0.95.

Similar medical image segmentation work is done by Måns Larsson et al. [32]. In the paper, a convolutional neural network is trained to segment different organs in one contrast CT image, such as spleen, kidney, gallbladder and so on. The average dice score achieved is around 0.76, with a highest dice score of 0.93 of spleen and a lowest dice score of 0.57 of portal vein and splenic vein.

2

Theory

2.1 Fat detection

In a conventional CT scan machine, the image is usually generated by measuring the intensity of a penetrating x-ray beam. Figure 2.1 sketches the principles of image acquisition in a typical fourth generation CT machine. The x-ray beam is generated from the x-ray tube and then transmitted through the human body. The detectors are illuminated by the attenuated x-ray beam, whose intensity usually varies with locations, since the human body contains various tissues with different absorption abilities.

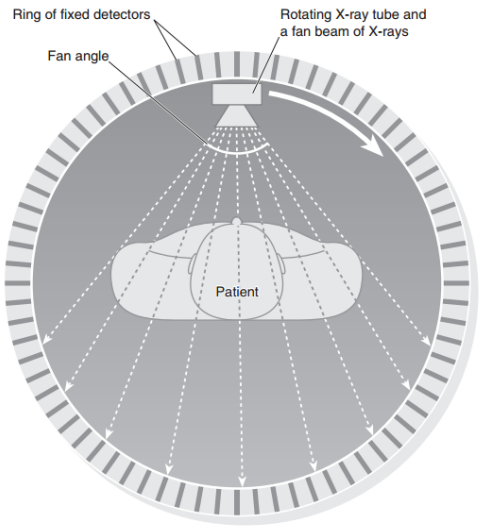

Figure 2.1: An illustration of a CT scan machine. [1]

The intensities assigned by the detectors are then compared to the attenuation value of water and displayed as Hounsfield Units (named after Sir Godfrey Hounsfield). In practice, the Hounsfield scales of different tissues are not the same for different CT scanners, but most of them lie in a relatively fixed range, shown in Figure 2.2. As illustrated in the figure, a typical Hounsfield scale is from -1000HU to +1000HU, which stands for a shade of grey from black to white. In clinical applications, the Hounsfield Scale intervals of a tissue can usually be observed from the intensity

histogram of the CT images [2].

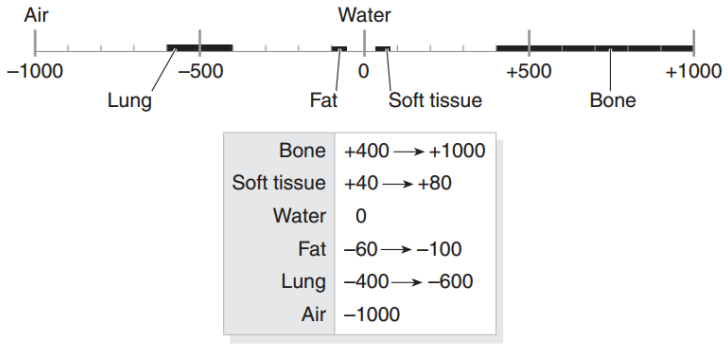

Figure 2.2: The Hounsfield Scale of CT numbers [2].

2.2 Image registration

Image registration or image alignment is to align multiple images of the same object, so that images taken under different conditions, such as different perspectives, different sensors or different time, can be compared. The correspondences between images can usually be represented by geometric transformations. Applying a geometric transformation will map points and areas in one image to another image. Examples of applications are to align medical images captured by different modalities (MRI, CT or SPECT), or to compare images taken at different times to discover the variation of a tumor.

The methods of image registration can be generally classified into two families: feature-based methods and intensity-based methods. In feature-based methods, correspondences are found between image features, such as the SIFT feature points [6], while in intensity-based methods, intensity patterns of the entire image or subimages are compared.

In image registration, it is often the case that one image is the target image while the rest of the images are the source images. The goal of image registration is now to calculate the most appropriate transformation for each source image that warps the source image onto the target image.

In Figure 2.3, the MRI image slice $u(x)$ is the target image and the CT image $v_0(x)$ is the source image. The warped CT image, $v(x)$, is an ideal image perfectly aligned with the target image $u(x)$. An unknown transformation matrix relates the warped image $v(x)$ to the source image $v_0(x)$ by

$$v_0(x) = v(\boldsymbol{T}^*(x)). \tag{2.1}$$

The goal of the registration process is to estimate a matrix $\tilde{\boldsymbol{T}}$ that is close to $(\boldsymbol{T}^*)^{-1}$.

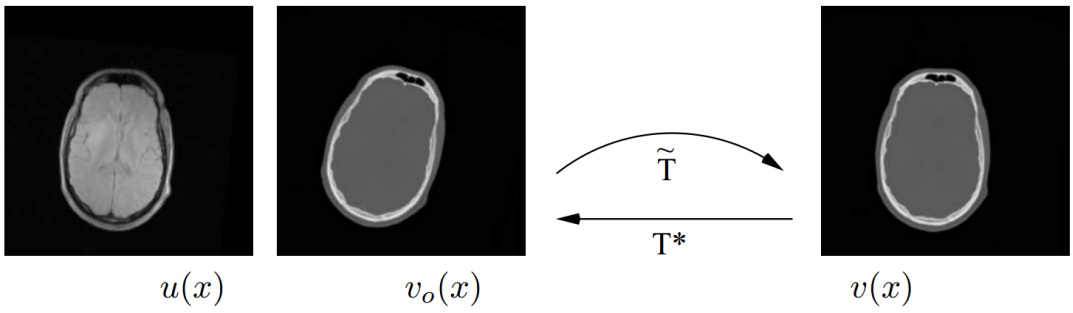

Figure 2.3: An example of 2D registration [4].

2.2.1 Transformation model

Geometric transformation models determine how one image transforms into another one. Two commonly used models are affine transformations and non-rigid transformations. An affine transformation can be formed by four basic transformations: translation, scaling, rotation and shearing.

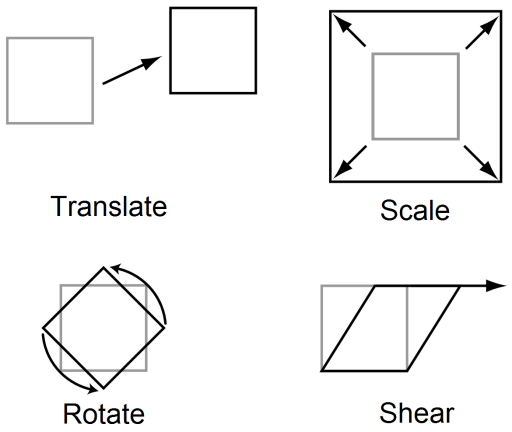

Figure 2.4: Basic affine operations [3].

Consider a three-dimensional Cartesian coordinate system, whose basis vectors are e_i, e_j and e_k. A translation transformation simply moves the origin of the coordinate system along a certain vector, i.e.

$$\begin{pmatrix} \tilde{e}_i \\ \tilde{e}_j \\ \tilde{e}_k \end{pmatrix} = \begin{pmatrix} e_i \\ e_j \\ e_k \end{pmatrix} + \begin{pmatrix} t_i \\ t_j \\ t_k \end{pmatrix}. \tag{2.2}$$

In order to represent all kinds of transformation by matrices, especially for the nonlinear translation transformation, homogeneous coordinates are used. The above translation can be formed as

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & t_i \\ 0 & 1 & 0 & t_j \\ 0 & 0 & 1 & t_k \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

and thus, the translation operation can be rewritten as

$$(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k, 1)^T = \boldsymbol{T}(e_i, e_j, e_k, 1)^T. \tag{2.3}$$

A slightly more complex transformation model is the rotation transformation. A rotation transformation in 3D space can be decomposed into three 2D rotations along the basis vectors e_i, e_j and e_k. The 2D rotations with an angle θ about the three axes are given by

$$\boldsymbol{R}_i = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\boldsymbol{R}_j = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\boldsymbol{R}_k = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

The corresponding 3D rotation operation can be written as

$$(\tilde{e}_i, \tilde{e}_j, \tilde{e}_k, 1)^T = \boldsymbol{R}_k \boldsymbol{R}_j \boldsymbol{R}_i (e_i, e_j, e_k, 1)^T. \tag{2.4}$$

Rotation and translation transformation retains the scale of the basis vector, while the scaling operation allows different scaling factors in different directions. The scaling matrix is given by

$$\boldsymbol{S} = \begin{pmatrix} s_i & 0 & 0 & 0 \\ 0 & s_j & 0 & 0 \\ 0 & 0 & s_k & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Finally, shearing offsets a set of points a distance proportional to the coordinates

$$\boldsymbol{H} = \begin{pmatrix} 1 & h_{xy} & h_{xz} & 0 \\ h_{yx} & 1 & h_{yz} & 0 \\ h_{zx} & h_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The affine transformation is the combination of these four basic operations. There-

fore, an affine transformation can be written as

$$\begin{pmatrix} \tilde{e}_i \\ \tilde{e}_j \\ \tilde{e}_k \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & t_i \\ d & e & f & t_j \\ g & h & i & t_k \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} e_i \\ e_j \\ e_k \\ 1 \end{pmatrix}. \tag{2.5}$$

Often, in medical imaging applications, mere affine transformations are not adequate to describe all kinds of naturally occurring transformations. Non-rigid transformation on the other hand allows an arbitrary way to align one image to another. One way to represent a non-rigid transformation is to use a deformation field, which stores all corresponding coordinates between the source image and the target image. Clearly, the more detailed the non-rigid transformation is, the more computational power it requires. In medical imaging, a B-spline based, free-form deformation model is usually used to balance the trade-off [26].

2.2.2 Feature-based image registration

Given a set of points $\boldsymbol{P}_i(i = 1, 2...n)$ in the source image and the corresponding matched points in the target image $\boldsymbol{Q}_i(i = 1, 2...n)$, it is possible to calculate a transformation matrix that maps $\boldsymbol{P}_i$ to $\boldsymbol{Q}_i$. To achieve this, it is apparent that n should be at least four, since a general affine transformation has 12 degree of freedom. For cases with n larger than four, different affine transformation matrices can be calculated through different four-point subsets of $\boldsymbol{P}_i$ and $\boldsymbol{Q}_i$. Among all possible transformation matrices, some are better than others. Thus, the goal now is to select the best one, which is equivalent to a model fitting problem.

The model fitting problem can be furthermore concluded as an optimization problem once an evaluation criterion, usually a cost function, is set. A simplified 2D example is sketched in Figure 2.5. Stars and circles are used to mark the matched points in the sources and target image respectively. Three pairs of points are selected to calculate a 2D affine transformation matrix. By applying the transformation matrix $\boldsymbol{T}$ on the source image, a warped image is obtained, shown as the tilted one on the right. Comparing the target image and the warped image, it is clear that some points align almost perfectly with their matched points, such as the red pair with a small distance between the points. Other points are far away from their corresponding points, for example the green pair. It is also reasonable to infer that a transformation will be the best one if as many pairs as possible are well aligned.

Thus, it is straightforward to define a cost function as the number of outliers, which is also called l^0 norm. Let θ be a threshold such that a pair of points with a distance smaller than the threshold are considered inliers, otherwise outliers. The distance between the warped points and the corresponding points in the target images is given by

$$r_i(\boldsymbol{T}) = \|\boldsymbol{P}_i\boldsymbol{T} - \boldsymbol{Q}_i\|. \tag{2.6}$$

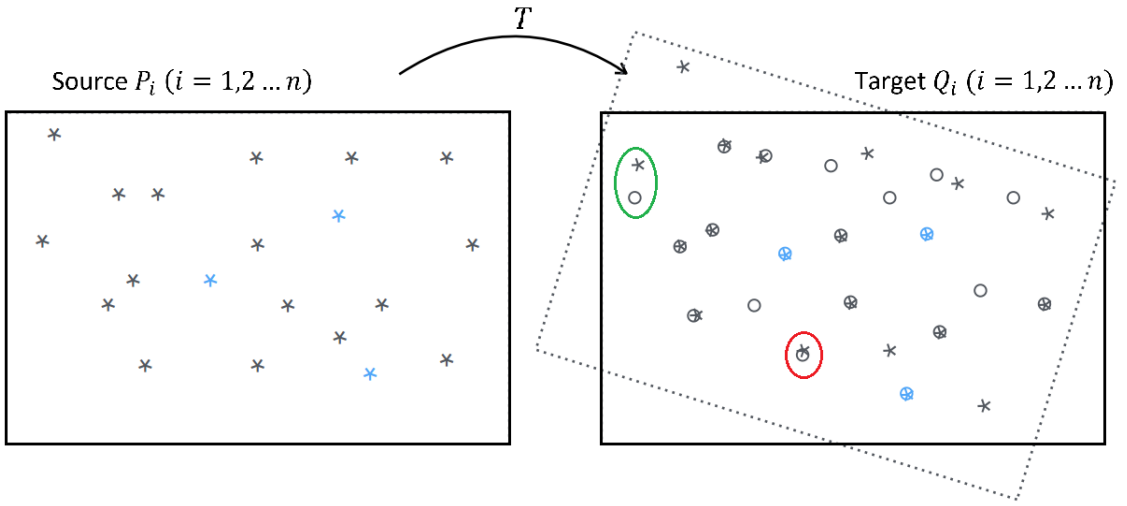

Figure 2.5: An example of a transformation matrix estimation. The source image is on the left and the target image on the right. The matrix is estimated through the light blue pairs. The result circled by red is an almost perfect alignment, while the green one is poorly aligned.

Therefore, a cost function based on the l^0 norm can be defined as

$$L = \sum_{r_i(\boldsymbol{T})>\theta} 1. \tag{2.7}$$

Another approach to define the cost function is to use l^1 norm instead, i.e.

$$L = \sum_i r_i(\boldsymbol{T}). \tag{2.8}$$

In this approach, the cost function is the sum of residuals, which punishes the pairs far away from each other.

A similar approach to the l^1 norm is to use l^2 norm, i.e. the sum of squared distances

$$L = \sum_i r_i^2(\boldsymbol{T}). \tag{2.9}$$

A modification of the l^2 norm is known as truncated l^2 norm, which aims at preventing the system from being dominated by a few large distances.

$$L = \sum_i \min(r_i^2(\boldsymbol{T}), \theta^2). \tag{2.10}$$

As discussed above, the desired transformation matrix is achieved when the following is true

$$\boldsymbol{T}^* = \arg\min_{\boldsymbol{T}} L. \tag{2.11}$$

Algorithm 1 RANSAC algorithm

1: Randomly select a minimum subset of data that is large enough to estimate the model
2: Construct the model using the subset
3: Evaluate the model by a certain cost function
4: Keep the model if the cost is minimal so far
5: Repeat step 1 to 4 until a certain number of iterations is reached or cost is lower than some threshold

In medical imaging, this problem can usually be well solved by an iterative method named random sample consensus (RANSAC) [5]. A general RANSAC program works as Algorithm 1 describes.

The remaining task is to find reliable matched points. This can be achieved by comparing features in the source and target images. A gradient based algorithm called scale-invariant feature transform (SIFT), proposed by David G. Lowe [6], detects and describes feature points in the images. A point will be considered as a feature point only when it is an extrema point once a difference of Gaussian function has been applied to the image. The feature itself is described by a 128-dimensional vector.

