Department of Computer Science
Faculty of Engineering
The University of Hong Kong

Deep Learning Based Heart Disease Prediction

Final Report

16 April, 2022

CHOI Chong Hing 3035564940

Supervised by Dr. Wong, Kenneth K.Y.

Abstract

Heart disease is one of the leading causes of death worldwide. While early detection could prevent severe damages, modern heart examination using 2D echocardiography is time-intensive and complex. On the other hand, 3D echocardiography can scan the heart within 10 seconds quickly. However, the visualisation and interpretation of the returned 3D volumetric data remain a challenge. This project will, therefore, develop an algorithm to convert the 3D volumetric data to 2D data for interpretation using deep learning methods. Furthermore, this project will also develop an algorithm to predict heart disease from the echo inputs. The project is progressing as scheduled. We are currently working on the view classification convolutional neural network, mainly on collecting 2D echo data. The development and training for the network can begin when a sufficient amount of data is collected.

Acknowledgements

First and foremost, I would like to express my special thanks of gratitude to Dr Kenneth Wong for giving me this valuable chance to get involved in this research project. This project would not be possible without his guidance and support throughout the project.

I would like to thank Dr Loretta Choi for giving helpful advice in medical visualization and many other areas.

Also, I would like to thank our research team members Mr Justin Yum, Mr Jerry Tam and Mr Tommy Chan for their continued support and help for this project.

I would also like to acknowledge Dr Lee, and all doctors involved, who assisted our team by providing medical data and medical knowledge for the project.

Last but not least, I would like to thank my family, who has been supporting and encouraging me throughout my years of study.

Contents

List of Figures

List of Tables

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
CNN	Convolutional Neural Network
DICOM	Digital Imaging and Communications in Medicine
Echo	Echocardiography
FYP	Final Year Project
MSE	Mean Squared Error
ROI	Region of Interest
SD	Standard Deviation

Notations and Symbols

$\mathbf{A}$	Matrix representing intensity of an image
c	Classification class
$\mathbf{G}$	Gradient
H	Cross Entropy
M	Number of classes in a classification CNN
$M_{\mathrm{A}\to\mathrm{B}}$	Transfromation matrix for the transformation from Space A to B
μ	Mean value
$\boldsymbol{\mu}_i$	The mean vector of the points in cluster i (i.e. Cluster centre of cluster i)
n	Number of data in a dataset
N_s	Number of frames in the sequence
$\vec{P}$	A point (landmark) P in landmark detection
$\vec{P}_{i_j}$	The j-th landmark of the i-th data in the dataset
$\vec{P'}_{i_j}$	The predicted j-th landmark of the i-th data in the dataset
$\mathbf{S}$	The set of cluster sets
$\mathbf{S}_i$	The set of points in cluster i
σ	Standard deviation
Θ	Gradient angle
$\mathbf{x}$	A data point in clustering analysis
x_i	Value of the point at time instance i
y_c	Boolean value for data y belongs to class c
$\hat{y}_c$	Model predicted value for data y in class c

1 Introduction

1.1 Overview

Heart disease is one of the leading causes of death both locally and globally. It accounts for over 17.9 million deaths globally [1], and 77 thousand deaths in Hong Kong each year [2]. A large number of patients causes high demand for medical resources, including heart examination, hospitalisation and treatments. Early detection of heart disease could prevent patients from having severe damage. Therefore, an early heart examination is crucial in lowering the risk of death.

1.2 Modern Heart Examination

Modern heart examination captures 27 different cross-sections and some other views of the heart using 2D echocardiography. Unlike coronary angiogram, where a special dye is injected into the vein to capture the heart images, the echocardiogram method is non-invasive, making it a very common heart test. During an echocardiogram test, an ultrasound device sends out an ultrasound signal to the heart and receives the reflected signal to reconstruct the 2D heart image.

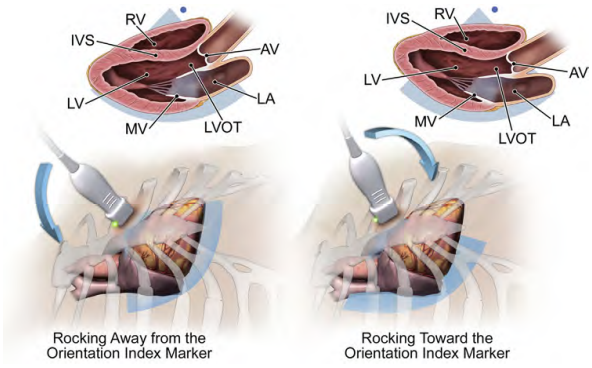

Figure 1.1: An illustration of an echocardiogram examination [3]

However, as illustrated in Figure 1.1, there are bones between the device and the heart, which will block the ultrasound signals. Therefore, echo examination is very difficult and time-consuming, and could only be performed by trained specialists. The trained doctors have to control the device accurately so that the signals could travel through the gaps between the bones to obtain the correct heart slice views. One complete echo examination could take up to 45 minutes or more. This lengthy process causes a long waiting time of an average of 81 weeks for an echo examination in Hong Kong public hospitals, delaying the diagnosis and treatment of patients. To reduce the examination time, 3D echocardiography is brought as a solution.

1.3 3D Echocardiography

Similar to 2D echocardiography (2D echo), 3D echocardiography (3D echo) uses a simailar technique to obtain heart information.

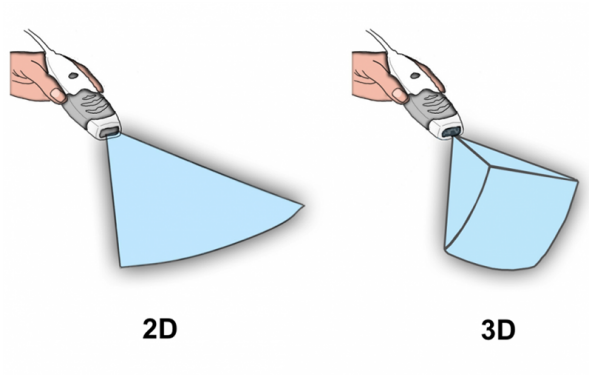

Figure 1.2: 2D echography vs 3D echography [4]

As illustrated in Figure 1.2, a 3D echo device shoots an array of ultrasound signals, whereas a 2D echo device shoots only one ultrasound signal. Using 3D echocardiography, a volumetric 3D representation data of the heart could be obtained in under 10 seconds. The specialists do not need to manoeuvre the device to capture views using the 3D echo device.

However, the specialists are trained to interpret the 2D cross-section views of the heart only. The volumetric data captured by a 3D echo device is difficult to interpret. The visualization of volumetric data is non-trivial either. To solve this problem, our project proposes a solution using a 3D landmark detection algorithm.

1.4 3D Landmark Detection

A 3D landmark detection algorithm can find specific landmarks (points) in a 3D space. Having three points in 3D space, a 2D plane could be defined in that 3D space. Therefore, identifying three points for each view from the 3D data, the 27 standard views could then be reconstructed. This type of algorithm is typically constructed using a convolutional neural network (CNN) using deep learning technique, like Loc-Net [5], an CNN that identify the position of the bifurcation of the carotid arteries. These networks comprise multiple layers so that the higher-level features could be extracted from the raw data input, giving the landmarks needed. The 2D views can then be reconstructed using the 3D data and landmarks.

1.5 Heart Disease Prediction

With the 27 standard 2D views, doctors could decide if the heart is healthy or not. To further facilitate the examination process, our project proposes another CNN model to predict heart disease using the 27 views as inputs. Such CNN model is similar to the 3D landmark detection CNN described above, except the heart disease prediction CNN outputs the probability that there exist heart disease. A

higher output value means that the heart disease prediction CNN is confident that there are some problems with the heart, vice versa.

1.6 Objectives

In this project, we aim to implement two convolutional neural networks to reduce the examination time. The first CNN will be the 3D landmark detection algorithm, where the landmarks for the 27 views could be extracted so that the views could be reconstructed. With the volumetric data and the reconstructed 2D views, the second CNN will be trained to predict heart disease directly.

1.7 Project Contribution

With the increasing demand for heart examinations and insufficient number of available specialists, it is anticipated that the waiting time for an examination will increase accordingly. This project will change the method of heart examination completely, in a more expeditious and efficacious way. It is hoped that this project will reduce heart examination time from 45 minutes to less than 3 minutes, and, consequently, significantly reduce the waiting time from 81 weeks to less than 10 weeks.

1.8 Report Outline

This progress report is structured into five chapters. The first chapter offers a brief overview of heart examination and the background of the project. It also gives the objectives and the significance of the project.

Chapter two analyses the methodology of the project. Convolutional Neural Network is explained in this chapter. The two CNNs, 3D landmark detection CNN and heart disease prediction CNN, are also explained. This includes the collection of datasets, the training methods, and the validation process. A brief architecture of the 3D landmark detection CNN will also be shown and discussed.

Chapter three shows the current progress of the project. It also presents the project schedule. Future plans for the project outside the scope of this Final Year Project (FYP) are discussed in this chapter.

Chapter four explains some results in this FYP. Some proposed solutions to the problems are explained in this chapter.

Chapter five concludes the report, summing up all the results made and the coming plan for the project.

2 Methodology

2.1 Introduction

This chapter explains the methods used in training and testing the two CNNs in this project. It also presents the data collection methods, as well as data analysis methods including data augmentation and processing. Image processing methods like Canny Edge Detection and Hough Line Transform are also discussed in this chapter. Figure 2.1 shows the overview of the methodology for the project. The blue part corresponds to the first objective - Landmark Detection; while the red part corresponds to the second objective - Heart Disease Prediction. Details will be explained in the following subsections.