The correspondences between the feature points in the source image and target image are constructed by calculating the distance between the 128-dimensional vectors and thereafter applying the following matching criterion proposed by Lowe:

Let $\boldsymbol{a}_i(i = 1, 2..m)$ be the descriptors of the source image, $\boldsymbol{b}_j(j = 1, 2..n)$ be the descriptors of the target image, then $d_{ij} = \|\boldsymbol{a}_i - \boldsymbol{b}_j\|$ is the distance between the descriptor $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$. For a certain descriptor $\boldsymbol{a}_i$, the best match is attained when $d_{ij} = \min(d_{ij})$ for all $j = 1, 2..n$. In case of many similar features in the images, a correspondence is marked as valid when the following criterion is true:

$$\frac{\text{distance of the best match}}{\text{distance of the second best match}} < \sigma, \tag{2.12}$$

where σ is a threshold between 0 and 1. A reasonable value of σ is between 0.6 and 0.9. For medical applications, one usually needs a value between 0.9 and 1.

2.2.3 Intensity-based image registration

Multimodal images are acquired through different methods and represent different properties of the object, such as CT, MRI and PET, therefore the intensities of the same object may vary a lot. However, they are far from independent. The correspondence between the images can be statistically measured. To statistically describe the intensity patterns in one image, the information theory proposed by Shannon [7], can also be applied here [8].

The entropy, which is the expected information, of an image A with the intensity a of a certain pixel is defined as:

$$H(A) = -\sum_{a} p_A(a) \log p_A(a), \tag{2.13}$$

where $p_A(a)$ stands for the probability of pixel intensity a in image A, i.e. the ratio of the number of pixels with intensity a to the total number of pixels in image A.

Consider another image B. If A and B are multimodal images, then A and B will share some information. For example, if A and B are CT and MR images of the same patient's heart, though they contain different details, they share the same information about where the heart is located. The joint entropy of A and B, which describes the overall expected information of the system AB, will be less than the simple summation of the individual entropy. It is defined as:

$$H(A, B) = -\sum_{a,b} p_{A,B}(a, b) \log p_{A,B}(a, b). \tag{2.14}$$

This shared information situation is illustrated in Figure 2.6. The blue overlapped part is the information that both A and B contain and is known as mutual information. The relation of entropy and mutual information is:

$$\begin{aligned} I(A, B) &= H(A) + H(B) - H(A, B) \\ &= \sum_{a,b} p_{AB}(a, b) \log_2 \frac{p_{AB}(a, b)}{p_A(a) \cdot p_B(b)}. \end{aligned} \tag{2.15}$$

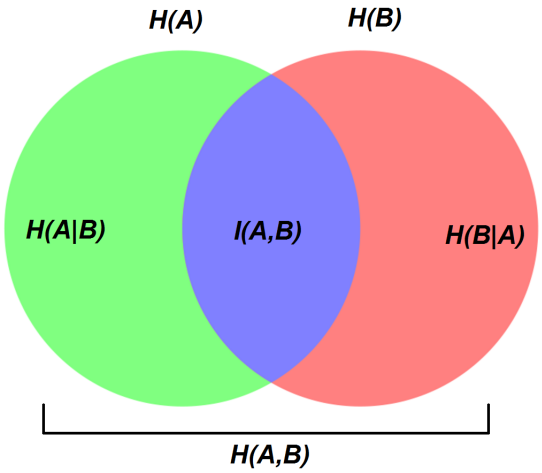

Figure 2.6: Graphical explanation of mutual information.

Image registration is the alignment of images. Thus, it is obvious that the mutual information $I(A, B)$ would be largest when the two images are perfectly aligned. Assuming that image A and B are related by a geometric transformation $\boldsymbol{T}$, defined by parameters α, a pixel L with intensity a in image A is statically dependent on pixel $\boldsymbol{T}_\alpha(L)$ with intensity b in image B, that is:

$$a = A(L),$$

$$b = B(\boldsymbol{T}_\alpha(L)).$$

According to the argument of mutual information theory, the following equation holds true:

$$\alpha^* = \arg\max_\alpha I(A, B), \tag{2.16}$$

where $\boldsymbol{T}_{\alpha^*}$ is the ideal transformation that perfectly align the two images.

2.3 Deep learning

Arthur Samuel, a pioneer in the field of artificial intelligence, said in 1959: *machine learning is a field of study that gives computers the ability to learn without being explicitly programmed* [9]. It is a method of data analysis which allows the automation of analytical model building. In general, machine learning algorithms are used to find out patterns in the given data, such as speech recognition and face detection in images. Based on different tasks, machine learning algorithms can be roughly classified into three types: supervised learning, unsupervised learning and reinforcement learning [10].

In supervised learning, the algorithm is presented with training data together with the desired output as a guideline. The goal of the algorithm is to learn the specific patterns that map the input to the output. For unsupervised learning, the algorithm does not have any knowledge about the desired output. Thus, the objective is usually to discover the hidden patterns in the given data. Unlike supervised and unsupervised learning, reinforcement learning deals with a dynamic environment. It is often used to perform a certain goal without being told if it is correct or not, such as driving an autonomous car. From the perspective of the outputs, machine learning algorithms are usually applied to solve the following problems: clustering, regression, classification, density estimation and dimensionality reduction [10]. In this thesis, machine learning are used for a classification task.

Figure 2.7 presents a simple classification task performed by a supervised learning algorithm. The training data is marked by dots, which have two colors, red and blue, representing different desired output, 1 and -1 respectively. Based on the visualization of the training data, the pattern that the algorithm is supposed to learn is that the plane can be roughly divided by two lines into two parts. To test how well the algorithm learns this pattern, the algorithm needs to predict the output of another data set, called test set, marked as asterisks in Figure 2.7 and colored by their true values. As can be observed from the figure, most of the validation data are correctly predicted, with only a small proportion wrong, which are marked by green circles. This indicates that the algorithm has learned the suggested pattern well.

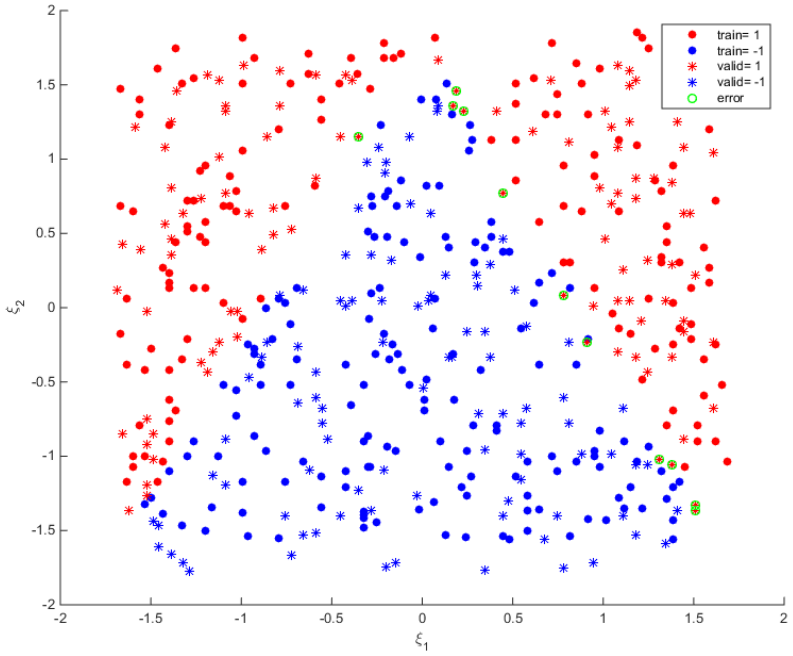

Figure 2.7: A typical classification task.

2.3.1 Artificial neural networks

Among all machine learning algorithms, such as support vector machines, random forests and Bayesian networks, neural networks excel in many important problems, for instance image recognition tasks and natural language processing [10].

As the name neural network implies, this type of machine learning algorithm is inspired by the real neural network inside our brains. The artificial neural network is composed of connected artificial neurons. The simplest artificial neuron is a perceptron. Developed by Frank Rosenblatt [11] and based on previous work by Warren McCulloch and Walter Pitts [12], a perceptron takes binary inputs and gives one binary output, where inputs are the dendrites, output is the axon and the artificial neuron is the nucleus in the real neuron.

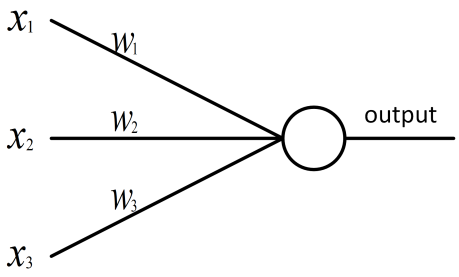

Figure 2.8: An example of a perceptron.

The perceptron shown in Figure 2.8 has three inputs, x_1, x_2, x_3. Each input is connected to the neuron with a weight w_i. The rule to compute the output is rather

simple. The neuron will be fired and output 1 if the weighted sum is larger than a certain threshold, otherwise the output is set to 0, i.e.

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold}, \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold}. \end{cases} \tag{2.17}$$

It is apparent that by adjusting the weights, the output of the perceptron can be changed even though the inputs remain the same. This indicates that the perceptron model can be used to make decisions, i.e. to map the input and the output. It seems feasible to say that a complicated network composed of many connected perceptrons can make delicate decisions based on many input factors. In the following example illustrated by Figure 2.9, the three perceptrons in the first layer only make very simple decisions according to the inputs, while the decisions made by the four perceptrons in the second layer are based on the results of the first layer perceptrons. This mechanism means that the decision made by a latter layer is usually on a more abstract, complex and probably more general level than its former layer.

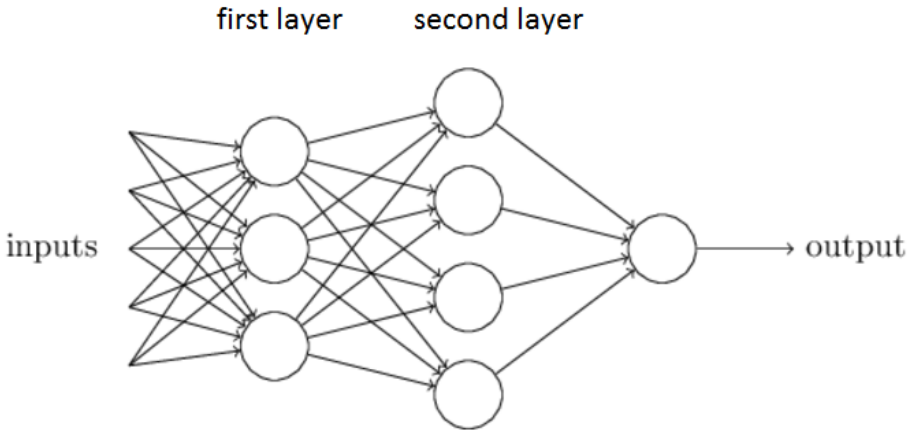

Figure 2.9: Network of perceptrons. Note: the multiple outputs of a perceptron are only a way to imply that the sole output is used as inputs to different perceptrons in the next layer.

In a supervised learning process, the weights of the network and the thresholds of the neurons (usually called bias, denoted by b) need changing in order to map the inputs to the desired outputs, until a certain criterion is fulfilled. For most feasible learning methods, it is reasonable to infer that a small change of the parameters will also cause a small corresponding change in the results. However, the perceptron does not satisfy the general learning rule. As a matter of fact, due to the binary output, a small change in the weights or bias of any perceptron in the network might totally flip the final output. This makes the learning process hard to control.

To overcome this obstacle, another type of neuron is introduced, the sigmoid neuron. It is much alike the perceptron with different weights for different inputs. But instead of the binary 0 and 1, the output of the neuron is $\sigma(\sum_j w_j x_j + b)$, where σ is the sigmoid function, defined as

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}. \tag{2.18}$$

Generally, the function calculating the output is called the activation function, as in the neuron's being activated under certain conditions. Thus, the activation function of a perceptron is a step function. From Figure 2.10, it is clear that a small change of $z = z + \Delta z$ in the step function may completely change the result. On the contrary, the sigmoid function is smoother. In fact, the change in the output can be well approximated by

$$\Delta\text{output} \approx \sum_j \frac{\partial\,\text{output}}{\partial w_j}\Delta w_j + \frac{\partial\,\text{output}}{\partial b}\Delta b. \tag{2.19}$$

However, this equation is not true for perceptrons. Therefore, a network made up of sigmoid neurons learns better than a perceptron network.

As stated above, the learning process is a procedure for changing the weights and biases. The learning outcome is a function that maps the inputs to the output. As a matter of fact, a multilayer network is a universal approximator [13], which means it has the ability to simulate any arbitrary functions.

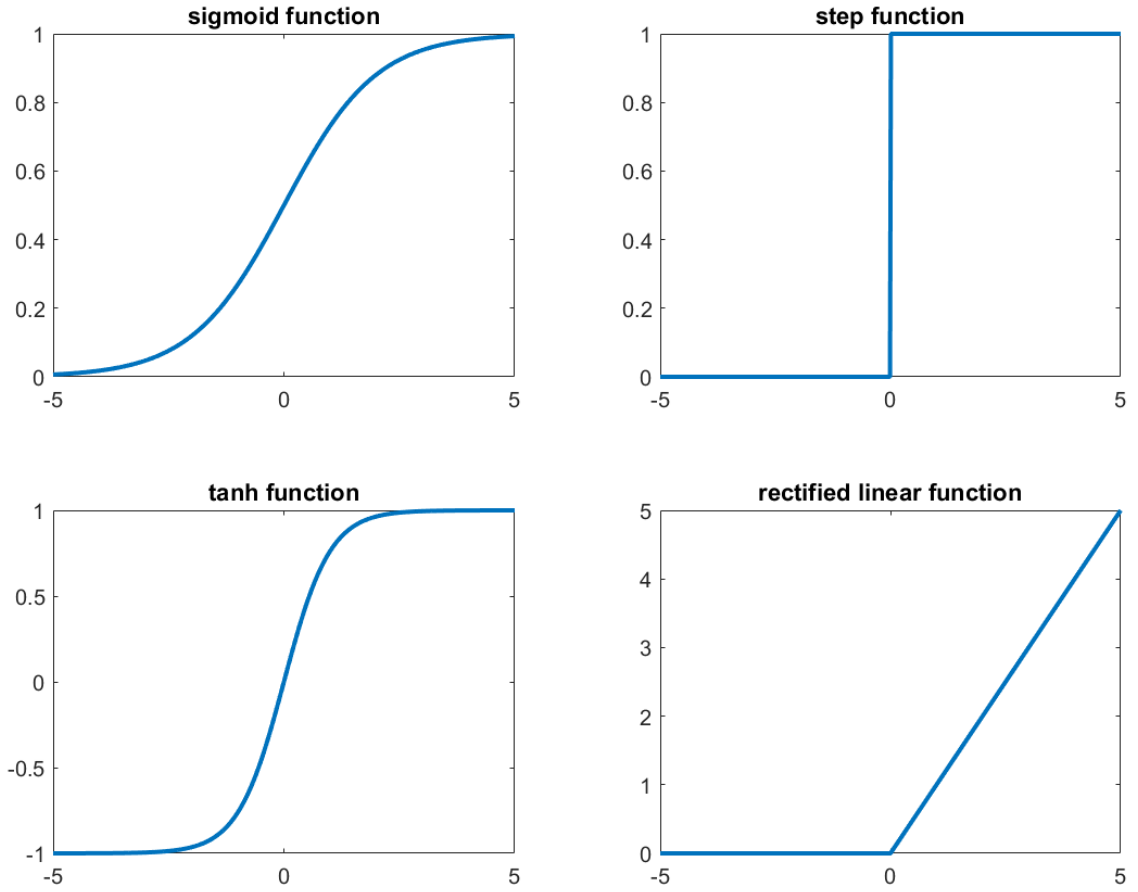

Figure 2.10: Widely used activation functions. Sigmoid function: $\sigma(z) = 1/(1 + e^{-z})$. Step function: $\sigma(z) = (\text{sgn}(z) + 1)/2$. Hyperbolic tangent function: $\sigma(z) = \tanh(z)$. Rectified linear function: $\sigma(z) = \max(0, z)$

2.3.2 Back propagation

To successfully build the connection between the inputs and the output, a strategy of tuning network parameters, i.e. weights and biases, is required. This problem can be formed as an optimization problem, as in finding out a set of weights and biases that produces outputs as close to the desired outputs as possible. This problem can be solved by the back propagation algorithm, proposed by David Rumelhart [14]. To illustrate this method, the network structure shown in Figure 2.11 is used.