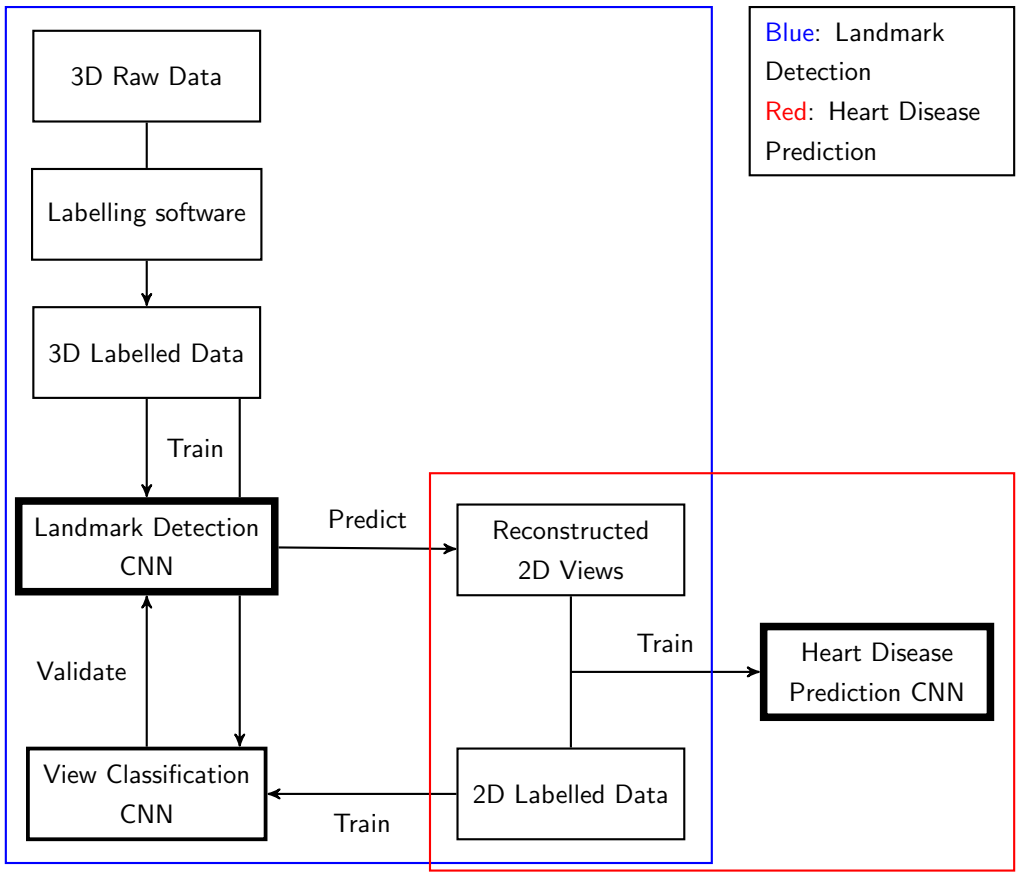

Figure 2.1: Overview of Methodology

2.2 Convolutional Neural Network

The convolutional neural network is a common artificial neural network that analyses 2D and higher-dimensional data. A typical convolutional neural network consists of two parts - feature extraction and prediction. The feature extraction part extracts different features of the data, providing useful information of the image to the later stage. The second part of the network takes those feature information to do specific predictions, like a typical artificial neural network. The prediction can be classifying different objects, locating different objects, detecting certain landmarks, etc. Figure 2.2 shows an overview of a classification convolutional neural network.

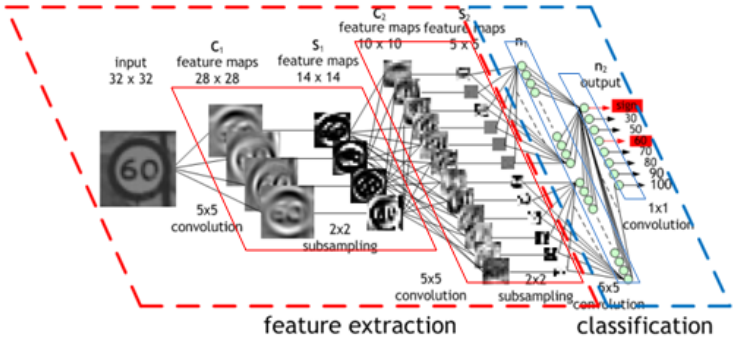

Figure 2.2: Classification Convolutional Neural Network

2.2.1 Convolutional Layer

A typical artificial neural network, like perceptrons, is usually constructed with fully connected layers, where every neuron at layer i is connected to every neuron at layer $i+1$ with a parameter. A convolutional neural network, however, uses convolutional layers, where a filter or kernel is convolved across the input volume to create the output using dot product.

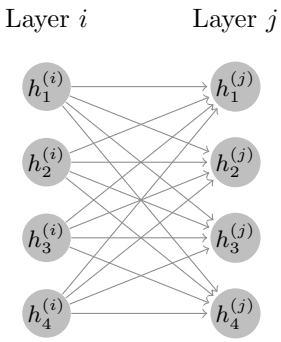

Figure 2.3: A Fully Connected/Dense Layer

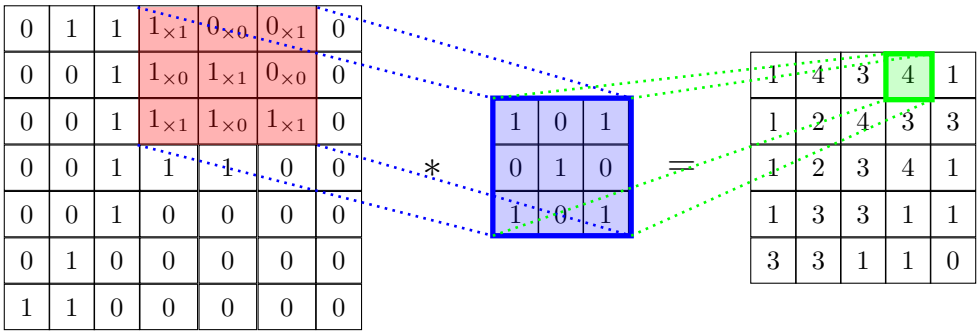

Figure 2.4: A 2D Convolutional Layer

Figure 2.3 shows an example of a fully connected (dense) layer, where 4 input neurons are connected to 4 output neurons. This, in total, has 16 parameters to train in this layer. On the other hand, Figure 2.4 shows an example of a 2D convolutional layer with one kernel, where a 7x7 input is convolved with a 3x3 kernel to form a 5x5 output. This has only 9 parameters in this layer. One could discover that a convolutional layer could process more inputs with fewer parameters than a fully connected layer. This gives a huge advantage to the convolutional neural network, in which the inputs are typically 2D or higher-dimensional data.

2.2.2 Feature Extraction

Convolutional layers can extract different features from data. Long before its application in deep learning, convolutional operations were widely used to apply filters to the image, like smoothing, sharpening and blurring, and extracting image features, including corners, edges and blobs. The kernels applied are manually adjusted for specific purposes.

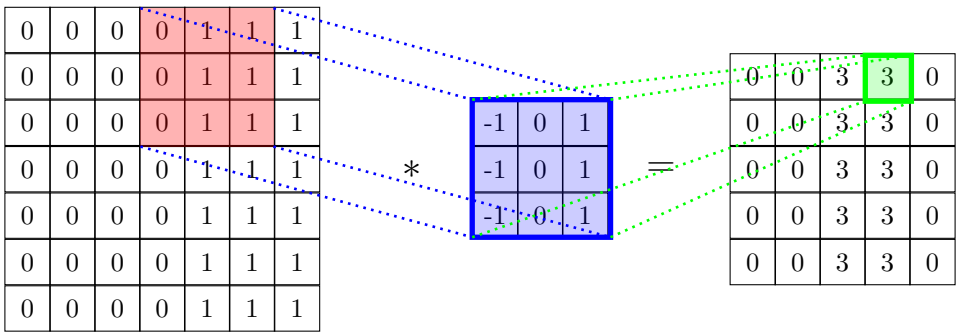

Figure 2.5: Vertical Edge Detection using Convolution

Figure 2.5 shows a vertical edge detection using convolution. There is a vertical edge at the 4th and 5th columns in the input. By convolving the input with the specified kernel, the output specifies the positions where vertical edges exist.

In contrast to the manually adjusted kernel in image processing, convolutions in the convolutional

neural network are not specified and adjusted by man. Instead, parameters in kernels constantly evolve in the training phase. The parameters in kernels are adjusted by error, which is determined by the difference between model prediction and correct output. In other words, the adjustments of the parameters are purely result-orientated. Therefore, unlike the edge detection example, the feature extractions in the network are often not understood by us.

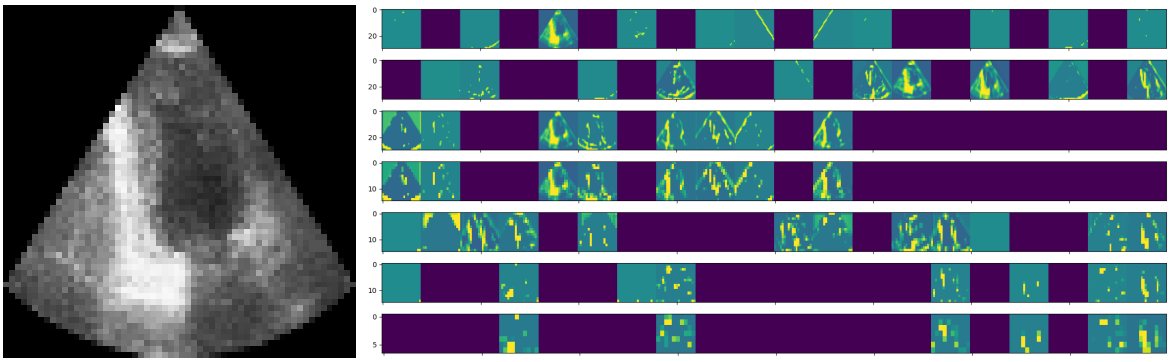

Figure 2.6: Feature Extraction in a Convolutional Neural Network

Figure 2.6 shows the feature extractions of a view classification CNN. The image on the left is the input image, while the table on the right shows the feature extractions of the input image in different layers. The first row corresponds to feature extractions in the first convolutional layer, the second row corresponds to the second layer and so on. Each small image corresponds to one kernel in that layer. The brightened parts represent the processing parts of the kernels in the layer. In general, the first few layers extract low-level features, including edges and corners. As the image is passed to back layers, high-level features are extracted, like segmentations of the heart chamber, which are nearly impossible to be extracted using human-adjusted kernels. With the high-level feature extracted, the convolutional neural network processes the feature data like a normal neural network using fully connected layers, giving the predictions.

2.3 Training Categories of CNN

In training a CNN model, there are two major training categories: supervised learning and unsupervised learning. Supervised learning utilises differences between the predictions from the model and the correct labels of the data to adjust the model using differential methods. Unsupervised learning, on the other hand, uses other metrics, like different clustering methods, to adjust the model. There also exists semi-supervised learning, where mixes the two training methods. In this project, different state-of-art learning methods will be examined and recreated.

2.3.1 Supervised Learning

Supervised learning computes loss function during training phrase. The loss function describes the error of the prediction, that is the difference between the model's prediction and the correct label. Many different loss functions suit different circumstances.

For example, for a classification model, the cross-entropy loss is a common choice for training the model. Cross-entropy loss is given by:

$$H = -\sum_{c=1}^{M}(y_c \log \hat{y}_c)$$

where M is the number of classes
y_c is the boolean value of y being class c
$\hat{y}_c$ is the predicted value of y in being class c

As there is only one true class for each data, there is only one c such that y_c is 1 for each data. Thus, only one term of the whole summation is valid, where y_c equals 1. Then, the magnitude of the cross-entropy loss is purely determined by the predicted value of the data in its true class, i.e. $\hat{y}_c$ where y_c is 1.