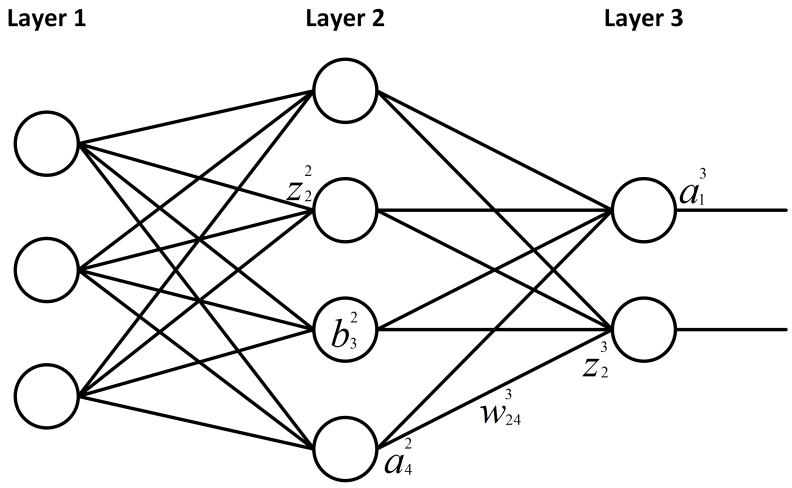

Figure 2.11: Structure of a multilayer neural network. w_{jk}^l stands for the weight from the kth neuron in layer $l-1$ to the jth neuron in layer l. b_j^l represents the bias of the jth neuron in layer l. a_j^l is the activation of the jth neuron in layer l. Also denote z_j^l as the weighted sum to the jth neuron in layer l.

The first layer of the network is the input layer, thus is $\boldsymbol{a}^1$ the inputs. The last layer, layer L, is the output layer, and therefore is $\boldsymbol{a}^L$ the outputs. All the other layers in between are called hidden layers. To make use of the fast linear algebra routines to perform the computation, it is possible to vectorize the variables and furthermore define weight and bias matrices as $\boldsymbol{w}^l = w_{jk}^l$ and $\boldsymbol{b}^l = b_j^l$. Hence, for the network in Figure 2.11, it holds that $\boldsymbol{w}^2 \in \Re^{4\times 3}$ and $\boldsymbol{w}^3 \in \Re^{2\times 4}$.

As defined previously, the relation between these variables are:

$$\boldsymbol{z}^{l+1} = \boldsymbol{w}^{l+1}\boldsymbol{a}^l + \boldsymbol{b}^l, \tag{2.20}$$

$$\boldsymbol{a}^{l+1} = f(\boldsymbol{z}^{l+1}). \tag{2.21}$$

Computing these variables is called forward propagation. Since the network does not contain any direct loops, it is called a feed forward neural network.

Similar to most optimization problems, it is necessary to define a cost function. As an example, the cost function of the network in Figure 2.11 can be described by a quadratic function

$$C = \frac{1}{2}\|\boldsymbol{a}^L - \boldsymbol{y}\|^2 \tag{2.22}$$

where y represents the desired outputs, in this case $\boldsymbol{y} \in \Re^{2\times 1}$. Thus, it is natural to define a parameter updating rule as follows

$$w_{jk}^l = w_{jk}^l - \eta \frac{\partial C}{\partial w_{jk}^l}, \tag{2.23}$$

$$b_j^l = b_j^l - \eta \frac{\partial C}{\partial b_j^l}. \tag{2.24}$$

Such an updating rule guarantees that the updated parameters w_{jk}^l and b_j^l will lead to a lower cost when the cost function is not at its local minima and stay at the local minima after reaching it. This method is known as gradient descent, where the parameter η, the learning rate, controls how fast the descent will be. A crucial part of gradient descent is to calculate the partial derivatives, which can be efficiently computed by the back propagation algorithm.

To calculate the derivatives, it is useful to define an error term δ_j^l, which is used to describe how much error in the outputs that the neuron causes with respect to its weighted sum input

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}. \tag{2.25}$$

The following derivation shows how to calculate the error term for different neurons in different layers.

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial f(z_k^L)}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} f'(z_j^L). \tag{2.26}$$

For $l = 2, 3..L-1$:

$$\begin{aligned} \delta_j^l &= \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial}{\partial z_j^l} (\sum_m w_{km}^{l+1} a_m^l + b_m^l) \\ &= \sum_k \delta_k^{l+1} (\sum_m \frac{\partial}{\partial z_j^l} w_{km}^{l+1} f(z_m^l)) = \sum_k \delta_k^{l+1} w_{kj}^{l+1} f'(z_j^l). \end{aligned} \tag{2.27}$$

The above two equations can also be vectorized

$$\boldsymbol{\delta}^L = \nabla_{\boldsymbol{a}} C \odot f'(\boldsymbol{z}^L), \tag{2.28}$$

$$\boldsymbol{\delta}^l = (\boldsymbol{w}^{l+1})^T \boldsymbol{\delta}^{l+1} \odot f'(\boldsymbol{z}^l). \tag{2.29}$$

The symbol $\odot$ here represents element-wise product, or Hadamard product.

The reason to introduce the error term is that it makes it easier to calculate the partial derivatives in Equation 2.23 and 2.24. The partial derivatives can be derived as follows.

$$\begin{aligned} \frac{\partial C}{\partial w_{jk}^l} &= \sum_k \frac{\partial C}{\partial z_k^l} \frac{\partial z_k^l}{\partial w_{jk}^l} = \sum_k \frac{\partial C}{\partial z_k^l} \frac{\partial}{\partial w_{jk}^l} (\sum_m w_{km}^l a_m^{l-1} + b_m^{l-1}) \\ &= \sum_k \sum_m \delta_k^l a_m^{l-1} \frac{\partial w_{km}^l}{\partial w_{jk}^l} = \delta_j^l a_j^{l-1}, \end{aligned} \tag{2.30}$$

$$\frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial b_j^l} = \sum_k \delta_k^{l+1} \frac{\partial}{\partial b_j^l} (\sum_m w_{km}^{l+1} a_m^l + b_m^l) = \delta_j^{l+1}. \tag{2.31}$$

From Equation 2.28 to 2.31, it is clear that the error term and the derivatives are calculated backward from the last layer to the first layer, therefore, this method is called back propagation. The back propagation algorithm combined with gradient descent may be formulated as Algorithm 2.

Algorithm 2 Gradient descent with back propagation

1: Initialize the network weights and biases, see Section 3.2.2 for details
2: Perform forward propagation, as in Equation 2.20 and 2.21
3: Evaluate the network by the cost function
4: Perform backward propagation, as in Equations 2.28 to 2.31
5: Update the weights and biases, as in Equation 2.23 and 2.24
6: Repeat step 2 to 5 until a certain criterion is reached

2.3.3 Convolutional neural networks

The multi-layer network discussed above is powerful but still has some problems. It is a fully connected network, i.e. a neuron in one layer connects to all the neurons in the next layer. Such full connectivity makes it difficult to train a larger network. Due to the unstable gradient [15] and the large number of hidden layers, the learning will be slow. In addition, in image recognition, it is natural to extract and identify features. However, such features are not well expressed in a fully-connected multi-layer network. In recent years, the convolutional neural network, originally proposed by Yann LeCun [17], has been widely used in image recognition, language processing and other related areas.

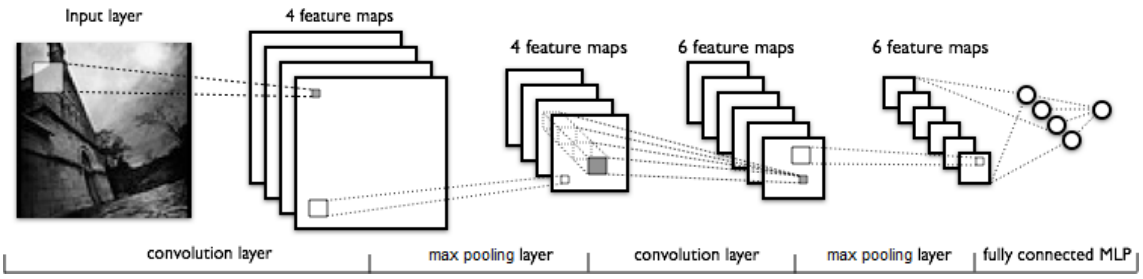

Figure 2.12: Structure of a convolutional neural network [18]. Details are explained in the following sections.

The most distinctive feature of a convolutional neural network is that the neurons are arranged in 3D, as in the example shown in Figure 2.12. Normally, convolutional neural networks are built based on three basic ideas: sparse connectivity, shared weights and pooling.

As stated above, a neuron in a hidden layer of a multi-layer neural network has connections to all the neurons in the former layer. Contrary to full connectivity, a convolutional network is sparsely connected. Each hidden neuron in the convolutional network has its own local receptive field and is not "responsible" for any variations outside the field. In this way, the hidden neurons are only sensitive to a certain spatially local input pattern. Figure 2.13 illustrates the local receptive field in the first hidden layer.

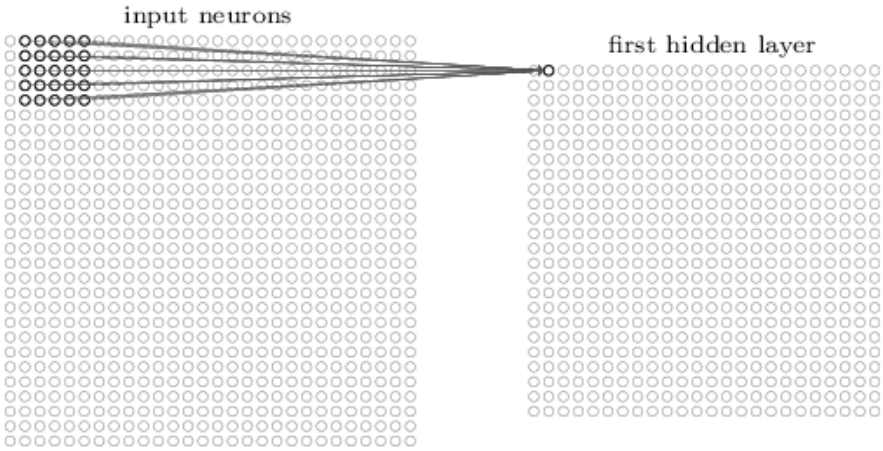

Figure 2.13: Local receptive field [15]. The input layer has 28×28 neurons (i.e. pixels). The local receptive field is 5×5 , that is, a neuron in the hidden layer is only connected to 25 certain neurons in the input layer. The first hidden layer is then of size 24×24.

Still using the example in Figure 2.13, each neuron has $5 \times 5 = 25$ weights and one bias. Unlike the fully-connected multi-layer network, where the weights of the neurons are independent, in a convolutional network, all the neurons in one hidden layer share the weights and bias. This implies that for the j, kth neuron in the hidden layer, the output is

$$a^2_{jk} = f(b + \sum_{m=1}^{5} \sum_{n=1}^{5} w^2_{mn} a^1_{j+m,k+n}). \tag{2.32}$$

It is obvious that the shared weights and bias dramatically reduce the number of parameters, making the network easier to train and also reducing the risk of overfitting. Also, it makes all the neurons extract the same feature at different locations, which is quite useful when analyzing images. Therefore, the hidden layers are also called feature maps and the weights matrix is also known as a kernel or a filter. From a closer look at Equation 2.32, it is easy to spot that the summation is actually a convolution, and the equation can be vectorized as

$$\boldsymbol{a}^2 = f(b + \boldsymbol{w}^2 * \boldsymbol{a}^1). \tag{2.33}$$

The number of filters is usually not 1. Figure 2.14 shows an example of how convolution works for multiple feature maps and filters. There are 4 feature maps in

layer $l-1$, and 2 in layer l. Let $\boldsymbol{w}_{pq}^{l}$ be the weight matrix from the qth feature map in layer $l-1$ to the pth feature map in layer l, $\boldsymbol{a}_{p}^{l}$ and $\boldsymbol{b}_{p}^{l}$ be the activation matrix and the bias vector of the pth feature map in layer l respectively. Then, the output may be written as

$$\boldsymbol{a}_{p}^{l} = f(\boldsymbol{b}_{p}^{l} + \sum_{q} \boldsymbol{w}_{pq}^{l} * \boldsymbol{a}_{q}^{l-1}). \tag{2.34}$$

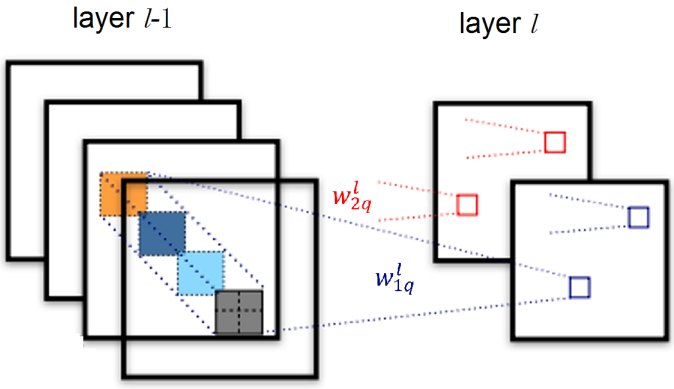

Figure 2.14: An example of a convolutional layer. There are 4 feature maps in layer $l-1$, and 2 feature maps in the following layer l. Each feature map in the latter layer corresponds to one filter, $\boldsymbol{w}$.

Another concept in convolutional networks is pooling. It is usually used right after the convolutional layer. The pooling layer extracts information from feature maps and compresses them into a smaller map. Normally, the information is the maximum value of a non-overlapping rectangle, which is called max-pooling. Unlike the convolutional layer, the pooling is performed on each feature map separately. If the pooling size is 2×2, and there are 4 feature maps of size 24×24, the result will be 4 condensed feature maps with size 12×12. In addition to max-pooling, there are also other types of pooling layers, such as l^2 pooling, which uses the l^2 norm instead of the maximum value. Using pooling can obviously reduce the amount of computation for latter layers, but is also provides a translation invariance [19]. An example is described in Figure 2.15.

Combining everything together, a convolutional network consists of one or several serial or parallel convolutional layers and pooling layers. In order to obtain the desired output, a fully connected multi-layer network is usually appended in the end.