The goal of training a supervised learning model is to minimize its loss, which means better prediction. The parameters in the model are then adjusted by the resulting loss function value, each by their contribution to the result using differential methods. This results in a better model with lower loss and better prediction when predicting the same data.

The true class term y_c in the loss functions implies the need for labelled data in supervised learning. The model can only learn when the data are labelled. This creates a problem if labelling is a complex or difficult task. However, as the model is trained using labelled data, the model usually can obtain high accuracy, compared to unsupervised learning.

2.3.2 Unsupervised Learning

In contrast to supervised learning, unsupervised learning does not compute loss functions to adjust its model. Instead, it adjusts the model through cluster analysis. In classification, data in the same class have a certain similarity in different features. Clustering can groups data with similar features together such that classification can be done.

Take K-means Clustering, a common clustering method, as an example. The goal of K-means Clustering is given by:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathbf{S}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

where $\mathbf{S}$ is the set of all cluster sets
$\mathbf{S}_i$ is the set of points in clutser i
$\boldsymbol{\mu}_i$ is the cluster centre of cluster i

The K-means Clustering finds an equilibrium where the distances between every data point to its cluster centre are minimum. The algorithm first chooses k points as the seeds of the clusters. Other points are then classified to the closest cluster. The cluster is then recomputed, and the process is repeated until no further changes. In the final result, every point is classified to the closest clusters.

These clustering methods do not use the labels of data to do clustering; thus, unsupervised learning does not need any labelled data. This is a great advantage compared to supervised learning, especially when labelling is a difficult task. However, unsupervised learning often has lower accuracy, compared to supervised learning, as no labelled data is used in training. Moreover, finding a suitable clustering method for the different datasets is also needed.

2.3.3 Semi-supervised Learning

Apart from the two extremes, semi-supervised Learning, as the name suggests, sits between supervised learning and unsupervised learning. Semi-supervised learning methods train the network with little labelled data using, perhaps, a mixture of supervised and unsupervised learning methods. In general, semi-supervised learning tries to achieve high accuracy like supervised learning while using less labelled data. There exist many interesting and creative semi-supervised learning methods.

2.4 Landmark Detection CNN

The 3D landmark detection CNN takes the 3D raw data as input and outputs the landmarks for the 27 views. The network processes a huge amount of data, so the size of the network is often very deep, resulting in a long training time. To overcome the huge size of the network, a combination of different deep learning architectures will be used to build the network, including U-Net and multi-scale shift networks. Previous research has shown that such a multi-scale network can reduce the size of the network, and consequently, reduce the training time for the network [5].

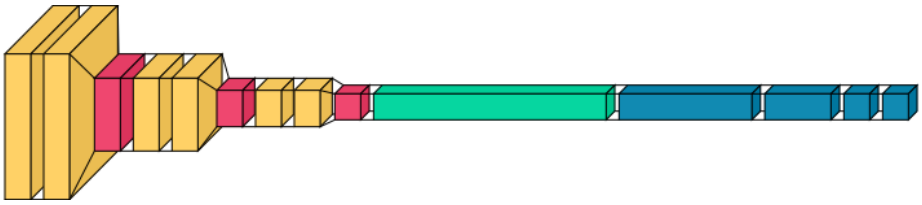

Figure 2.7: Model Architecture

The model architecture of the landmark detection CNN would be similar to Figure 2.7, where the data goes through the CNN from left to right. The architecture contains several blocks of 3D convolutional layers (coloured in yellow), separated by max-pooling layers (coloured in red). The data is then flattened (coloured in green) and passed to dense layers (coloured in blue). Finally, the last dense layer outputs the predictions of the landmarks.

The landmark detection CNN will be trained using 3D raw data of different patients, where the landmarks of the views are labelled. The model will adjust itself by the distance differences between the predicted and labelled points.

2.4.1 Landmark Detection CNN: Dataset

The 3D raw data will be provided by the Chinese University of Hong Kong, which contains the unlabeled 3D echo data from local patients in the past 10 years. To obtain the labelled 3D data, software with an easy-to-use graphical user interface for labelling data will be developed. This software will be developed as an extension of 3D Slicer [6], a free, open-source software widely used in the medical industry. 500 - 2000 patient data is planned to be labelled for training and testing the model.

2.4.2 Landmark Detection CNN: Validation

There are two main methods to validate the performance of the landmark detection CNN model. First, a labeled test dataset is passed to the trained model. Then, the mean squared error (MSE) of the position of the landmarks is calculated, which is given by:

$$\text{MSE} = \frac{1}{81n} \sum_{i=1}^{n} \sum_{j=1}^{81} \left\| \vec{P}_{i_j} - \vec{P'}_{i_j} \right\|^2$$

where n is the number of data in a dataset
$\vec{P}_{i_j}$ is the j-th landmark of the i-th data in the dataset
$\vec{P'}_{i_j}$ is the predicted j-th landmark of the i-th data in the dataset

MSE calculates and summarises the differences between the predicted landmarks and labeled landmarks. It takes the average squared distance difference of the points (each data has 81 points). A high MSE means that the differences between the predicted and labeled landmarks are high, impling the model has a poor performance, vice versa.

Other than using 3D labelled data to validate the model, the model could also be tested with a 2D view classification CNN. A view classification CNN is a model that could classify different views by checking the similarity between the input image and the views. A view with a higher score represents that the network has high confidence that the input image is of that view. This could determine how good is the landmark detection network by giving scores to the reconstructed views.

2.5 Heart Disease Prediction CNN

The heart disease prediction CNN takes the patient 2D views data, and outputs the possibility of the patient having a heart disease. Similar to the landmark detection CNN, this network processes a large amount of data. Therefore, different architectures will be implemented, trained and tested. The training and testing datasets include the 2D labelled data and the reconstructed 2D data from the prediction of landmark detection CNN. Data augmentations, including flipping, rotation, and scaling, will also be applied to enlarge the dataset.

2.6 Canny Edge Detection

Canny Edge Detection is a popular multi-stage edge detection algorithm. It first applies noise reduction to the image with a Gaussian filter. Then, the smoothened image is filtered with the Sobel kernel. Edges are then detected using non-maximum suppression. Finally, edges with low intensity and pixel noises are discarded in hysteresis thresholding. This section will explain each step involved in Canny Edge Detection.

2.6.1 Gaussian Blur

Gaussian blur is a very common convolution operation that convolves an image with a Gaussian function. This produces a blurred image as a result. The Gaussian function is given by:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

where σ is the standard deviation of the Gaussian distribution

A larger σ results in a blurrier image. During the blurring, it smoothens the edges and bumps, removing the noise in the image. This prevents detecting very details of the bumps at the edges or false detection caused by noises during the edge detection.

2.6.2 Sobel Kernel

Sobel kernel is a discrete differential operator that can create an image with emphasised edges by computing the gradient of the image in horizontal and vertical directions. The formulae of the gradients are given by:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \qquad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

where $\mathbf{A}$ is the matrix representing an input image

The directional gradients are then used to calculate the total gradient and gradient angle of each point, which are given by:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \qquad \Theta = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

where $\mathbf{G}$ is the gradient of the image
Θ is the gradient angle

The gradient angle is rounded to one of the four angles: vertical, horizontal, and two diagonals. Then, the gradient and gradient angle of each point are used to extract edges via Non-maximum Suppression.

2.6.3 Non-maximum Suppression

Non-maximum Suppression, as the name suggests, suppresses points that are not local maxima so that only points at the edges remain. Figure 2.8 shows an illustration of non-maximum suppression.

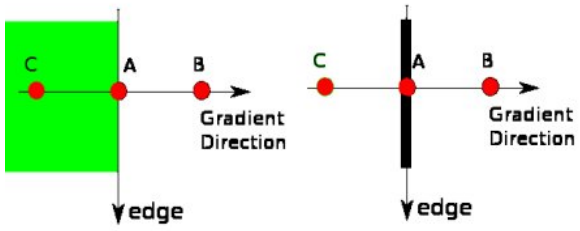

Figure 2.8: Non-maximum Suppression in Canny Edge Detection [7]

In both cases, point A is on the edge where the gradient direction is horizontal. Non-maximum suppression checks the gradient values of the nearby points along the gradient direction. In both cases, point B and C have a low gradient, while point A has a high gradient as the intensity changes at point A. Point A, in this case, forms a local maximum with points B and C; thus, point A is considered an edge. This checking is repeated with every pixel. Points that do not form local maxima will be suppressed to zero. This leaves only the points at the edges in the image.

This resulting image is then passed to be applied hysteresis thresholding.

2.6.4 Hysterssis Thresholding

Hysteresis thresholding filters insignificant edges and noises out from the edges detected from the last step. In hysteresis thresholding, two threshold values are set: maxVal and minVal. Edges with an intensity well above maxVal are considered as edges, while edges with an intensity well below minVal are not considered as edges and discarded. For edges with intensity in between, there are two possible cases. Figure 2.9 shows the two cases.

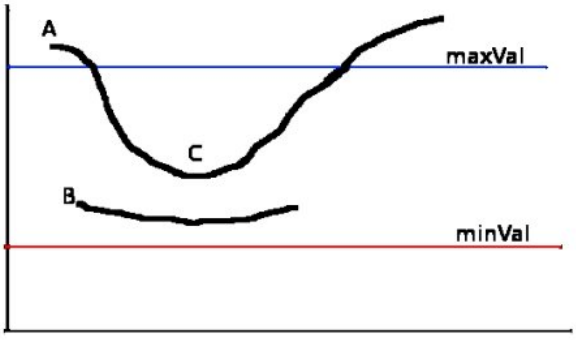

Figure 2.9: Hysterssis Thresholding in Canny Edge Detection [7]

In Figure 2.9, edge A is above the maxVal, so it is considered to be an edge. The intensity of edge

C is between the two threshold values, but it is connected to edge A, which is considered as an edge. Therefore, edge C is also counted as an edge. On the other hand, edge B also sits between the two values but is not connected to any edges above the maxVal. So, edge B is discarded.

In this step, it removes any small pixels noises in the image so that only long edges with high intensity are retained.

2.7 Hough Line Transform

Hough Line Transform is a transform used to detect straight lines. It is commonly used after applying edge detection to the image. This section will explain the hough line transform in detail.

2.7.1 Hough Line Transform in c-m Space

In Cartesian coordinate system, the equation of a line is given by:

$$y = mx + c$$

where m is the gradient/slope of the line
c is the y-intercept

Rearranging the straight line equation, we have

$$c = xm - y$$

For every point (x_0, y_0) that lie on the same line, it must satisfy the equation with the same m and c. This implies all the points on the same line in xy coordinate intersect in mc coordinate at (m, c). Figure 2.10 shows the transformation between the x-y and c-m space.

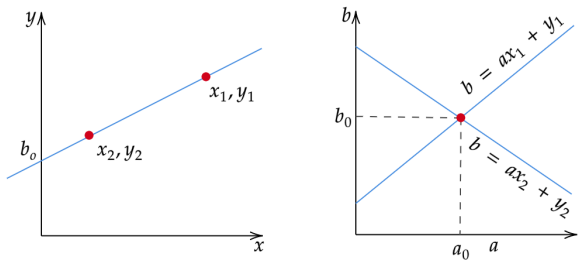

Figure 2.10: Hough Line Transform in c-m Space [8]

In Figure 2.10, the straight line equation is given by $y = a_0x + b_0$. The points (x_1, y_1) and (x_2, y_2) lie on the line. Transforming the two points into cm space, they intersect at (a_0, b_0)

2.7.2 Hough Line Transform in ρ-θ Space

An alternative representation of the Hough Line Transform uses polar coordinates. In Figure 2.11, it shows the transformation between the x-y and ρ-θ space. The graph on the left shows an alternative to writing the straight equation:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{\rho}{\sin\theta}\right)$$

$$\rho = x\cos\theta + y\sin\theta$$

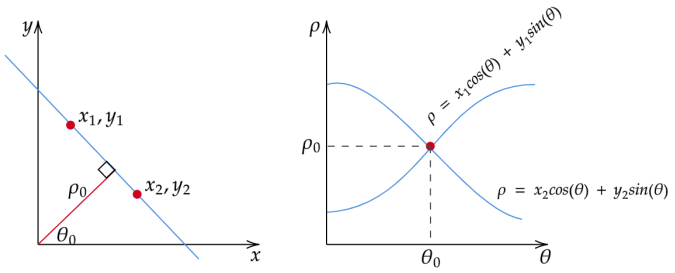

Figure 2.11: Hough Line Transform in ρ-θ Space [8]

In Figure 2.11, both points (x_1, y_1) and (x_2, y_2) lie on the line, meaning they satisfy the same straight line equation with $\theta = \theta_0$ and $\rho = \rho_0$. Therefore, the representations of the points in ρ-θ space intersect.

2.7.3 Line Detection

Hough Line Transform transforms points to another space. If the points lie on the same line, they intersect in Hough Space. Hough Line Transform set a threshold value, outputting a line when the number of intersections exceeds the threshold. The line can then be reproduced using the parameters in Hough Space. Figure 2.12 shows a complete visualisation of Line Detection using Hough Line Transform.

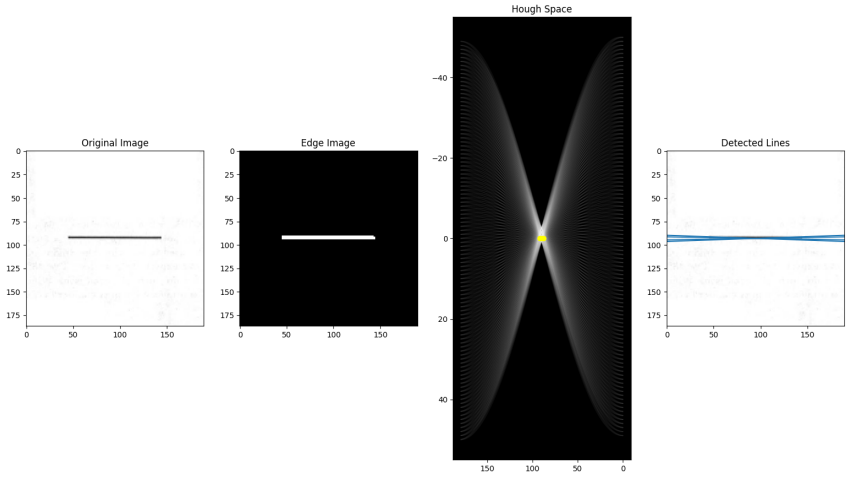

Figure 2.12: Visualisation of Line Detection using Hough Line Transform [8]

2.8 Linear Transformations

A linear transformation, or a linear map, is a mapping between two vector spaces $V_1 \rightarrow V_2$ where addition and multiplication of vectors are preserved. Namely, a function $f : V_1 \rightarrow V_2$ is a linear transformation iff

$$f(\mathbf{a} + \mathbf{b}) = f(\mathbf{a}) + f(\mathbf{b})$$
$$f(c\mathbf{a}) = cf(\mathbf{a})$$

where $\mathbf{a}$ & $\mathbf{b}$ are any two vectors
c is a scalar constant

Any linear transformation function f can be represented by a transformation matrix.

2.8.1 Transformation Matrix

For linear transformation $T : V^n \rightarrow V^m$,

$$T(\mathbf{v}) = A\mathbf{v}$$

where $\mathbf{v}$ is a vector in $\mathbb{R}^n$
A is the transformation matrix with m rows and n columns

In this project, linear transformations in 2D and 3D are applied in different parts.

2.8.2 2D Linear Transformation

In 2D linear transformation, translation, stretching (scaling), rotation, shearing, reflection and flipping are some of the common linear transformations.

Translation

The translation of the vector space could be represented by a 3x3 homogeneous transformation matrix

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

where t_x is the translation in x axis
t_y is the translation in y axis

Stretching / Scaling

The stretching of vectors in a xy-plane could be represented by a 3x3 homogeneous transformation matrix

$$\begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where k_x is the scaling factor in x axis
k_y is the scaling factor in y axis

Rotation

A counterclockwise rotation about the origin by θ has the functional form

$$\begin{cases} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{cases}$$

where (x', y') are the new position of the point (x, y)

This could be represented by the 3x3 homogeneous transformation matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing

A shearing transformation could be represented by a 3x3 homogeneous transformation matrix. A shearing parallel to x axis could be represented by

$$\begin{bmatrix} 1 & k_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where k_y is the shearing factor by y axis

A shearing parallel to y axis could be represented by

$$\begin{bmatrix} 1 & 0 & 0 \\ k_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where k_x is the shearing factor by x axis

Flipping

Flipping of an axis could be represented by a 3x3 homogeneous transformation matrix. Flipping the x axis could be represented by

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fllipping the y axis could be represented by

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

General Transformation

A general transformation in 2D space could be represented by

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix}$$

2.8.3 3D Linear Transformation

2D Linear Transformations could be extended by introducing an extra dimension. A similar transformation matrix in size of 4x4 could represent 3D Linear Transformation. The general transformation matrix of 3D Linear Transformation is given by

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.8.4 Coordinate System Transformation

In this project, coordinate system transformation is heavily used in reconstructing the views, and data processing. A coordinate system transformation is a linear space transformation. A transformation $T : V_A \to V_B$ transforms from coordinate system A to coordinate system B. The transformation function is given by

$$\mathbf{v}_B = M_{A\to B}\mathbf{v}_A$$

where	$M_{A\to B}$	is the transformation matrix of the transformation from system A to B
	$\mathbf{v}_A$	is a vector in system A
	$\mathbf{v}_B$	is the vector $\mathbf{v}_A$ in system B

The transformation function could include multiple transformation matrices. The function of the same coordinate system transformation T could also be

$$\mathbf{v}_B = M_{V\to B}M_{A\to V}\mathbf{v}_A$$

where	$M_{A\to V}$	is the transformation matrix of the transformation from system A to V
	$M_{V\to B}$	is the transformation matrix of the transformation from system V to B
	$\mathbf{v}_A$	is a vector in system A
	$\mathbf{v}_B$	is the vector $\mathbf{v}_A$ in system B

This shows an example of multiple transformations. The transformation function first transforms the system from A to V. Then, it is transformed from V to B.

2.9 Summary

This chapter explained the methodology used in the project. The CNN models used were presented, together with the training and validation processes. The data collection and processing methods for different types of data were also explained. The next chapter will present the project schedule.

3 Project Schedule

3.1 Overview

This chapter presents the project schedule, together with the current progress of the project. The project schedule is presented in Section 3.2, while Section 3.3 explains the progress of the project.

3.2 Project Schedule

This project is a 3-year project, commencing in May 2021. Table 3.1 shows the amended tentative schedule for the project.

Table 3.1: Tentative Project Schedule

Timeline	Task	Status
May 21 - Jun 21	Literature Review	Completed
Jul 21 - Sep 21	Labelling Software	Completed
Oct 21 - May 22	View Classification CNN	In Progress
Oct 21 - After May 22	Labelling Landmarks	In Progress
After May 22	Landmark Detection CNN	Pending
After May 22	Build Reconstructed 2D View Dataset	In Progress
After May 22	Heart Disease Prediction CNN	Pending
After May 22	Build Clinical Software	Pending

For the final year project, we will focus on implementing the view classification CNN, while the doctors will label the 3D data. After labelling the data, we may start implementing the landmark detection CNN. Other parts of the project, including building the heart disease prediction CNN and clinical software, are not within the scope of this final year project.

3.3 Current Progress

Currently, the development of labelling software is complete. The labelling task and collection of 2D data have also begun. Figure 3.1 above shows the status of different components in the project. The current progress is satisfactory.