2.3.4 Hyper parameters

The presented theory above is sufficient to build a convolutional neural network. Provided the gradient descent and back propagation algorithm, it is also possible to start training the network to learn the patterns in the given data. However, a practical question is how to choose the parameters that define the network and the

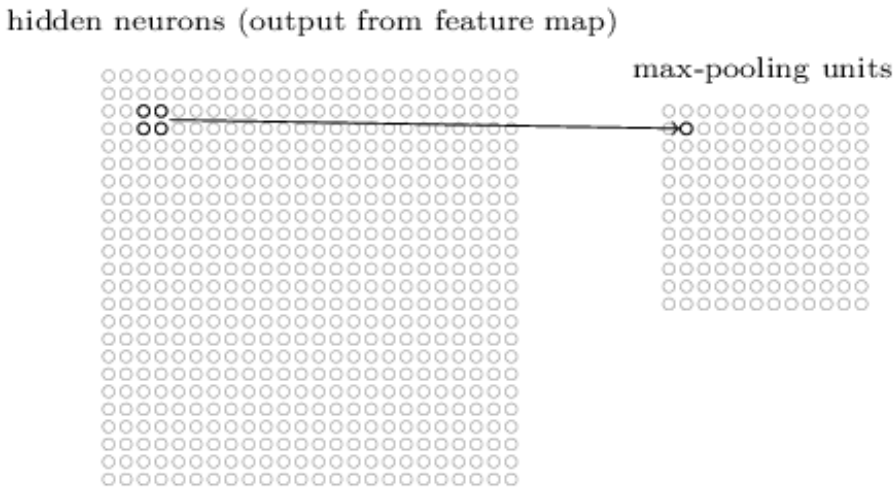

Figure 2.15: An example of a max-pooling layer [15].

learning process. It is usually the case that an improper choice of parameters results in the inability of the network to actually discover any patterns. It is therefore of great significance to be aware of a rough direction of tuning the parameters, otherwise the output may not be better than a random guess. Hyper parameters tuning is a very broad topic. The methods discussed below are some of the most commonly used [15].

Cost function

The learning process is indeed the process to minimize the cost function. Thus the choice of cost function will undoubtedly affect the result. Previously, as an example, the cost function is set to be a quadratic function. However, this specific form cannot guarantee a fast convergence speed. In fact, if the cost function is defined by Equation 2.22, the learning speed can be derived as

$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x, \tag{2.35}$$

$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z). \tag{2.36}$$

where x is the training input, σ is the sigmoid activation. Recall the shape of sigmoid function in Figure 2.10. It can be seen that when the output is close to 1, the curve becomes quite flat and its derivative becomes very small. This will inevitably cause the learning to slow down.

One way to deal with this problem is to use a cross-entropy cost function,

$$C = -\frac{1}{n}\sum_{x}(y \ln a + (1 - y)\ln(1 - a)), \tag{2.37}$$

where n is the number of training data. Using chain rule, its derivatives are

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j(\sigma(z) - y), \tag{2.38}$$

$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x (\sigma(z) - y). \tag{2.39}$$

The results indicate that using the cross-entropy cost-function, the learning speed is controlled by the error in the output layer. The larger the error is, the faster it corrects.

Another similar and widely used method for solving this problem is by introducing a softmax layer as a type of output layer. The inputs to the softmax layer are of no difference to the previous layers. When calculating the output of the softmax, a *softmax function* is applied instead of the normal activation functions, such as a sigmoid function. The output, i.e. activation, is given by

$$a_j^L = \frac{\exp(z_j^L)}{\sum_k \exp(z_k^L)}. \tag{2.40}$$

One characteristic of this function is that all the outputs are positive numbers. Another is that the summation of all the outputs equals one. Hence, the outputs from a softmax layer can be regarded as probabilities.

The cost function following a softmax layer is similar to the cross-entropy cost function, but simpler. It is known as negative log-likelihood, defined as

$$C = -\ln a_y^L. \tag{2.41}$$

Its corresponding derivatives are

$$\frac{\partial C}{\partial w_{jk}^L} = a_k^{L-1}(a_j^L - y_j), \tag{2.42}$$

$$\frac{\partial C}{\partial b_j^L} = a_j^L - y_j. \tag{2.43}$$

It is apparent that the learning slowdown problem does not exist here.

In many cases, both the sigmoid plus cross-entropy and softmax plus negative log-likelihood work well. But choosing softmax is particularly useful when the probabilities are desired.

Learning control

The learning speed is not only controlled by the partial derivatives of the cost function. The learning rate, η also controls the gradient descent. A large learning

rate may help to give a quick start, but later on it may also miss the minimum. A small learning rate may help finding the correct minimum but take a tremendously long time. Therefore, the best approach is to vary the learning rate during the training process. This is called learning rate decay. There are three common types of learning rate decay schemes: step decay, exponential decay and $1/t$ decay [21]. The $1/t$ decay is described by the following equation,

$$\eta = \eta_0/(1 + kt), \tag{2.44}$$

where k represents learning rate decay, t denotes the number of epochs. Thus, a higher learning at the start provides a quick drop to the minima region, but it is slow enough later on to carefully examine the minima point.

Another method, inspired from real world physics, is called momentum based gradient descent [15]. To describe it, a velocity variable, v, is introduced and the rules for updating the weights, Equation 2.23 and 2.24, are slightly changed to

$$\begin{aligned} v' &= \mu v - \eta \nabla C, \\ w' &= w + v', \end{aligned} \tag{2.45}$$

where the momentum parameter μ controls how much friction there is in the system. In practice, μ is often set to a value between 0 and 1.

It is also possible to speed up the learning process by using batch mode, i.e. propagating forward and backward several inputs at the same time. Hence, the batch size becomes another hyper parameter. If too small, the speed up from the hardware and matrix processing library will not be significant. If too large, the weights are simply not updated frequently enough. The obvious trade-off can luckily be easily balanced since the batch size is under most conditions independent from other hyper parameters [15]. Assume the input is not x, but instead it is $x_i, i = 1 \dots m$, where m is called batch size. Batches will be sent forth and back in the network until all data has been processed exactly once. This is called an epoch. The gradients using batch mode are slightly changed according to

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{x_i}. \tag{2.46}$$

The corresponding weight and bias updating rules are also slightly changed

$$\begin{aligned} w^l_{jk} &= w^l_{jk} - \frac{\eta}{m} \sum_i \frac{\partial C_{x_i}}{\partial w^l_{jk}}, \\ b^l_j &= b^l_j - \frac{\eta}{m} \sum_i \frac{\partial C_{x_i}}{\partial b^l_j}. \end{aligned} \tag{2.47}$$

This type of gradient descent is also known as stochastic gradient descent.

Overfitting

As stressed previously, the learning process is an optimization procedure, while the learning outcome is the result of model fitting. Since the neural network (theoreti-

cally) has the ability to simulate any function, the ultimate result will be a model that maps all the input data to the desired output. However, in most circumstances, this ultimate result may not be the best one for classifying unseen data.

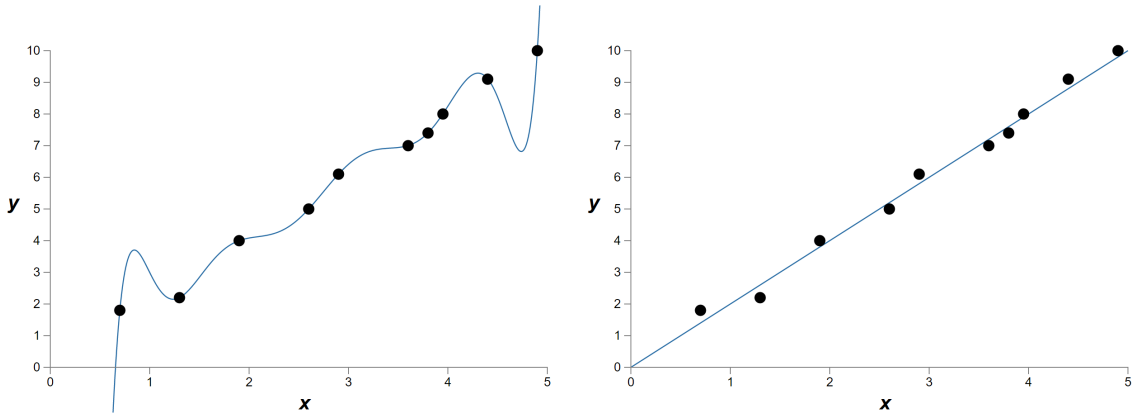

Figure 2.16: Model fitting. For the same points, the left figure shows a 9th order polynomial curve fitting while the right figure shows a straight line fitting.

Figure 2.16 shows an example when the best model might not be the one that maps all the data perfectly. The blue curves are the models that try to map the input data x to the desired output y. The left one, a 9th order polynomial, perfectly maps all the input data to the output data while the right one, a straight line $y = 2x$, does not provide an exact mapping, but it is still a good one. Although either of them could be correct, the simple one is more likely to reveal the underlying truth without any prior knowledge. In this case, the data can either be explained by a 9th order polynomial or a straight line plus some noise. If so, the 9th order polynomial just learns the local noise, and will fail to generalize to unseen data. This is known as overfitting.

To solve the overfitting problem, one obvious approach is to use more training data. The more training data the network uses, the better it generalizes. But sometimes obtaining data is not easy. A simple solution to this is to artificially expand the training data. For instance, adding a little noise or rotating the image by a small angle. This approach usually provides a good result and has been widely used, see [20].

Another technique to reduce the overfitting problem is to introduce a regularization term in the cost function. The reason for doing so is to prevent one or several weights from being too large such that it dominates the network and affects the output. Two commonly used regularization terms are l_1 and l_2 regularization. For l_1 regularization the cost function is now given by

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|, \tag{2.48}$$

and for l_2 regularization, it is given by

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2, \tag{2.49}$$

where C_0 is the original cost function and the parameter λ is called weight decay.

For neural networks, one unique method to reduce overfitting is called dropout. Unlike l_1 and l_2 regularization, dropout does not change the cost function. Instead, it modifies the network itself. As discussed above, the reason why overfitting could arise is because a large enough network has the ability to simulate any arbitrary function. Thus, what dropout does is to temporarily reduce the size of the network.

Supposedly, the training network is the left one in Figure 2.17. Before each forward propagation, dropout randomly deletes half of the neurons in the hidden layers, as shown in the right part of Figure 2.17. Thereafter, the forward propagation and back propagation are performed. After repeating this process in every network update, the network will be tuned as a half-sized one. When combining all of them, the amount of neurons will be twice the amount when during training. To compensate for this, the weights are divided by two. A normal back propagation builds up co-adaptations between neurons. This works for the training data, but not for the unseen validation and test data. Random dropout forces the neurons to be independent from each other, and to produce the correct output no matter which neurons are included in the network [27].

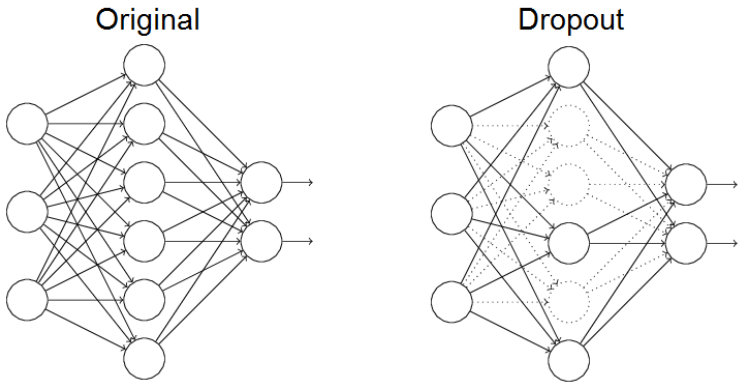

Figure 2.17: Dropout. The left figure presents the original network structure. The right figure shows the neurons that are randomly dropped out (dashed) during the training.

3

Methods

3.1 Ground truth creation

The purpose of this thesis is to automatically segment the pericardium with the help of supervised learning techniques. Though, for this data set, there are no manual delineations of pericardium. The ground truth segmentation file is crucial to the supervised learning algorithm, where it represents the desired output. Therefore, the first step is to create a manual segmentation for the non-contrast CT data. Luckily, the data set also includes contrast CT images with corresponding manual labelling for each patient. Thus, for each patient, a semi-manually delineated segmentation file was created based on the contrast CT images. This was made possible through image registration techniques, that is, registration between the contrast and non-contrast images of the same patient. After the image registration, the semi-manual delineations are examined by the same physician who manually delineated the contrast CT images. The results are scaled from one to five, where five represents better or equal to manual delineation.

3.1.1 Image registration

The registration is done in two parts. First, an initial registration result is obtained through feature-based registration. Then, this result is used as an initialization for the following intensity-based registration. This efficient approach is commonly used and has been proved to yield satisfactory results in [22] and [23].

The feature-based registration can be divided into four steps:

- **Feature detection**: Feature points are detected in both the target image (non-contrast CT) and the source image (contrast CT).
- **Feature description**: Gradient based, SIFT-like feature descriptors are calculated for the detected feature positions.
- **Feature matching**: The correspondence between a feature point in the source

image and another one in the target image is constructed based on their feature descriptors and a certain criterion, such as the Lowe criterion, i.e. the matched feature points are only considered valid if the distance between them is smaller than a certain threshold.

- **Transformation matrix estimation**: An individual affine transformation is estimated between each source and target image using the RANSAC algorithm with the valid matched feature points.

The semi-manual segmentation for the non-contrast image is then obtained by warping the segmentation of the contrast image according to the registration result. Figure 3.1 shows an example of both valid and invalid matched feature points.

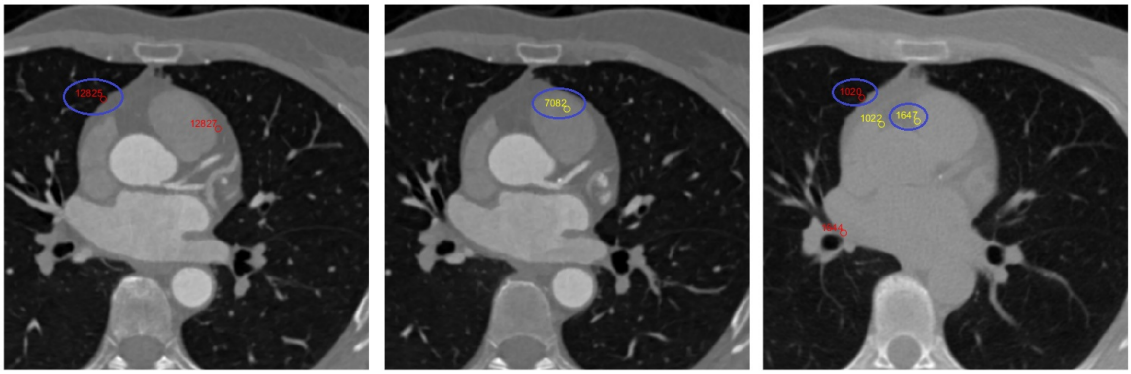

Figure 3.1: An example of matched feature points. The left one and the middle one are contrast images, the right one is the corresponding non-contrast image. Red points are valid matches while the yellow ones are invalid. The threshold for validation in this case is 5mm. In the left and the right images, point 1020 in the non-contrast image matches point 12825 in the contrast image. In the middle and the right images, point 1647 matches point 7082. Point 12827 in the left figure, point 1644 and point 1022 in the right figure are also feature points which match points in other slices and therefore not discussed here. To be noticed, the left and the middle images are not the same slice.

The contrast image and the non-contrast image of the same patient were retrieved at different times. This means the relative position between the patient and the CT scanner are different, which contributes to differences between the contrast and non-contrast images. In addition, the heart changes shape and location at different heart cycles. This contributes to even more differences between the two modalities. As a global transformation, the affine transformation does not contain any local information and cannot fully reflect the complex movement of the organs and tissues in human body. Therefore, a pure affine transformation is not sufficient to capture the full range of deformation needed to transfer the ground truth in an optimal manner. Hence, a second intensity-based, non-rigid registration is applied to compensate for the need of local non-rigid deformations.

The intensity-based registration uses the feature-based, affine registration as an initialization. The goal is to estimate a non-rigid transformation for each source

and target image. This is done by minimizing a cost function based on normalized mutual information. In this thesis, the non-rigid transformation is estimated using the software package *elastix*, which can be found freely available online.