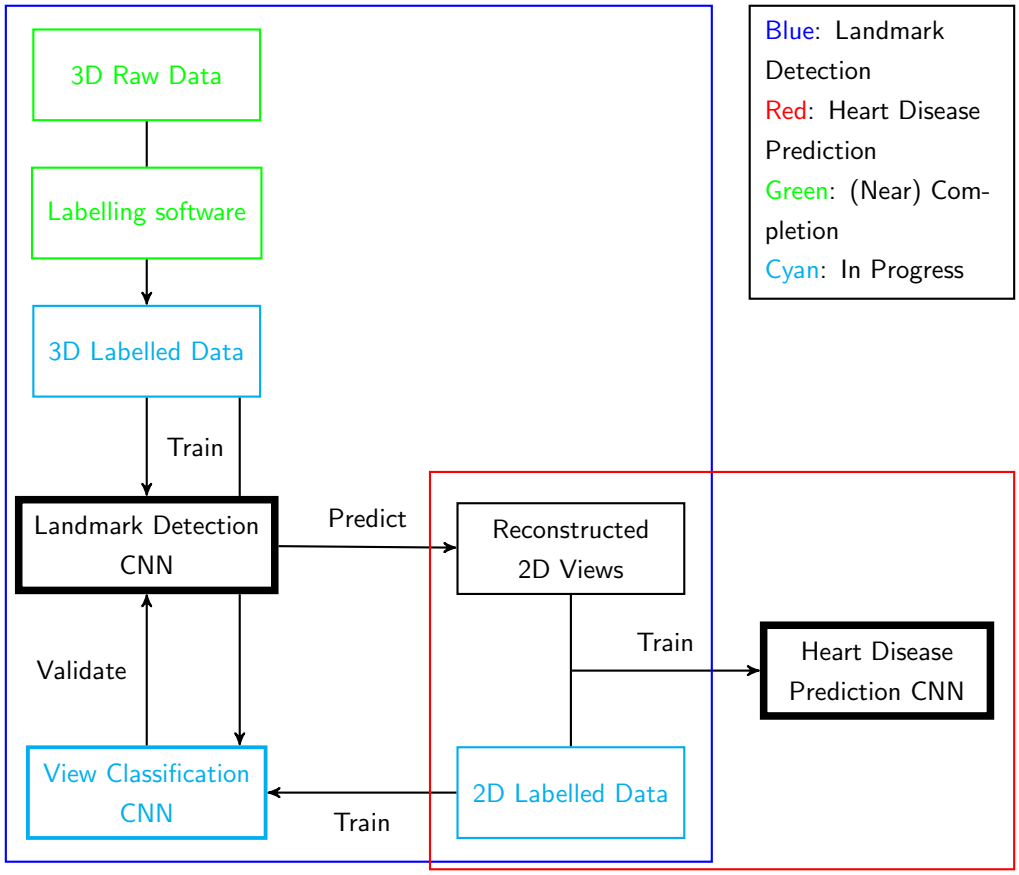

Figure 3.1: Overview of Current Progress

4 Project Results

4.1 Overview

This chapter presents the current results in developing the view classification CNN and preparing developments in later stages. It mainly includes the 2D raw echo data processing, proposing solution for view classification CNN, reconstructing 2D views from 3D data and 3D raw echo data processing, which are explained in Section 4.2, 4.3, 4.4 and 4.5 respectively. Challenges and proposed solutions are explained in detail.

4.2 2D Data Preprocessing for View Classification CNN

To train the View Classification CNN, classified 2D datasets are required. In this project, unlabelled 2D echo images have been collected. The highlighted component in the overview shows its corresponding related parts in the project.

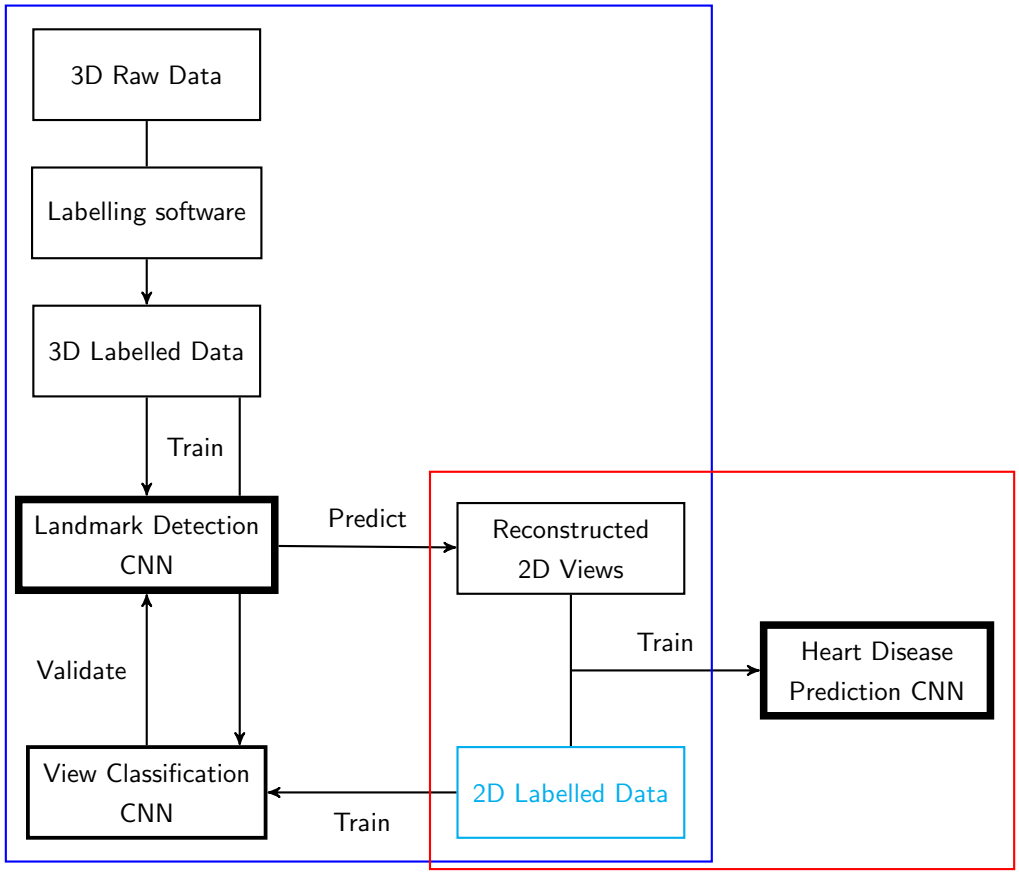

Figure 4.1: 2D Data Processing for View Classification CNN in Project Overview

4.2.1 Overview

The collected 2D data images contain a few problems. The images come in Digital Imaging and Communications in Medicine (DICOM) file format, where 3D data are mixed with the 2D ones. The images contain overlays, which affects the network training performance. It contains some images with multiple views and colour measurements, which are not useful and should be discarded. Most importantly, the images are unlabelled.

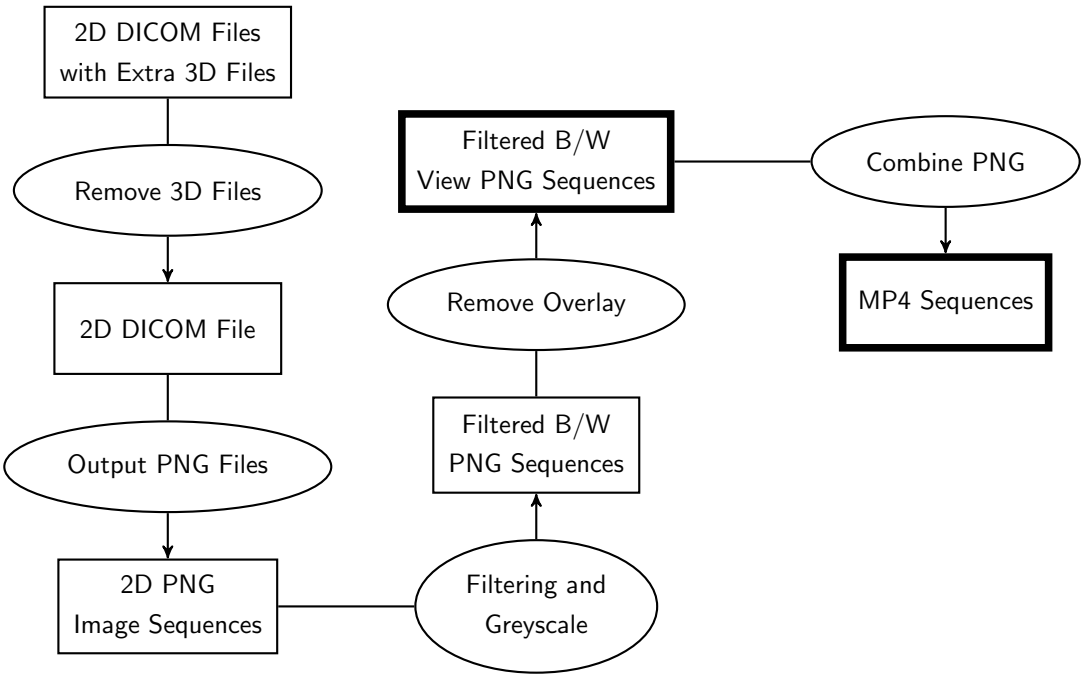

Figure 4.2: 2D Data Processing Overview

Figure 4.2 shows the overview of the 2D Data processing workflow in this project. The 2D DICOM files are first removed, and the remaining 2D DICOM files are exported to PNG Image Sequences. The sequences are filtered and transformed to greyscale. The filtering includes multi-view filtering and colour filtering, which are explained in Section 4.2.2 and 4.2.3. Then, the overlay is removed from the sequences, which is explained in Section 4.2.4. The overlay-free sequences could be used to train view classification CNN. At last, the sequence could also be combined into MP4 sequences.

4.2.2 Multi-view Filtering

In the 2D Image Sequences, there are sequences with more than one view. These multi-view images are not related to viewing classification; thus, should be removed. Figure 4.3 shows an example of a multi-view image.

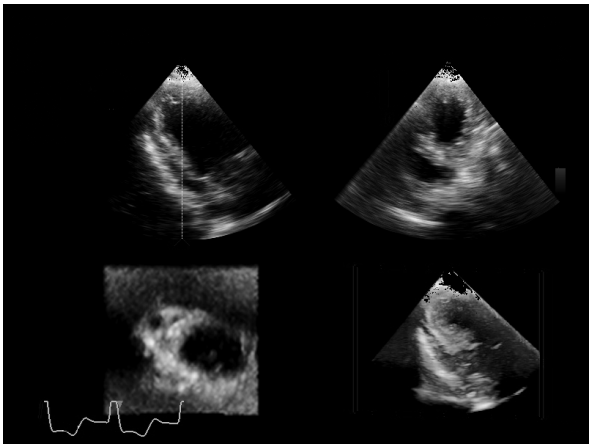

Figure 4.3: A Multi-view Image

To address the problem, a simple classification CNN is implemented to filter those multi-view images out. The multi-view classification CNN determines if an image consists of multiple views. Labelling of the data is quick and simple so supervised learning is adopted for this multi-view classification CNN. The model has performed well in filtering the images, the overall accuracy is 98.28%.

Dataset

Table 4.1: Number of Images in Multi-view Dataset

Datasets	**Multi**	**Non-multi**
Training Dataset	4,800	17,000
Testing Dataset	7,500	85,500

Table 4.1 shows the number of images used in the training and testing dataset. In the training dataset, the number of images in two classes is imbalanced. Therefore, a training weight Multi : Non-multi = 4 : 1 is introduced to the training process.

Data Augmentation

Simple data augmentation is also applied for training the network. The following list shows the data augmentation operations used:

- Rotation: Random rotation of the image up to 15 degrees;
- Flipping: Random horizontal flip.