3.1.2 Ground truth evaluation

One apparent approach to evaluate the semi-manual delineations is to visualize the results slice by slice in all three perspectives. Along with the visual inspection, an overall method can also be used for evaluation. The data used in this project consist of both contrast CT images and non-contrast CT images. The contrast and non-contrast images were taken at different patient position, but approximately within half an hour. Thus, the fat volume inside heart, i.e. the epicardial fat, should not differ and is therefore reasonable to use as a criterion to tell how good or bad the segmentation of the non-contrast images are.

As stated previously, the intensity of the CT images varies due to different absorption abilities for different parts of the human body. Therefore, it is reasonable to infer that voxels with similar intensities represent the same kind of tissue. Thus, it is possible to separate fat tissue from other tissues, such as muscle tissues, in a voxel intensity histogram.

Figure 3.2 illustrates an example of the histogram of a CT image, in which four local maxima can be clearly seen. They represent four different tissues. According to Figure 2.2, the peak at around -70HU represents fat tissue. Thus the area under that curve is the fat tissue volume.

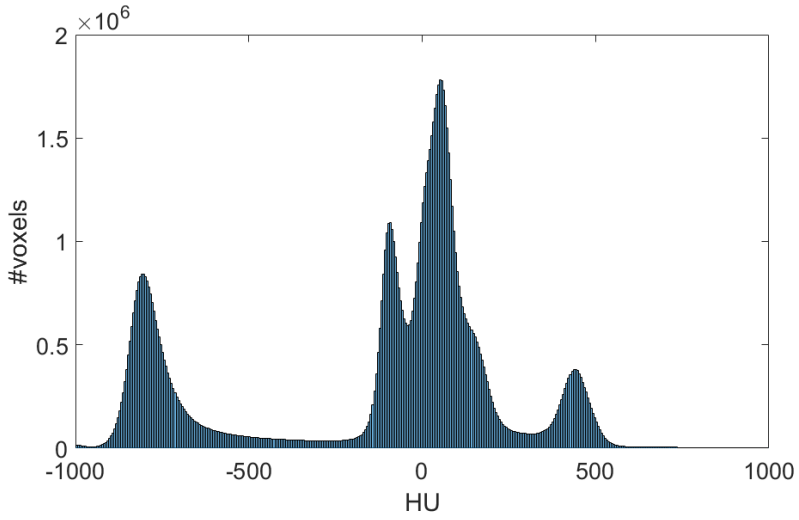

Figure 3.2: Histogram of a CT image. The x-axis represents the HU intensity and y-axis represents the number of voxels.

In order to eliminate the influence of the contrast agents diffused into the fat tissue, the Hounsfield Unit interval of fat tissue is calculated separately for contrast and

non-contrast images. In addition, since the same CT scanner settings were used when acquiring the images, the intervals should not significantly vary from patients to patients. Therefore, CT image histograms are constructed based on all patients' data, to be specific, the sum of the normalized probability of each patient.

3.2 Supervised Learning

A CNN is built in order to classify all the voxels in the non-contrast CT images as either inside pericardium or not. The results of the intensity-based registration are reckoned as ground truth in the supervised learning, while the original non-contrast CT volume data is input to the neural network. After consulting with the physician who manually delineated the contrast CT images, only the non-contrast CT images with the semi-manual segmentation that are equal to or better than segmentation delineated by experts are used for the supervising learning. For each input data, the output from the CNN is designated as a number between 0 and 1, which represents the input voxel's probability of being inside the pericardium. If all voxels from a CT image are sent into the CNN one after the other using a sliding window approach, the result can then be seen as a probability map.

3.2.1 Pre-processing

First, since the same network is trained by multiple images, the images are individually normalized by subtracting the mean and dividing by the standard deviation before any training.

After image normalization, the non-contrast CT image is reconstructed as 3D image patches. Each patch is centered on the voxel to be classified, as shown in Figure 3.3. Based on the position of the voxel, three types of patches are defined: inside the pericardium, outside the pericardium and right on the pericardium. The desired network output, i.e. classification result is chosen in two different ways. The first one classifies a voxel as background if the voxel is outside pericardium, like voxel 1. If the voxel is not outside, for example voxel 2 and 3, the label is denoted as foreground. The second approach focuses on the pericardium itself. A voxel is classified as foreground when it is located right *on* the segmentation, as voxel 3. Otherwise, as for voxel 1 and 2 for instance, it is labelled as background. To be more detailed, a voxel with a distance to the segmentation boundary lower than a certain threshold, for example 1mm, will be counted as right on the segmentation. Figuratively, the foreground in the first approach is a solid spheroid. While in the second approach, the foreground is just the surface of the object, a hollow spheroid.

The non-contrast CT image used in this project has on average about 20 million voxels. The computing time will unmanageable if patches from all voxels are used for training. Thus, the images are first down-sampled. The sampling strategy is

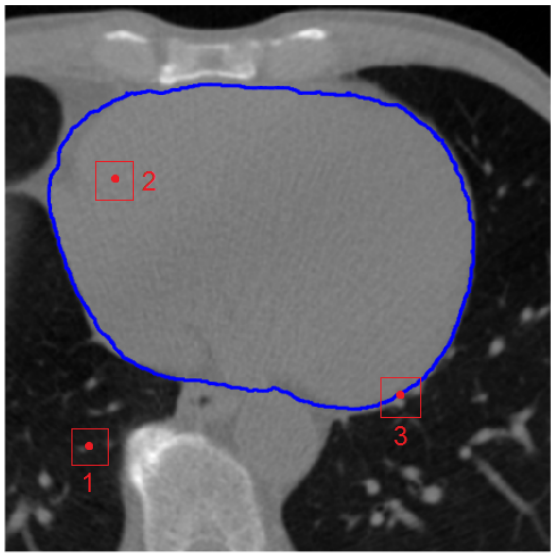

Figure 3.3: Input to CNN. The blue line is the ground truth. The three red boxes represent three different types of image patches. Voxel 1 is outside the pericardium. Voxel 2 is inside the pericardium. Voxel 3 is right on the pericardium.

defined as different sampling intervals in different dimensions. For example, every 25th voxel in the first dimension, every 20th voxel in the second dimension, every 10th voxel in the third dimension is selected and used to extract the corresponding image patch. Depending on the total number of foreground and background voxels, the sampling strategy might be different for foreground and background.

In the second approach, i.e. focusing on the pericardium itself, the number of background voxels is obviously much larger than that of the foreground, which results in an extremely unbalanced training set. In addition, only areas close to the pericardium are of interest. It is then not necessary to include areas such as the center of the heart and much of the lung, since those areas can be easily classified. To effectively exclude such areas, a multi-atlas initialization is applied [33]. The multi-atlas is computed based on all images that are available during the training process. The segmentation of each source image is warped to the target image by an affine transformation, which is calculated by feature-based registration. The sum of all warped segmentations is then called the multi-atlas. Areas that all atlases agree on, i.e. probabilities equal 1 or 0, are considered definitely either inside or outside pericardium and therefore excluded from the patch extraction.

3.2.2 Network training

The CT images are randomly divided into three different subsets: a training, a validation and a test set. Due to the scarcity of images, the training and validation set include more images than the test set: 9 in the training set, 6 in the validation set and 4 in the test set. The training set is the one that the network is actually

trained on. Weights and biases of the network will be tuned based on the training set through back propagation. The validation set is used to tune hyper parameters in the structure of the network, such as network architecture, regularization parameters, etc. The test set will not be used until the very end to evaluate the network's performance.

Three different network structures were tried out in this project, as shown in figure 3.4. Similar structures are commonly used in medical image classification. [32]

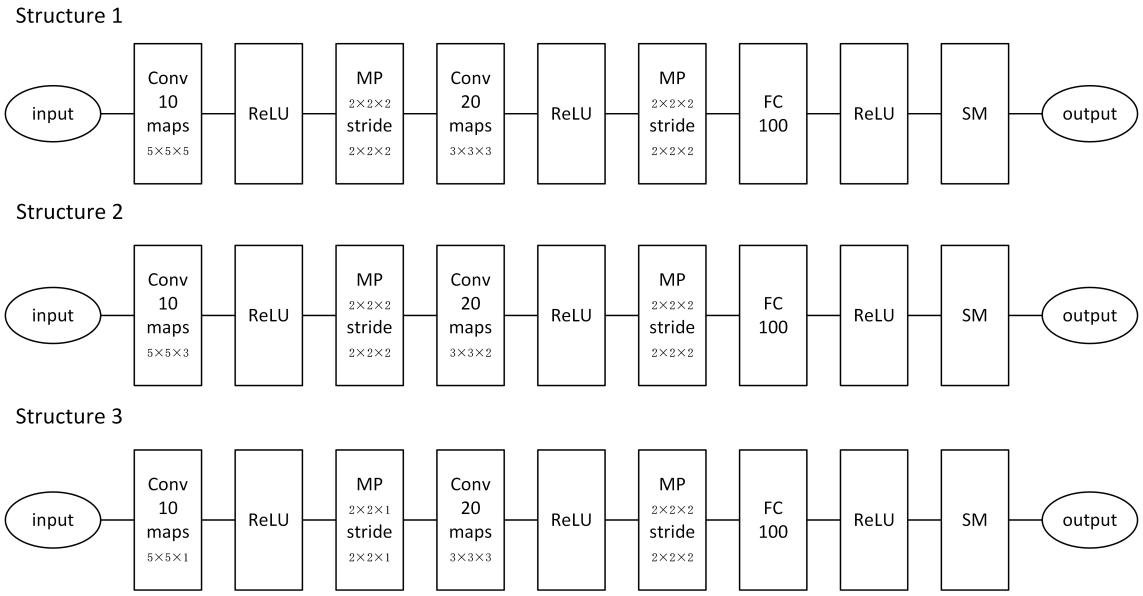

Figure 3.4: Network Structure. Conv: convolution layer, ReLU: rectified linear unit activation, MP: max-pooling, FC: fully-connected layer, SM: softmax layer.

The input patch size for network structure 1 is $28 \times 28 \times 28$, while it is of size $28 \times 28 \times 12$ for network structure 2 and 3. Such a choice is based on the fact that the voxel size is not the same for every dimension. For the first and second dimension, each voxel represents 0.33mm, while a voxel in the third dimension represents 1.5mm. The network structure and the patch size are chosen to contain the approximate same amount of information in each dimension.

For the implementation of the convolutional neural network, the *Torch7* package was used. However, to avoid neurons from being saturated and the resulting learning slow down problem, the default uniform distribution was not used. Instead, the weights and biases were randomly chosen from a Gaussian distribution with mean 0 and standard deviation $1/\sqrt{n_{in}}$, where n_{in} is the number of input neurons [15].

The training process will stop running when either the training accuracy no longer increases or the validation accuracy stops growing. Training or validation accuracy is defined as the ratio of the number of correctly classified voxels to the number of all voxels. A non-increasing validation accuracy indicates that the network has already encountered an overfitting problem, that is, all it learns are local noises instead of general pattern information. An example can be seen in Figure 3.5: although the

training accuracy continues increasing, the validation accuracy stays at the same level after around 250 epochs.

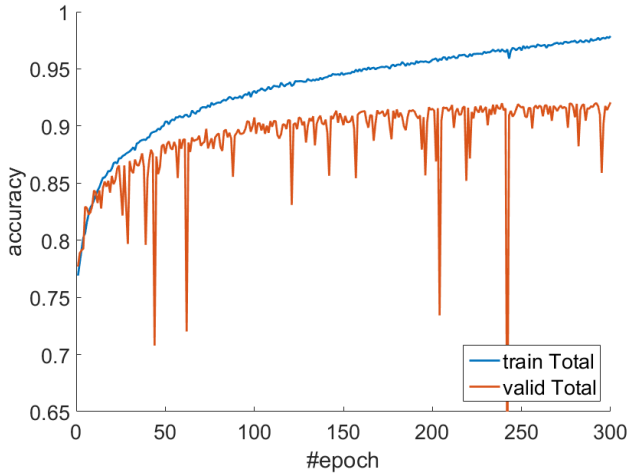

Figure 3.5: An example of overfitting. The blue line represents the overall training accuracy. The red line represents the overall validation accuracy.

3.3 Post-processing

The goal of post-processing is to generate the final segmentation from the output of the CNN, i.e. the probability map. A simple and intuitive method is to set a threshold, such that voxels with intensities below the threshold will be considered as background.

Figure 3.6 shows two examples of probability maps, generated by the two classification approaches, i.e. detection of inside/outside pericardium and on/off pericardium respectively. It is clear that the result is far from being perfect, since the white area includes structures other than pericardium. A simple thresholding will not solve the problem. Some complex post-processing work needs to be done before making any conclusions.

In the upper and lower part of the probability map, it can be seen that the bones are also classified as foreground. In order to get rid of voxels far away from heart, a multi-atlas approach is used to draw regions of interests. Similar to the multi-atlas approach applied in the pre-processing, if a voxel is considered to be background by all atlas maps, it will be marked as background.

Another issue of the probability map is that some voxels inside the heart have a lower probability, therefore a simple threshold will leave some "black holes" inside the heart region. The solution used here is using graph cuts. By defining a proper cost function, the problem can then be formulated as an optimization problem and solved by finding the parameters that minimize the cost function. The final segmentation

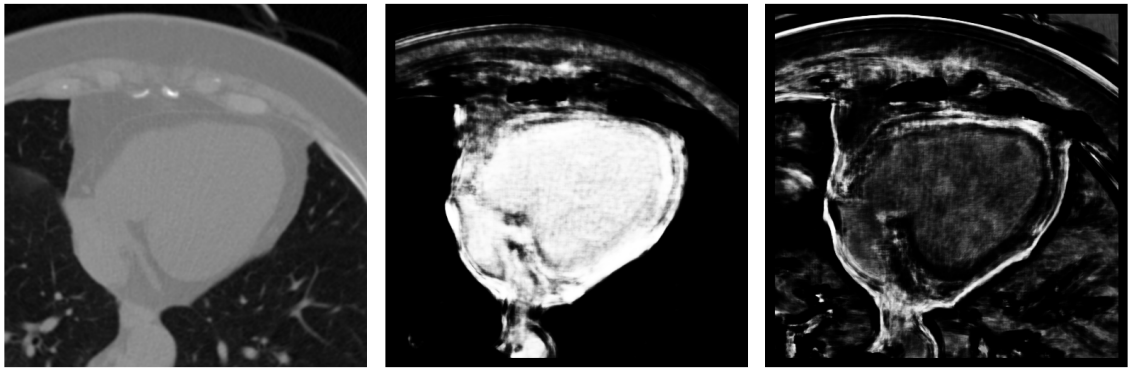

Figure 3.6: Probability Maps. The left figure is one slice of the input CT image, the middle figure is the probability of being inside pericardium, where white corresponds to a high probability and black corresponds to a low probability, the right shows the probability of a voxel belonging to the pericardium border. The larger the intensity is, the more confident the network will be. For the right one, the network is only trained on the data around the segmentation, defined by the multi-atlas initialization. The probability maps are generated by a sliding-window method over all voxels.

$\boldsymbol{x}^*$ is the solution to the following optimization problem [23]

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\{0,1\}^n}{\arg\min}(\sum_{i=1}^{n} x_i(\frac{1}{2} - P(i)) + \lambda \sum_{i=1}^{n} \sum_{j\in N(i)} x_i(1 - x_j)), \tag{3.1}$$

where $P(i)$ is the probability of voxel i given by the CNN output, λ is regularization weight and N is the number of connected neighbourhoods, where a 6-connected neighbourhood is used here. The optimization problem can be solved by using [24]. The first term of the approach is based on the probability P, while the regularization term keeps the segmentation smooth.