Model Architecture

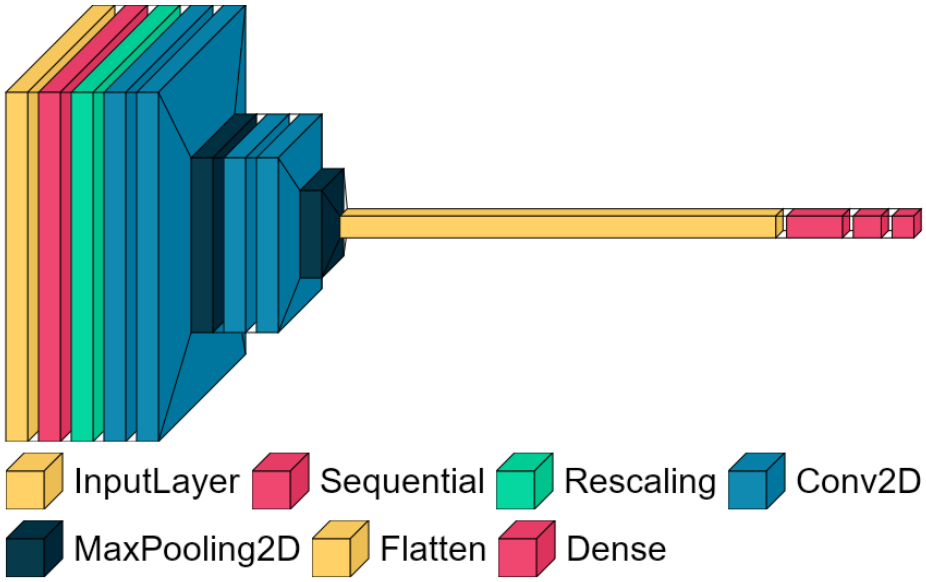

Figure 4.4: Model Architecture of Multi-view Classification CNN

Results

It performs well to filter the multi-view images. The network has achieved 98.28% accuracy in classifying the test dataset. This proposed solution is adopted.

4.2.3 Colour Measurement Filtering

The 2D data collected also contains coloured measurement images. These images are captured when the doctors were doing measurements. In these images, colour measurement overlay covers the heart data region so these images cannot be used and should be removed. Figure 4.5 shows an example of a coloured measurement image.

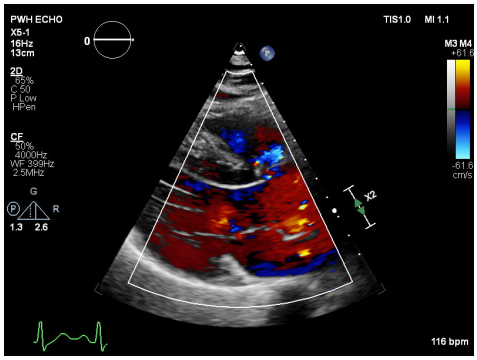

Figure 4.5: A Coloured Measurement Image

To remove these coloured images, a regional colour search is proposed as a solution. Some colours like orange can only be found in these images. This method tries to search for these exclusive colours that can only be found in these images. Every colour image consists of a colour legend on the right-hand side. A pixel-by-pixel scan throughout the image is conducted. If the exclusive colour is found, the whole sequence will be discarded.

To improve the performance of the scan, cropping is also applied so that the size of the scanning area is reduced. Figure 4.6 shows the cropped image for scanning.

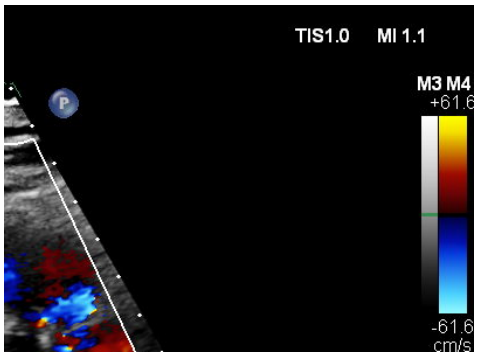

Figure 4.6: A Cropped Coloured Measurement Image

4.2.4 Overlay Removal

The 2D data collected contains overlays, which are needed to be removed. Figure 4.7 shows an example of the data image collected.

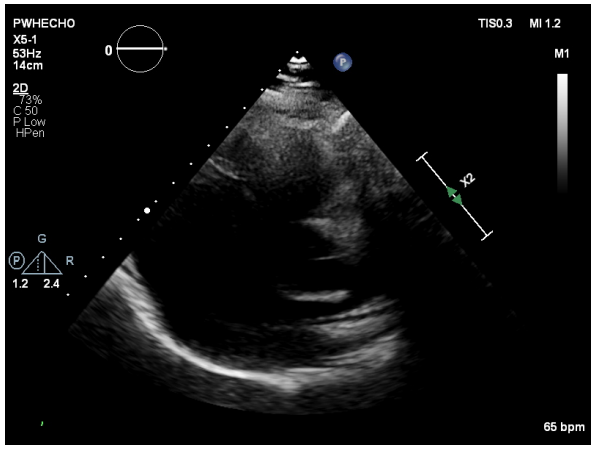

Figure 4.7: Data Image with Overlay

The image in Figure 4.7 contains a cross-section of the heart and the overlays with scales, text and other information. These overlays will affect the learning process of the network. The network may classify a view by identifying the overlays instead of data features. Therefore, the overlays should be removed. However, with the tremendous amount of data, it is impossible to remove them one by one manually. This creates a challenge in processing the data. To remove the overlays, two methods, mask removal method and line detection method, have been proposed as solutions.

Proposed Solution 1: Mask Removal

The data collected are five- to ten-second sequences of different view images. The cross-section part moves as the heartbeats. Most of the overlays, on the other hand, do not change in the sequence. Therefore, the dispersion of the brightness at the overlay is low. The standard deviation (SD) of the brightness should be close to 0. The formula of SD is given by:

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N_s}}$$

where x_i is the value of the point at time instance i
μ is the mean value at the point
N_s is the number of time frames in the sequence

Calculating SD of every point in the image and thresholding, we obtain the following mask in Figure 4.8.

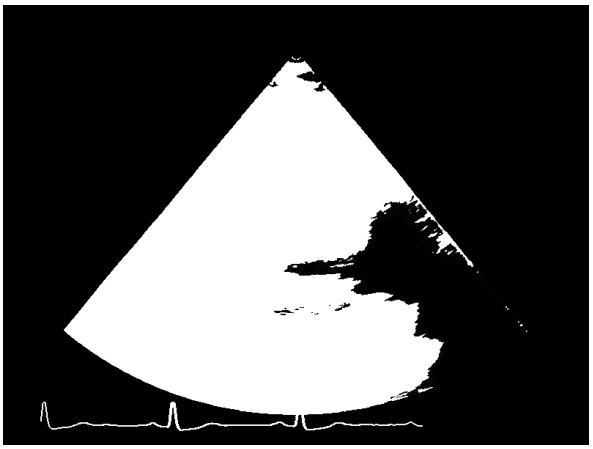

Figure 4.8: Standard Deviation Mask

In Figure 4.8, the white area represents points with high SD and is retained; the black area represents points with low SD and is abolished. One could discover that the heartbeat overlay in the bottom has high SD and is included in the mask. Meanwhile, some parts of the heart data have low SD and are not included in the mask. To recover the missing part of the heart data. A thresholding mask is also introduced. As the brightness at the heart is usually dimmer than the overlays, thresholding could cover most of the heart data. Then, we obtain the threshold mask in Figure 4.9

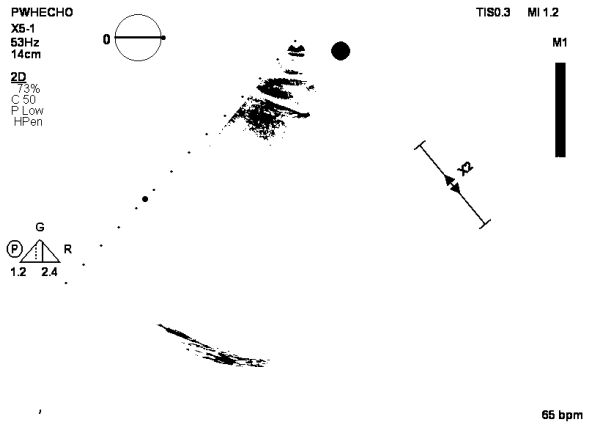

Figure 4.9: Threshold Mask

Figure 4.9 shows a threshold mask of the data in 4.7. The white area represents points lower than the threshold and is retained; the black area represents points higher than the threshold and is abolished. Combining two masks, we obtain the combined mask in Figure 4.10.

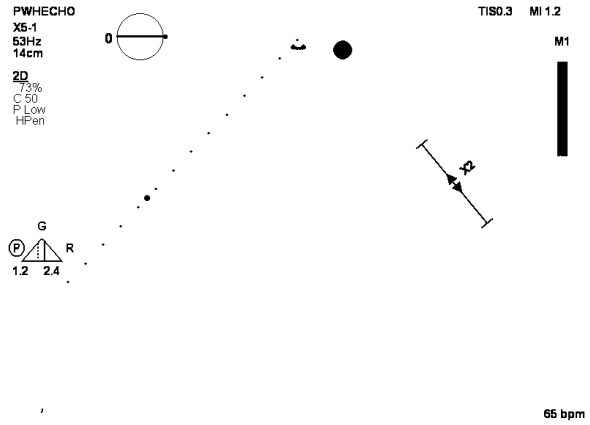

Figure 4.10: Combined Mask

In Figure 4.10, most of the heart is retained and most of the layout is abolished. Applying the mask to the image, we have the result in Figure 4.11.

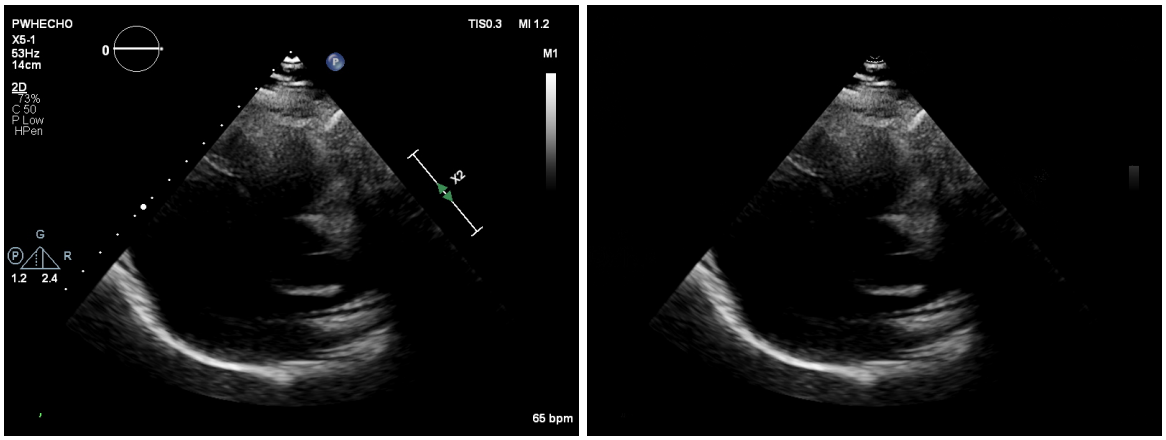

Figure 4.11: Overlay Removal Result

Figure 4.11 shows the image processing result. This proposed method can remove most of the overlay in the image. However, it may also remove heart data where the pixel has high value and does not change in the sequence, like the tip of the heart data in Figure 4.11.

The proposed method has demonstrated well in removing the overlay. This solution is adopted in the project.

Proposed Solution 2: Line Detection

The region of interest (ROI) of the image is a sector of a circle. This line detection method tries to extract the ROI instead of removing the overlay. This method first processes the image, including converting it to binary and blurring. Then, edge detection is applied to the image to extract the shape of the objects on the image. After that, a line detection is applied to the image to detect the two lines of the heart region. The intersection point of the two lines will serve as the centre of the circle. By calculating the radius, a circle could be drawn. Finally, by computing the intersections of the circle and the two lines, it gives the sector of the heart region, and ROI is extracted.

The implementation of this method uses the OpenCV library, which provides great and easy support in processing the image. Currently, the implementation of this line detection method is not complete. Here shows the temporary result of this method.

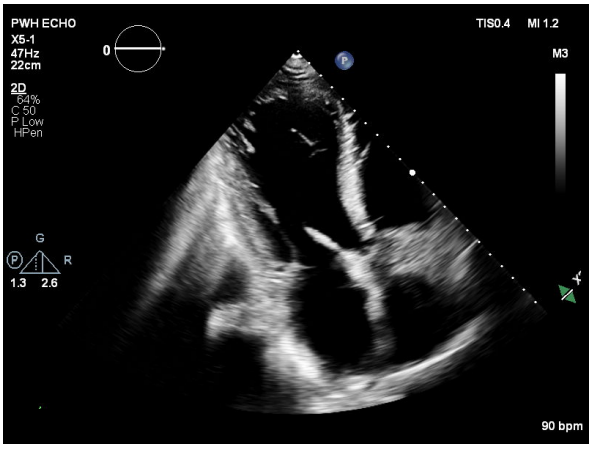

Figure 4.12: Original Data Image with Overlay

Figure 4.12 shows the original image with overlay. A convertion to binary image is done so that the shape of the heart data region could be better captured. Figure 4.13 shows the binary image created.

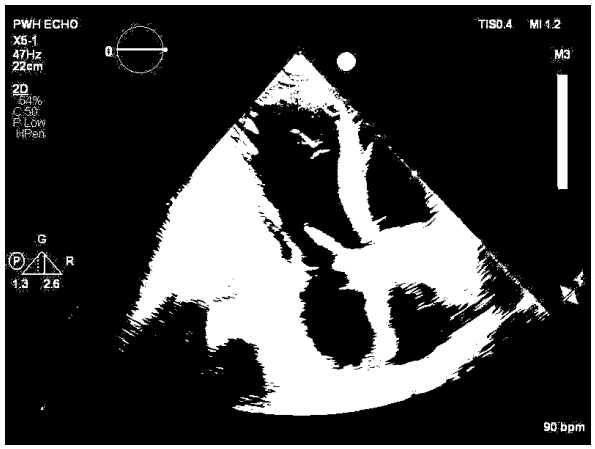

Figure 4.13: Binary Data Image

Then, Gaussian blur is applied to the binary image to reduce the bumps and noises created during the convertion to binary image. Figure 4.14 presents the blurred image after appling Gaussian blur.

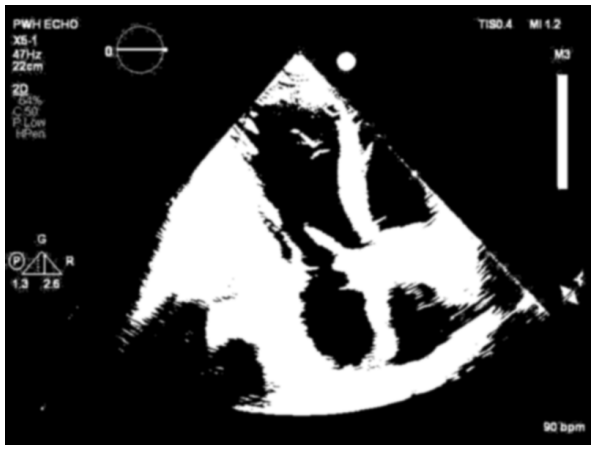

Figure 4.14: Blurred Binary Data Image

For the edge detection part, Canny Edge Detection is applied. Figure 4.15 shows the edge detected by Canny Edge Detection.

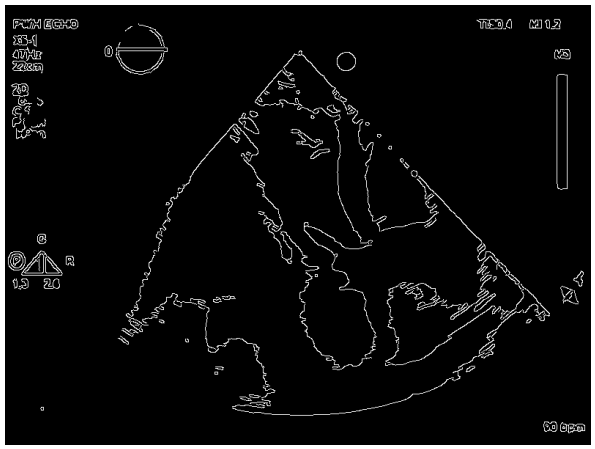

Figure 4.15: Edge Detected from Data Image

Afterwards, Hough Line Transform is applied to the edge detected. Figure 4.16 shows the line detection result.

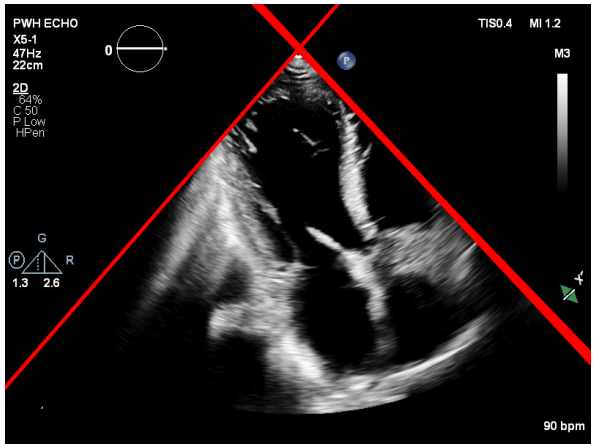

Figure 4.16: Line Detected from Data Image

In Figure 4.16, the two-line edges of the heart data were successfully detected. However, there are serval lines detection for the right edge because of the bumps caused by the overlapping overlay on that edge. Currently, the proposed solution to this problem is to average the duplicated lines detected.

The focus of building the dataset for training the view classification CNN has shifted to the labelled 3D. Thus, the implementation of the remaining steps is not yet completed and not adopted in the project.

4.2.5 Summary

In this project, the 2D data is processed to build the dataset for view classification CNN. This processing workflow transforms the raw DICOM files into PNG files. It also removes unwanted data and overlays in the image sequences, including both multi-view and colour measurement image sequences. This has provided a complete processing data flow for the raw 2D DICOM files.

4.3 Proposed Solution for View Classification CNN

The main challenge for the development of the view classification CNN is the lack of labelled data. Although there are 2D image sequences for training, the data is not labelled. The following highlighted component in the overview shows the corresponding related part of the project.

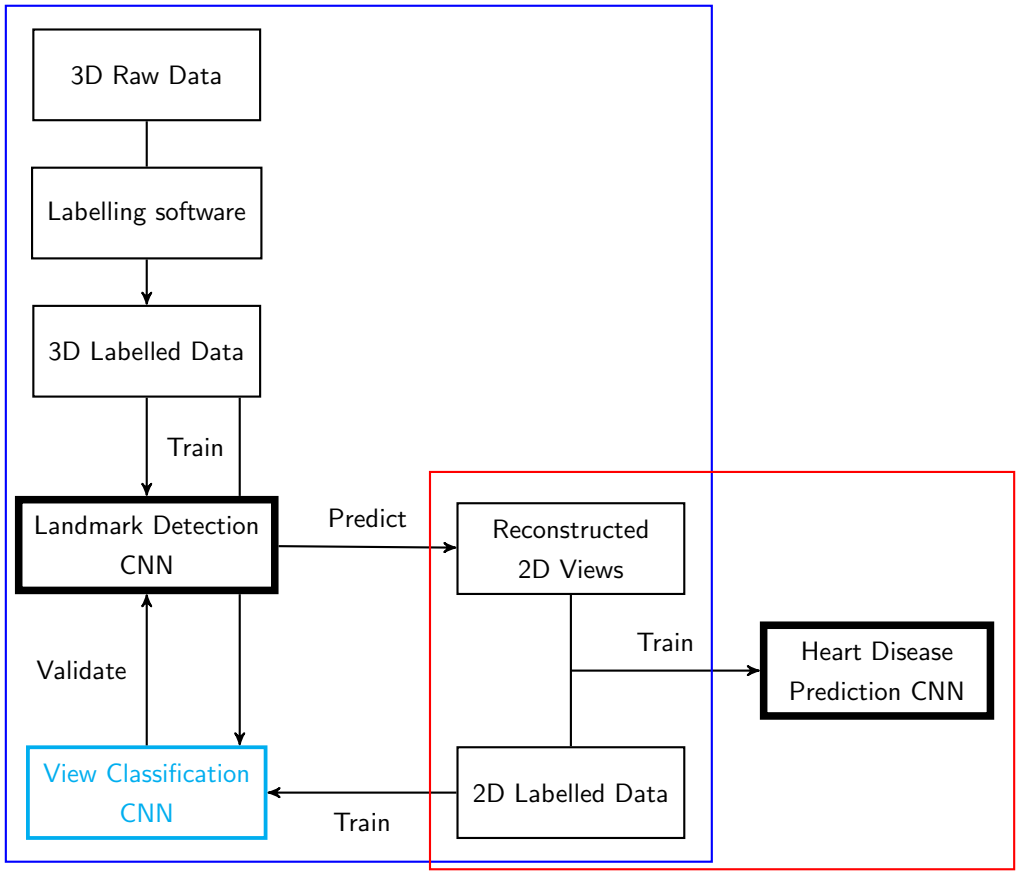

Figure 4.17: View Classification CNN in Project Overview

The initial approach to developing the view classification CNN is by semi-supervised or unsupervised learning methods. In this FYP, surrogate classification CNN is proposed as a solution.