Clearly, from Figure 3.6, the probability P in Equation 3.1 should be a combination of both the inside/outside and the on/off pericardium probability map, as the goal is to segment pericardium. Let p_1 be the probability map that tells if it is pericardium or not, i.e. the right figure in Figure 3.6, and p_2 be the probability map that tells inside or outside pericardium, i.e. the middle figure in Figure 3.6. Since p_1 contains information of pericardium and p_2 contains information of whether it is inside pericardium, as an intuition, P could be a linear combination of p_1 and a thresholded p_2:

$$P = p_1 + [p_2 > \alpha]. \tag{3.2}$$

3.4 Evaluation

Finally, the test images are segmented according to the whole pipeline presented in this chapter and then evaluated by the dice coefficient [25]. The dice coefficient is

calculated by the segmentation given by the graph cuts and the ground truth:

$$Dice = \frac{2|A \cap B|}{|A| + |B|}, \tag{3.3}$$

where A is the binary ground truth image and B is the binary segmentation image.

4

Results and Analyses

4.1 Ground truth creation

In order to obtain the best possible semi-manual pericardium segmentation of the non-contrast CT images, i.e. the targets for the supervised learning, the image registration is tuned individually based on each patient. Thus, the registration parameters vary from patient to patient. However, this makes the method less general. Ground truth creation must be done manually by adjusting the parameters and evaluating the performance.

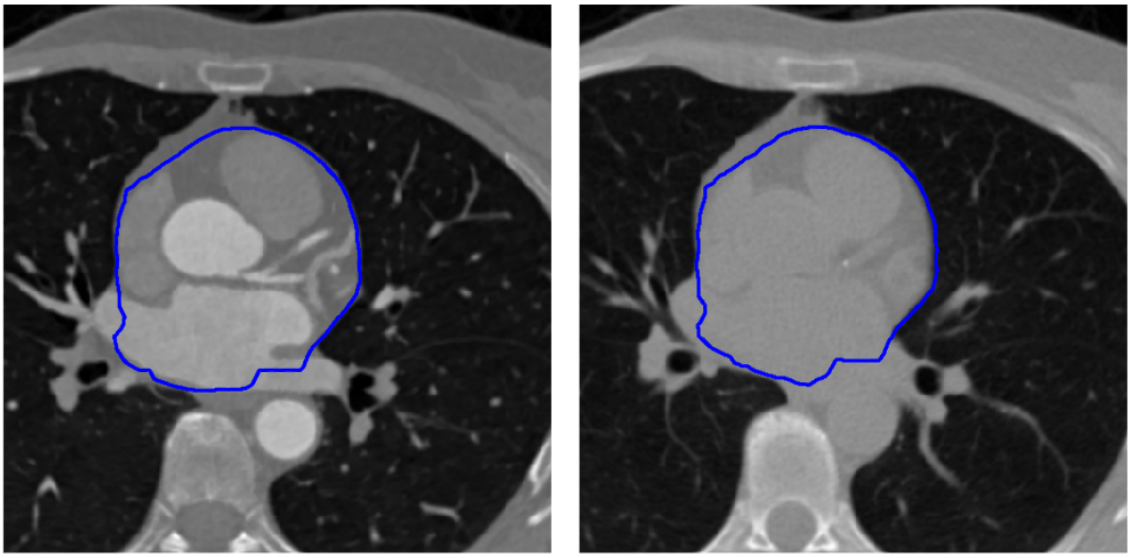

Figure 4.1: An example of feature-based registration result. The left figure depicts a slice of the contrast CT image, with the manual delineation marked as a blue line. The right figure shows the feature-based registration results of the non-contrast image. The semi-manual non-contrast segmentation is a result of warping the contrast segmentation by an affine transformation estimated through RANSAC algorithm.

Figure 4.1 shows an example of the feature-based registration result. As can be seen from the images, the warped segmentation does not perfectly align with the non-contrast volume image. For this specific example, to the right of the heart, the segmentation is supposed to be the boundary between the heart and the lung.

However, the blue line goes into the lung. This problem is mainly caused due to the negligence of the local intensity information. As stated in Section 3.1.1 the semi-manual segmentation can be improved by using an intensity-based non-rigid registration method. A closer look at the result may be found in Figure 4.2. The improvement is clearly visible. The pericardium can be vaguely seen in the left figure, from bottom left to top right. The blue line, which is the result of affine transformation, is further away from the pericardium than the red line, the result after non-rigid transformation. The local intensity information is taken into account when estimating the local deformation field even if the pericardium itself is barely seen.

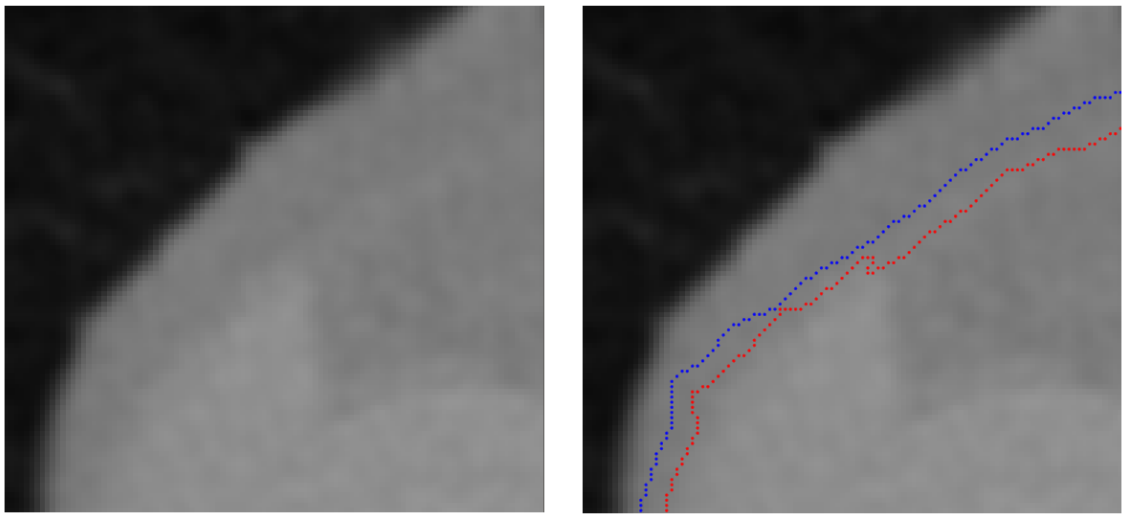

Figure 4.2: An example of the intensity-based non-rigid registration results. The left figure shows the non-contrast CT image. The pericardium, from bottom left to top right, can be vaguely seen. The right figure shows the same image, with the semi-manual segmentation on top of it. The blue line shows the result after featured-based affine registration and the red line marks the final segmentation after intensity-based non-rigid registration.

As described in the previous chapter, the evaluation of the ground truth is partly visual inspection and partly epicardial fat volume comparison, which is calculated from the histograms. Figure 4.3 shows the Hounsfield scales from -300HU to +300HU for contrast and non-contrast images respectively. It is clear that the muscle tissue in the non-contrast image, which peaks at 39HU, has a relatively lower range than the contrast image, which peaks at 60HU, due to the effect of contrast agents. On the other hand, the range of fat tissues differ only slightly in the two curves. Based on the two curves, it is understandable, though not extremely accurate, to say that the trough between the two peaks is the separation point of fat and muscle tissues. Combined with the results of the previous work [22], the fat tissue for contrast images is set between -192 to -41 HU. For non-contrast images, the interval is from -192 to -48 HU.

The epicardial fat volume was estimated for the contrast and non-contrast images respectively based on the method in Section 3.1.2. For the contrast images, the

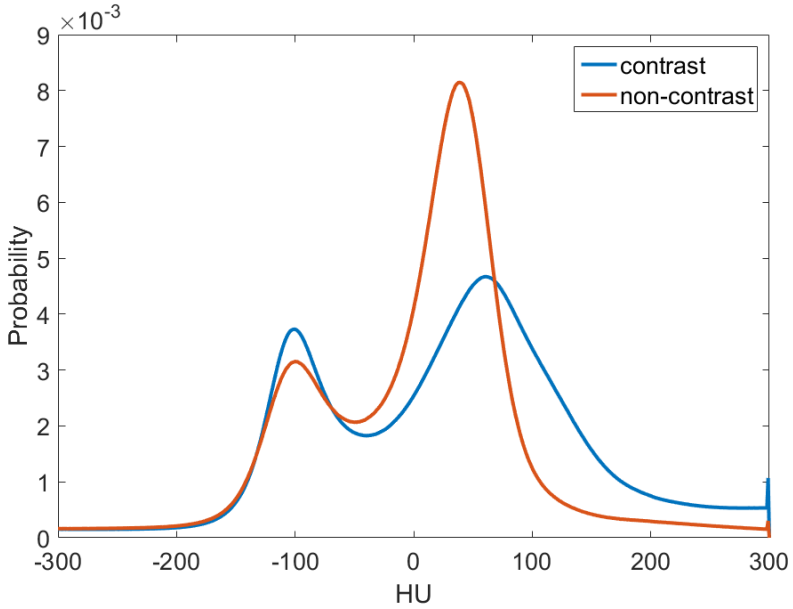

Figure 4.3: Histogram of normalized probability of all images. Only voxels inside the heart and with HU between -300 and 300 are counted.

segmentation used is the manually delineated one. For the non-contrast images, the utilized segmentation is the resulting semi-manual labelling computed by intensity-based non-rigid registration. It is reasonable to assume that if the epicardial fat volume of the contrast image is close to the fat volume of the corresponding non-contrast image of the same patient, the semi-manual segmentation could work well as non-contrast ground truth. However, as Figure 4.4 shows, the epicardial fat volume difference for the contrast and non-contrast images is significantly larger for some of the images. The unexpected large difference might be caused by erroneous semi-manual labelling due to inaccurate registrations or due to an inaccurate estimation of the fat tissue interval. Thus, analysis requires expertise in heart anatomy. After consulting with a physician, 19 out of 27 non-contrast segmentations are marked as correct and continually used for training the CNN later on.

4.2 Supervised learning

The network training process is an optimization procedure that aims to minimize the network cost function and maximize the classification accuracy. Although the cost function is what the gradient descent optimizes, from a practical point of view, what is truly important is the classification result. Hence, training and validation accuracy are used as indicators of how well the network performs.

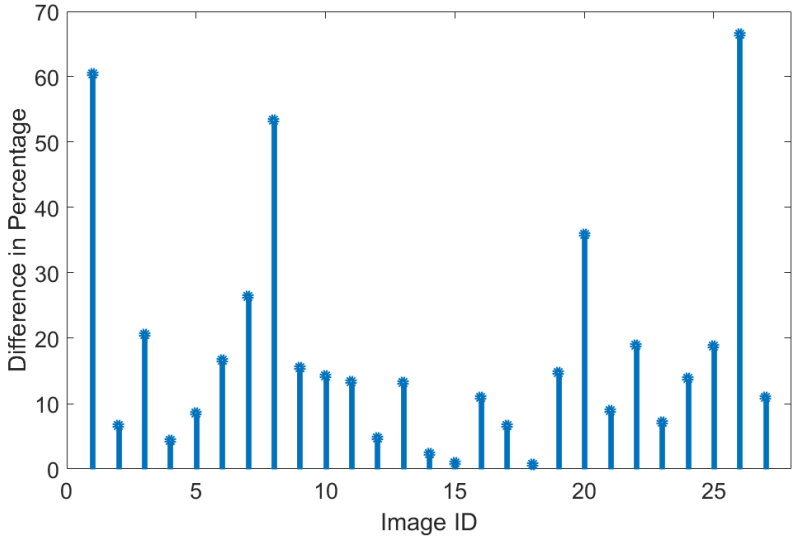

Figure 4.4: Epicardial fat volume difference for the contrast and the non-contrast images.

4.2.1 Hyper-parameter tuning

For the network training process, the hyper-parameters can roughly be divided into two groups. The first ones rather affect the optimization process, and tuning these aims at reaching a higher training accuracy. The second group of hyper-parameters mostly consists of regularization parameters, i.e. tuning these aims at solving the overfitting problem by narrowing the gap between the training and validation set.

Optimization

Learning rate is the main parameter that controls the learning process, defined by Equations 2.23 and 2.24. As discussed in Section 2.3.4, a larger learning rate equals a larger step size in the optimization process. Likely, a larger step size causes the algorithm to omit the global minimum point and thus it cannot escape a local minimum. This can be seen in Figure 4.5. The training and validation accuracy keep increasing until epoch 59, where the system drops into a local minimum. All patches, no matter foreground or background, are classified as background. After this point, the gradient descent algorithm does not manage to bring the system outside the local minimum and remains the same for the remaining 241 epochs.

Using learning rate decay could potentially solve this problem, as discussed in Section 2.3.4. This assumption was confirmed when running the experiments, which can be seen in the right figure of Figure 4.5. The learning rate decay scheme used here is defined by Equation 2.44.

The momentum and batch size (see the theory presented in Section 2.3.4) also affects the learning speed, which can be seen in Figure 4.6. The curve for momentum

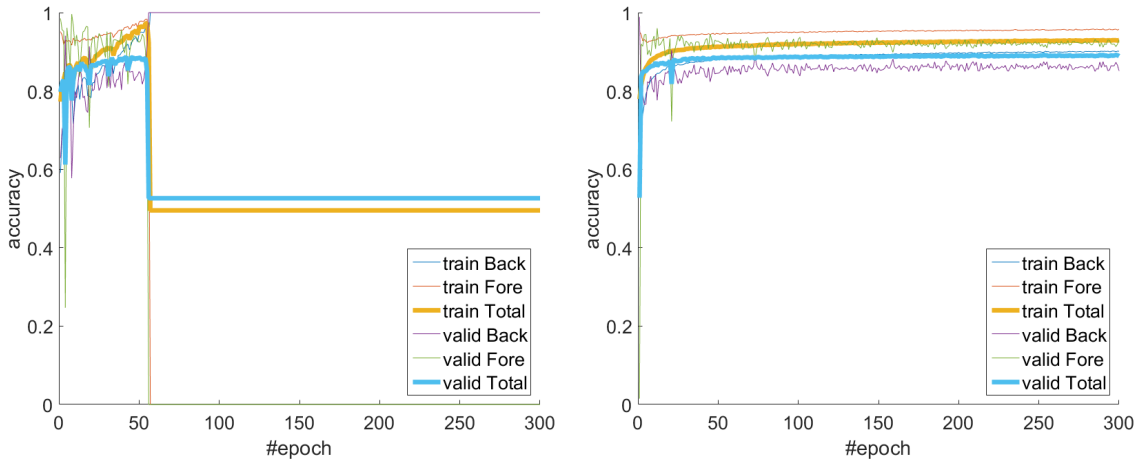

Figure 4.5: Learning rate and learning rate decay. The left figure illustrates the learning process when learning rate is too high. The right figure shows how using learning rate decay may solve the problem. train Fore: training accuracy for foreground voxels, i.e. voxels inside pericardium. train Back: training accuracy for background voxels, i.e. voxels outside pericardium.

implies that the momentum parameter does not have a great impact on the final accuracy, unless the momentum parameter is set to 1, where the velocity quickly builds up due to zero "friction". On the other hand, batch size significantly affects the classification result. This indicates that some trial and error experiments with respect to the batch size have to be conducted to achieve the best performance. In addition, the smaller the batch size is the longer the training time will be. Therefore, choosing a value that balances this trade-off is also of great importance.

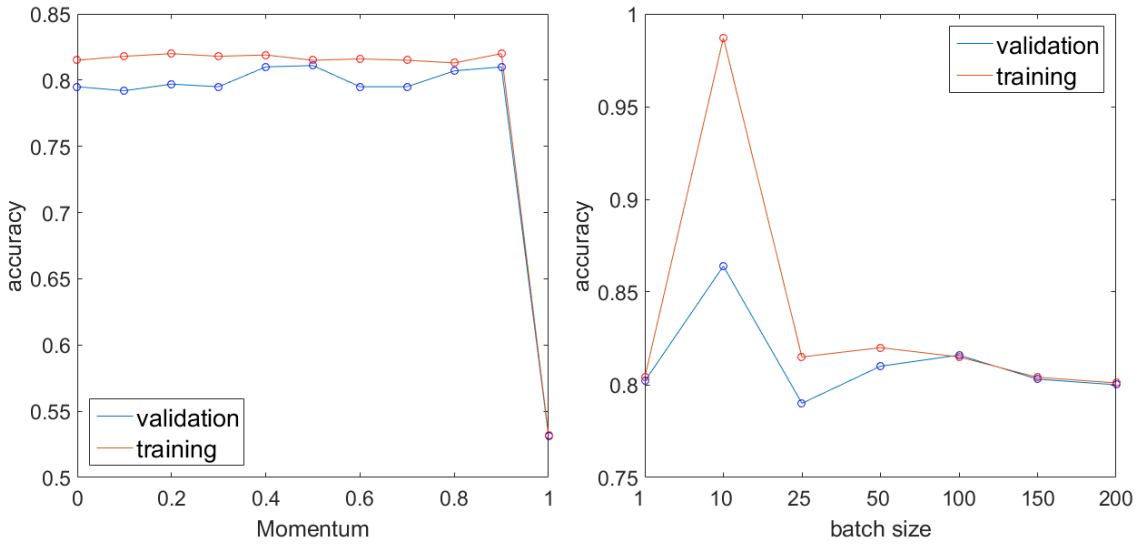

Figure 4.6: Effect of the momentum parameter and the batch size. The left figure shows the effect of momentum, the right figure shows the effect of batch size.

Regularization

In most cases, given enough time and a proper learning rate, the training accuracy is capable of reaching almost 100%. However, due to the scarcity of data and less accurate semi-manual segmentation, the validation accuracy always falls behind the training accuracy, sometimes the gap is so large that the network probably just learns local noise.

The method used in this project to improve this situation is adding weight decay, presented in Section 2.3.4, to the network cost function and dropout layers in the network structure. Figure 4.7 and 4.8 illustrates how weight decay and dropout affect the result. All parameters are the same except the change of weight decay or dropout. The accuracies of points in both figures are results after 300 epochs. This is because the training accuracies almost do not increase after 300 epochs. The computational time would be dramatically longer if the training processes aimed at one certain training accuracy, if possible at all.

As Figure 4.7 shows, there is a general trend that as the weight decay decreases, the training and validation accuracy tends to increase. However, some variations exist when the weight decay is relatively small. In general, a higher weight decay will make the regularization term dominate the cost function and the result will be inevitably around 0.5. Similar conclusion can be drawn from Figure 4.8. The higher the dropout ratio is, the more neurons the network randomly drops, the weaker the network's ability to generalize the data. Clearly, the network will not learn any features in the data if the almost all the neurons are dropped.

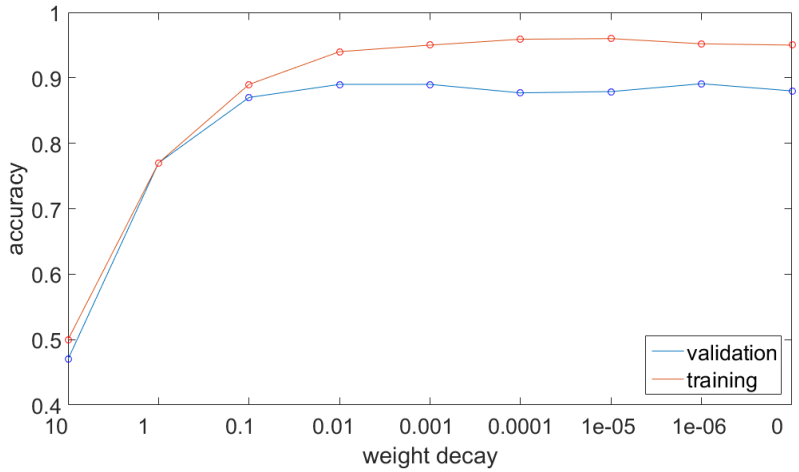

Figure 4.7: Effect of weight decay. The relation between weight decay and accuracy.

An optimal choice of weight decay from Figure 4.7 is 10^{-6}, with a higher validation accuracy and smaller gap between the validation and training, although a relatively lower training accuracy. Due to the same reasons, the dropout ratio could be between 0.1 and 0.3.

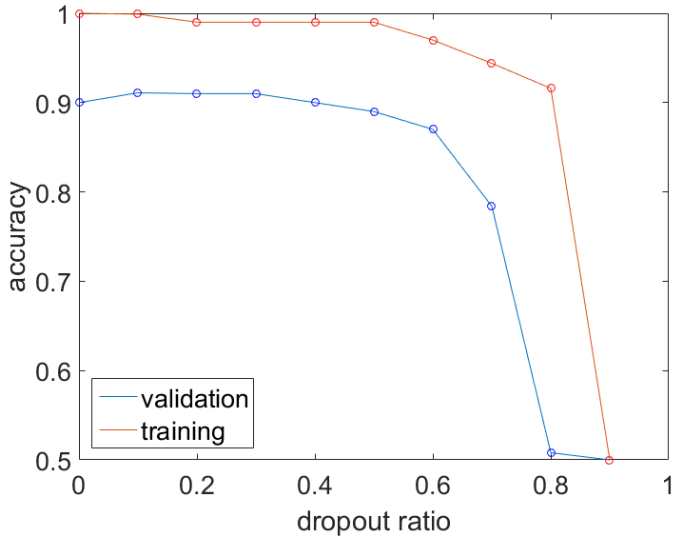

Figure 4.8: Effect of dropout. The relation between weight dropout and accuracy.

4.2.2 Dataset and network structure

Since the original data contains too many voxels, over 20 million on average, the images were down-sampled in order for the training process to finish within an acceptable time frame. Different voxel sampling strategies were used, as presented in Table 4.1, where Fore stands for foreground voxels and Back stands for background voxels. The parenthesis in D, E and F stand for the distance from the pericardium. Voxels with a distance smaller than the value specified in the parenthesis will be considered as boundary.

Table 4.1: Different data sets used in the training process

Data Set	Patch Size	Sampling Interval
A	$28 \times 28 \times 28$	Fore:{20,20,10}, Back:{25,25,15}
B	$28 \times 28 \times 28$	Fore:{10,10,4}, Back:{13,13,5}
C	$28 \times 28 \times 12$	Fore:{20,20,10}, Back:{25,25,15}
D	$28 \times 28 \times 12$	Fore:{15,15,7},Back:{20,20,10},Boundary:{10,10,5}(5)
E	$28 \times 28 \times 12$	Boundary:{4,4,1}(1), the rest:{13,13,6}
F	$28 \times 28 \times 12$	Fore:{10,10,4},Back:{13,13,6},Boundary:{7,7,3}(5)

Since the number of voxels inside and outside the pericardium are not the same, the first sampling strategy focus on creating a balanced training set. For the foreground voxels, every 20th voxel in the first and second dimension and every 10th voxel in the third dimension was selected. For the background, a slightly larger sampling interval was used: every 25th in the first and second dimension and every 15th in the third dimension.

To start with, the patch was extracted as a cube with 28 voxels as the length of each side, and the network used was the first structure in Figure 3.4. The best result for this dataset, dataset A, is 92% for the validation accuracy, as can be

seen in Figure 4.9. The hyper-parameters are: batch size 50, learning rate 10^{-3}, dropout ratio 0.3, weight decay, learning rate decay and momentum are set to 0. Although from the figure we can tell that the training accuracy is likely to continue growing, the validation accuracy however remains at a relatively fixed range. Thus continuing to train the network will only enlarge the overfitting and not learn any general information.

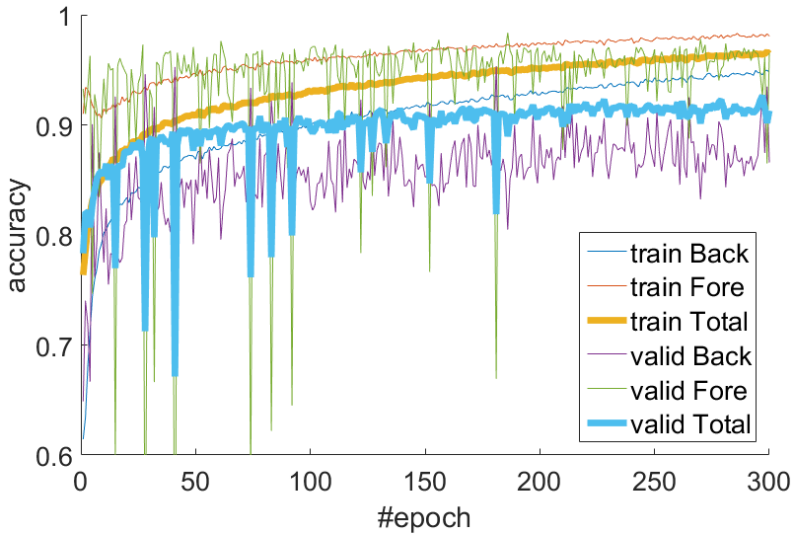

Figure 4.9: Training process of dataset A.

Using the same parameters, but including more input patches, the overfitting can be improved. When the voxel sampling interval was set to $10 \times 10 \times 4$ for foreground and $13 \times 13 \times 5$ for background the validation accuracy for dataset B increased to 94%.

Even though the validation accuracy reaches over 90% for this dataset, the cubic patch shape means that the amount of information is much richer in the third dimension than the other two since the voxel is of scales $0.3320\text{mm} \times 0.3320\text{mm} \times 1.5000\text{mm}$. In addition, such a patch size excludes all the voxels in the first and last 14 slices in the third dimension. The pericardium at the first and last several slices in the third dimension is surrounded by fat tissues and thus hard to segment, as compared to the slices where pericardium is almost the boundary of the lung.

Due to these reasons, the patch size was changed to $28 \times 28 \times 12$. In addition, the network structure is also changed to structure 2 in Figure 3.4, where a smaller filter size in the third dimension corresponds to a smaller patch size. The best validation accuracy for dataset C is 0.9. The decrease in accuracy compared to dataset A can be explained the fact that dataset C includes voxels that are harder to classify, as discussed above.

The voxels far away from the pericardium are easy to classify, while the ones near pericardium are harder to classify. The voxels and the corresponding patches that are outside but still close to the pericardium are very similar to the ones just inside the pericardium. This can be seen in Figure 4.10. While the patch surrounding voxel

3 has clearly different voxel intensities and patterns, and can be easily classified by the network, there are only minor differences between the patches extracted from voxel 1 and voxel 2. The segmentation is barely visible even for the human eye. Thus, the next adjustment is to include more voxels near the pericardium. That is, for voxels with a distance of less than 5 voxels to the pericardium segmentation, the sampling density is set to a higher value. The best validation accuracy achieved for dataset D is 84%. This validation accuracy is lower than the previous datasets because it includes more voxels that are difficult to classify. Thus to compare the result, the final segmentation must be evaluated rather than simply comparing the validation accuracy.

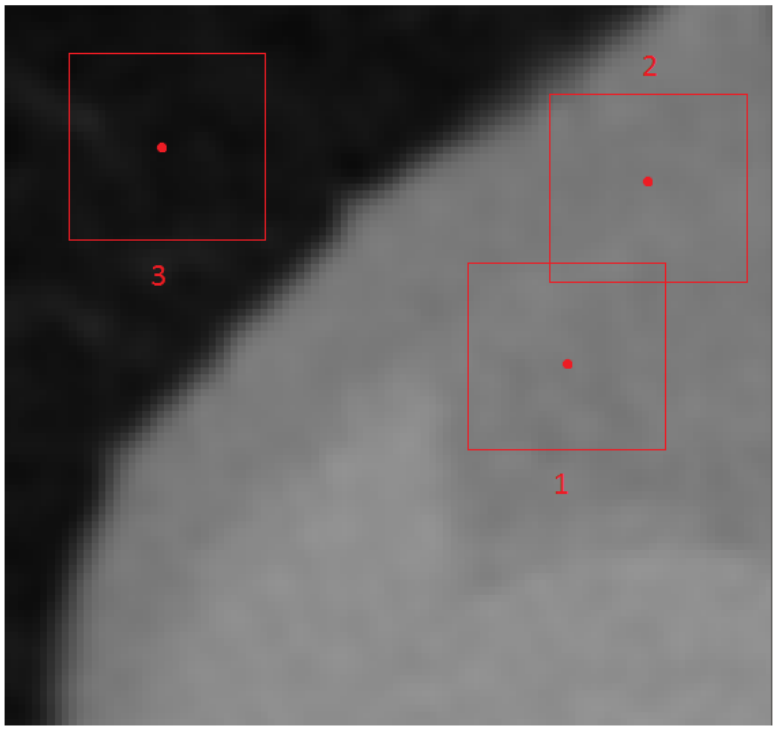

Figure 4.10: Patches inside and outside pericardium. Voxel 1 is inside and close to pericardium. Voxel 2 is outside and close to pericardium. Voxel 3 is outside and far away from pericardium. The segmentation of this part of pericardium is shown in the previous Figure 4.2.

From the results of all the training so far, the training accuracy is always able to reach 100%. This indicates that the network may have too many parameters, i.e. too many neurons. Moreover, since the voxel has a scale of $1 : 1 : 5$ in each dimension, the smaller patch size cannot make the amount of information in each dimension fully even. Therefore, by reducing the filter size in the first convolutional layer and pooling size in the max-pooling layer in the third dimension, the overfitting might be improved. The result of dataset D using network structure 3 is better than using network structure 2, with a two percent increase of the validation accuracy. A larger dataset F, similar to dataset D, yields a slightly higher validation accuracy.

Lastly, inspired by the previous work in [22], the foreground and background labels

are redefined as voxels located right *on* the segmentation boundary and the rest of the voxels respectively, in order to extract pericardium itself. To balance the training set, a multi-atlas algorithm is applied on the image before training. The best validation accuracy is 0.692. The hyper-parameters are: batch size 50, learning rate 10^{-3}, dropout ratio 0.5, weight decay 10^{-4} and 0 for learning rate decay and momentum. As a comparison with the previous validation accuracy, this one is particularly low. As a matter of fact, as Figure 4.11 shows, the validation foreground accuracy is not too much better than a random guess. One possible explanation is that since the segmentation itself is not accurate enough, using only the boundary as foreground makes the system extremely sensitive to the imperfect registration result. Therefore, the training data's generalization ability is furthermore impaired.

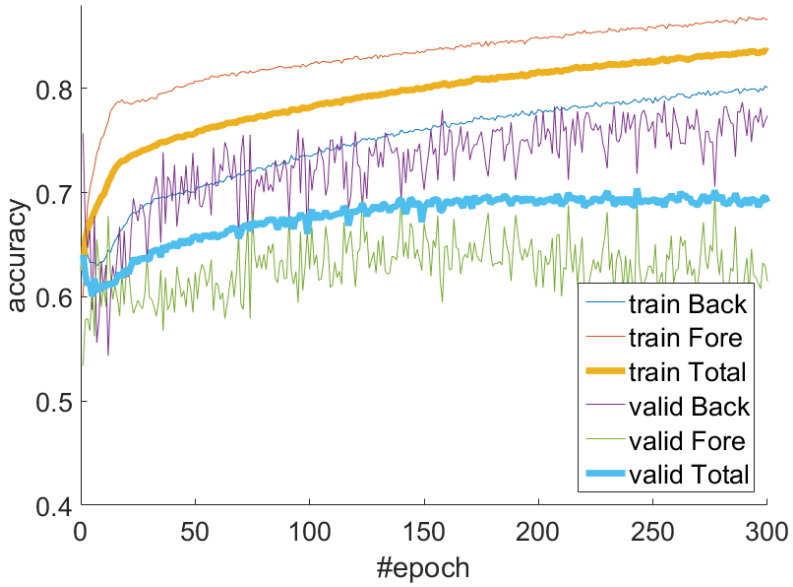

Figure 4.11: Training process of dataset E

Table 4.2 concludes the best result obtained for each data set.

Table 4.2: Best training result for each dataset. Parameters not presented are set to 0. Abbreviations: BS: batch size. LR: learning rate. LRD: learning rate decay. MOM: momentum. DO: dropout. WD: weight decay. Details of dataset can be found in Table 4.1. The network structures are shown in Figure 3.4.

Data Set (Network Structure)	Validation accuracy	hyper-parameter settings
A(1)	0.92	BS: 50, LR: 10^{-3}, DO: 0.3
B(2)	0.94	BS: 50, LR: 10^{-3}, DO: 0.3
C(2)	0.90	BS: 100, LR: 10^{-2}, WD: 10^{-2}
D(2)	0.84	BS: 50, LR: 10^{-2}, DO: 0.3
D(3)	0.86	BS: 50, LR: 10^{-3}, DO: 0.1, WD: 10^{-3}
E(3)	0.692	BS: 50, LR: 10^{-3}, DO: 0.5, WD: 10^{-4}
F(3)	0.865	BS: 50, LR: 10^{-2}, DO: 0.1, WD: 10^{-4}

4.3 Post-processing

As discussed in Section 3.3, the post-processing converts the probability map into a binary segmentation using a combination of multi-atlas and graph cuts.

When computing the final labelling, the average dice score of the validation set is presented in Figure 4.12 as a function of the post-processing hyper parameters: threshold α in Equation 3.2 and regularization term λ in Equation 3.1.

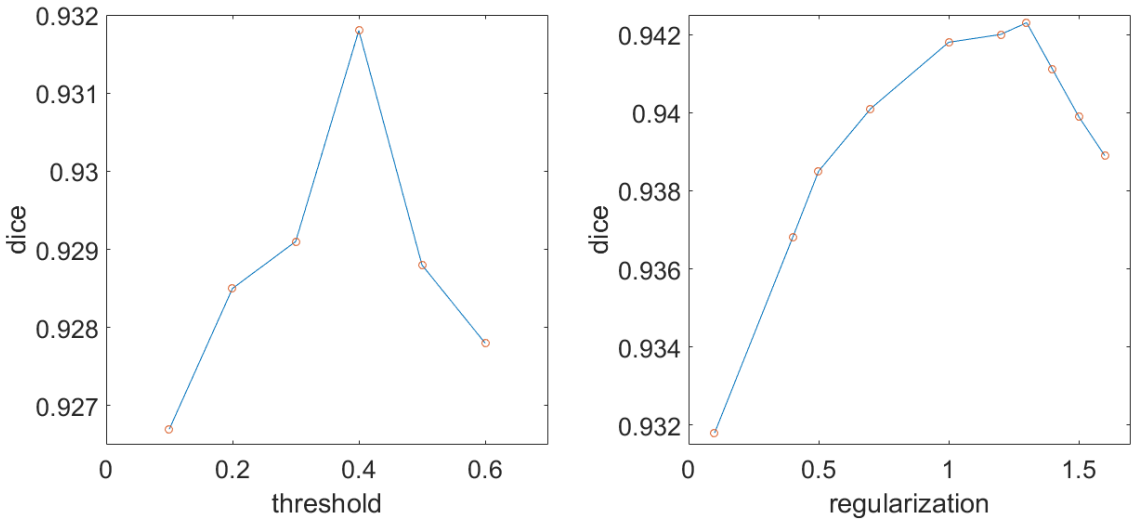

Figure 4.12: Dice score of the different hyper-parameters of post-processing. The left figure shows the effect of threshold α, and the right figure shows how the dice score changes with different regularization weight λ.

The dice score for the four test images can be found in Table 4.3, using the parameter configuration resulting in the best dice score for the validation images.

Table 4.3: Dice score of the test images.

image ID	1	2	3	4
Dice score	0.9582	0.9553	0.9563	0.9490

A closer look at one test image is shown in Figure 4.13. Comparing the top left figure and the bottom left figure, it can be seen that the pericardium is indeed detected with high probability. In the bottom left figure, the blue line at the lower part of the image goes into the black background. This indicates that the multi-atlas algorithm does not perform perfectly and excludes a part of the pericardium and the heart. Fortunately, this part of heart does not usually contain too many fat tissues. In addition, in the bottom right figure, the blue ground truth also disagrees with the red segmentation result in the upper left part of the heart. The graph cuts algorithm chooses the outer lung boundary, which is also of high probability as shown in the top left figure, instead of the inner pericardium. Unlike the lower part of the heart,

the upper part of the heart is where the fat tissues are usually located, as illustrated by the volume image at the top right.

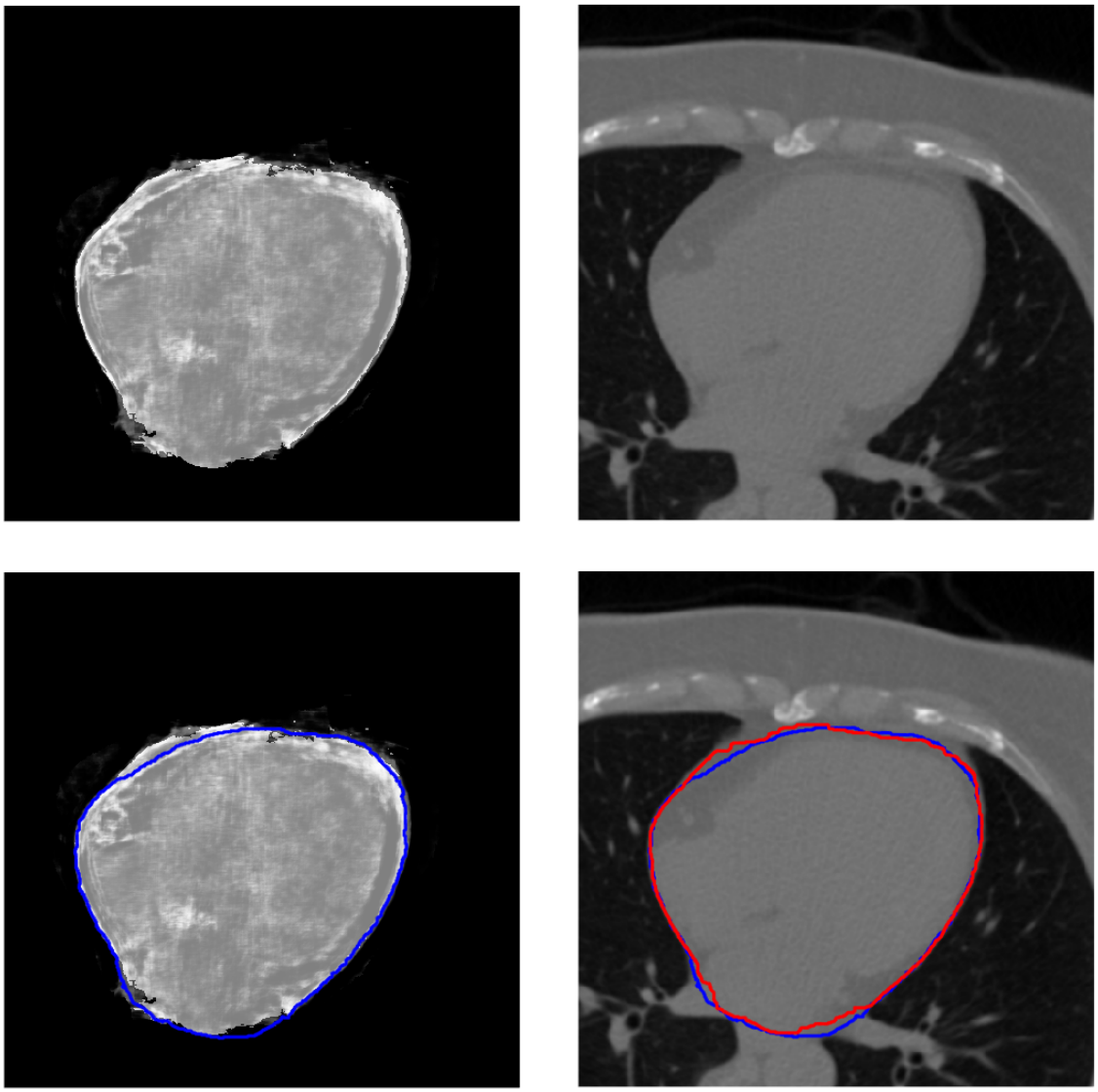

Figure 4.13: The final results on one of the four test images. The top left figure shows the probability P in Equation 3.1. The top right figure is the original non-contrast CT volume image. The bottom left shows the probability P with the ground truth marked as blue. The bottom right plots the blue ground truth and the red final segmentation results on top of the volume image.

The rest of the test images are very similar to the example in Figure 4.13. In some cases, the multi-atlas fails to include all parts of pericardium. In other cases, the graph cuts fails in choosing the correct line between the pericardium and the lung boundary.

5

Conclusion

In this thesis, an automated method to segment the pericardium structure in non-contrast CT images is presented. The method combines supervised learning with a CNN, multi-atlas and graph cuts. To create a ground truth labelling that can be used for the supervised classification, image registration, including both featured-based and intensity-based methods, was used to register manual delineations from the contrast image to the non-contrast image.

The image registration is done between one patient's contrast image and the same patient's non-contrast image. A feature-based registration method is first applied, where an affine transformation is estimated for each patient. The result is then used as an initialization for the intensity-based registration method. The final registration results are examined by a professional.

The following segmentation task is handled by deep learning. A convolutional neural network is trained by different approaches. The outcome probability maps include both inside/outside pericardium and on/off pericardium. A combination of those two types of probability maps is then processed by multi-atlas and graph cuts to produce an inside/outside pericardium segmentation. The final segmentations achieved an average dice score of 0.9547 on the test images, with minor misalignment in some regions.

6

Future Work

Since the image registration result is not perfect, especially for some key regions, it is hard to conclude how good this supervised neural network method performs. Hence, to continue working on the non-contrast CT images, the first issue to solve would be improving the accuracy of the semi-manual segmentation. In addition, to evaluate the method, and also to compare with the previous work, a network based on the contrast images should be trained.

In the convolutional neural network part, this thesis uses image patches with one certain size. Thus the classification result is entirely based on the information within the specific size. However, a larger patch size together with a parallel pipeline in the CNN's structure might help the network to learn some global information and improve the probability by eliminating the regions far away from the pericardium or even the boundary between the heart and the lung, where the post-processing method used in this thesis sometimes fails.

As discussed in Section 3.2.2, the training, validation and test set have a very limited number of images. A reasonable resolution is to use data augmentation. By increasing the variety of the training dataset, the network might yield a higher validation accuracy and better segmentation result on the test images.

Bibliography

[1] Mahajan, Kunal. “BASICS of CT Head.” *SlideShare*. Mar. 2013. Web. 30 Aug. 2016.

[2] “A Hounsfield Unit.” *Computerized Tomography Scan*. n.d. Web. 30 Aug. 2016.

[3] Gentle, James E. "Matrix transformations and factorizations." *Matrix Algebra: Theory, Computations, and Applications in Statistics* (2007): 173-200.

[4] Zöliei, L., John Fisher, and William Wells. "An introduction to statistical methods of medical image registration." *Handbook of Mathematical Models in Computer Vision*. Springer US, 2006. 531-542.

[5] Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM* 24.6 (1981): 381-395.

[6] Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60.2 (2004): 91-110.

[7] Shannon, Claude Elwood. "A mathematical theory of communication." *ACM SIGMOBILE Mobile Computing and Communications Review 5.1* (2001): 3-55.

[8] Maes, Frederik, Dirk Vandermeulen, and Paul Suetens. "Medical image registration using mutual information." *Proceedings of the IEEE* 91.10 (2003): 1699-1722.

[9] Simon, Phil. *Too Big to Ignore: The Business Case for Big Data*. Vol. 72. John Wiley & Sons, 2013.

[10] “Machine learning.” *Wikipedia*. N.p.: Wikimedia Foundation, 29 Aug. 2016. Web. 30 Aug. 2016.

[11] Rosenblatt, Frank. The perceptron, a perceiving and recognizing automaton Project Para. *Cornell Aeronautical Laboratory*, 1957.

[12] McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas imma-

nent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

[13] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." *Neural networks* 2.5 (1989): 359-366.

[14] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *Cognitive modeling* 5.3 (1988): 1.

[15] Nielsen, Michael A. "Neural Networks and Deep Learning", *Determination Press*, 2015

[16] "Unsupervised feature learning and deep learning Tutorial." n.d. Web. 30 Aug. 2016.

[17] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

[18] *Deep learning tutorials — DeepLearning 0.1 documentation*. 2008. Web. 30 Aug. 2016.

[19] Scherer, Dominik, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition." *International Conference on Artificial Neural Networks*. Springer Berlin Heidelberg, 2010.

[20] Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." *ICDAR*. Vol. 3. 2003.

[21] Karpathy, Andrej. Convolutional neural networks for visual recognition. n.d. Web. 30 Aug. 2016.

[22] Alexander Norlen, et al. "Automatic pericardium segmentation and quantification epicardial fat from computed tomography angiography". *Journal of Medical Imaging*

[23] Kahl, Fredrik, et al. "Good features for reliable registration in multi-atlas segmentation." *Proceedings of the VISCERAL Challenge at ISBI* 1390 (2015): 12-17.

[24] Jamriška, Ondřej, Daniel Sýkora, and Alexander Hornung. "Cache-efficient graph cuts on structured grids." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.

[25] Zou, Kelly H., et al. "Statistical validation of image segmentation quality based on a spatial overlap index 1: Scientific reports." *Academic radiology* 11.2 (2004): 178-189.

[26] Pluim, Josien PW, JB Antoine Maintz, and Max A. Viergever. "Mutual-information-based registration of medical images: a survey." *IEEE transactions on medical imaging* 22.8 (2003): 986-1004.

[27] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

[28] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[29] Gorter, Petra M., et al. "Relation of epicardial and pericoronary fat to coronary atherosclerosis and coronary artery calcium in patients undergoing coronary angiography." *The American journal of cardiology* 102.4 (2008): 380-385.

[30] Dey, Damini, et al. "Computer-aided non-contrast CT-based quantification of pericardial and thoracic fat and their associations with coronary calcium and metabolic syndrome." *Atherosclerosis* 209.1 (2010): 136-141.

[31] Bergström, G., et al. "The Swedish cardiopulmonary bioimage study: objectives and design." *Journal of internal medicine* 278.6 (2015): 645-659.

[32] Larsson, Måns, et al. "DeepSeg: Abdominal Organ Segmentation Using Deep Convolutional Neural Networks." *Swedish Symposium on Image Analysis 2016'*. 2016.

[33] Iglesias, Juan Eugenio, and Mert R. Sabuncu. "Multi-atlas segmentation of biomedical images: a survey." *Medical image analysis* 24.1 (2015): 205-219.