4.3.1 Surrogate Classification CNN

This solution is a modification to the method proposed by Ghahramani et al., 2014 [9]. They have proposed an unsupervised learning method to address the problem of expensive labelling. In this proposed solution, a semi-supervised learning approach is adopted instead. This modified method requires little labelled data as 'seeds'. Each 'seed' is considered as a separate surrogate class in the classification model. Data augmentation is applied to each seed to create extra data for each surrogate class. Finally, the model is trained with the surrogate class using supervised learning method.

Data Augmentation

Each 2D data is a sequence of images; thus, all images in the sequence have the same label. The proposed method uses a whole sequence as a 'seed'. Figure 4.18 shows an example of the 'seed' of a surrogate class.

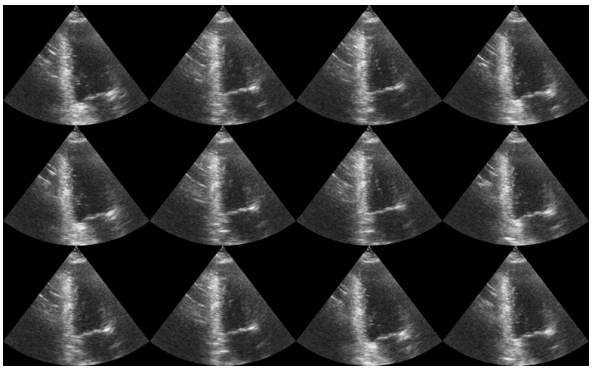

Figure 4.18: Orginal Images or 'Seed' of a Surrogate Class

New images are created through the data augmentation on the 'seed'. The data augmentation includes the following list:

- Rotation: Random rotation of the image up to 15 degrees;
- Translation: Random translation of the image in vertical and horizontal direction up to 10% of the image size;
- Scaling: Random scaling of the image up or down up to 10%;
- Contrast: Random adjustment to the contast of the image by a factor up to 10%.

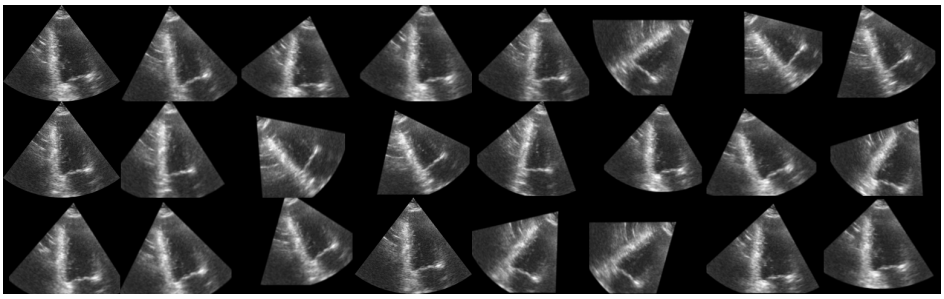

Figure 4.19: Augmented Images of a Surrogate Class

Figure 4.19 shows the results of random augmentation from the 'seed'.

Learning Algorithm

The learning algorithm of the surrogate classification CNN is similar to a typical classification CNN. It treats the surrogate classes as separate classes. Formally, the algorithm minimises the cross-entropy

loss function, which is given by:

$$H = -\sum_{c=1}^{MS}(y_c \log \hat{y}_c)$$

where M is the number of true classes
S is the number of surrogate classes per true class
y_c is the boolean value of y being surrogate class c
$\hat{y}_c$ is the predicted value of y in being surrogate class c

Validation Methods

The model introduces multiple surrogate classes for the same true class. Thus, the accuracy of the model correctly predicting the surrogate classes do not reflect the performance the best. Instead, many validation methods could be introduced - the method to interpret predicted data and classify the input image. The following list shows some of the many possible validation methods:

1. Predicted Surrogate Class:
   True class of the predicted surrogate class wins.

2. Top-k Majority:
   Consider the top k highest prediected surrogate classes, the image is classified to majority of the true class. If there is a tie, true class with highest sum wins.

3. Highest Average:
   Averaging all predicted surrogate classes in the same true class. True class with highest average wins.

4. Top-k Highest Average:
   Averaging top k predicted surrogate classes in the same true class. True class with highest average wins.

5. Thresholding Majority:
   Count number of surrogate classes in the same true class that surpass the thresholding value, the image is classified to majority of the true class. If there is a tie, true class with highest sum wins.

Results

The implementation of the surrogate classification CNN is complete, including the data augmentation and training steps. Validation methods, predicted surrogate class and top-k majority, are implemented. However, this is not adopted in the project due to the use of labelled 3D data for the view classification CNN.

4.3.2 Summary

Although the 2D data processing workflow provides datasets for training the view classification CNN, the labelling problem remains a huge problem. The poor performances in accuracy using semi-supervised and unsupervised learning methods are also a problem. Therefore, another approach of building 2D image datasets from labelled 3D data via reconstruction means is proposed. The dataset for the view classification CNN contains both 2D image sequences and reconstructed 2D image sequences from 3D data.

4.4 View Reconstruction for View Classification CNN

With the annotated 3D echo data, we have enough information (i.e. 3 points) to define the annotated views. Hence, the approach of building the view classification CNN datasets has been changed to use and reconstruct views from labelled 3D data. View reconstruction can also be used to reconstruct the predicted views from the landmark detection CNN. The following highlighted component in the overview shows the corresponding related parts of the project.

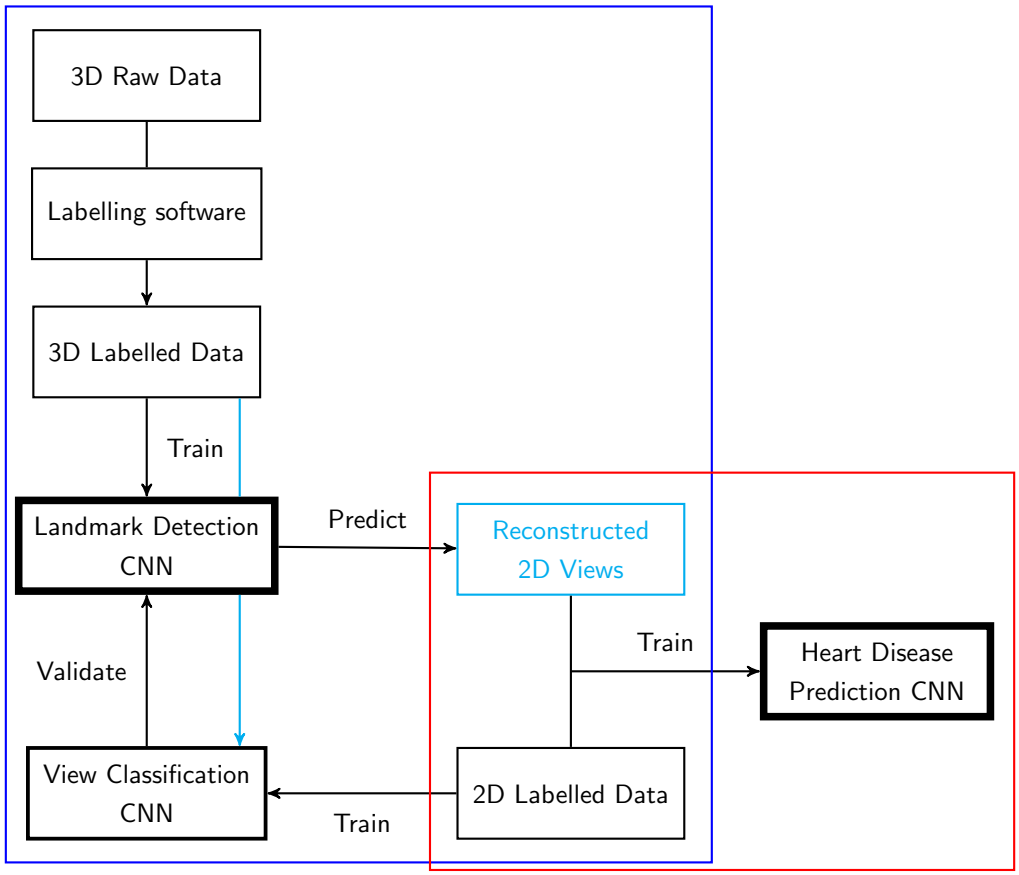

Figure 4.20: View Reconstruction in Project Overview

4.4.1 View Reconstruction

In the view reconstruction process, transformations between four different vector spaces are involved.

1. XY Space: The vector space representing the image
2. Plane Space: The vector space representing the 2D plane
3. RAS Space: The vector space representing the real world system
4. IJK Space: The vector space representing the 3D data array

The view reconstruction could be simplified into a multi-system transformation. For each point in the image (XY Space), get the value of the data at the point in IJK Space, namely

$$\text{Image}[\mathbf{v}] = \text{Data}[M_{\text{RAS}\to\text{IJK}} M_{\text{Plane}\to\text{RAS}} M_{\text{XY}\to\text{Plane}} \mathbf{v}]$$

where	Image	is the output image
	Data	is the 3D data array
	$\mathbf{v}$	is a vector representing a point in image (XY Space)
	$M_{A\to B}$	is the transformation matrix transforming from Space A to B

Each transformation has a certain meaning to the image or the data:

1. $M_{\text{XY}\to\text{Plane}}$: It specifies the image dimension, field of view and origin (location)
2. $M_{\text{Plane}\to\text{RAS}}$: It specifies the plane orientation and defines the plane
3. $M_{\text{RAS}\to\text{IJK}}$: It specifies the data space orientation

Therefore, there are two approaches in reconstructing the view.

1. Save $M_{\text{Plane}\to\text{RAS}}$ when annotating
2. Build $M_{\text{Plane}\to\text{RAS}}$ from the three points

Both methods could be used to reconstruct the views using labelled data. However, as the Landmark Detection CNN outputs only the positions of points, only the second method could be used to do reconstruction for visualising the output of Landmark Detection CNN.

Saving Transformation Matrix

Saving the transformation matrix is a straightforward approach. The transformation matrix $M_{\text{Plane}\to\text{RAS}}$ is saved when annotating. When we reconstruct the view, we could simply apply the same matrix to $M_{\text{Plane}\to\text{RAS}}$, setting other matrices manually. The orientation of the slice would be the same as the slice during labelling.

Build Matrix from Three Points

Using three points, a plane can be defined in a 3D space. The points could also be used to define the transformation matrix $M_{\text{Plane}\rightarrow\text{RAS}}$. However, the defined plane orientation would not be the same as the slice during labelling. The plane rotation and plane normal may be incorrect.

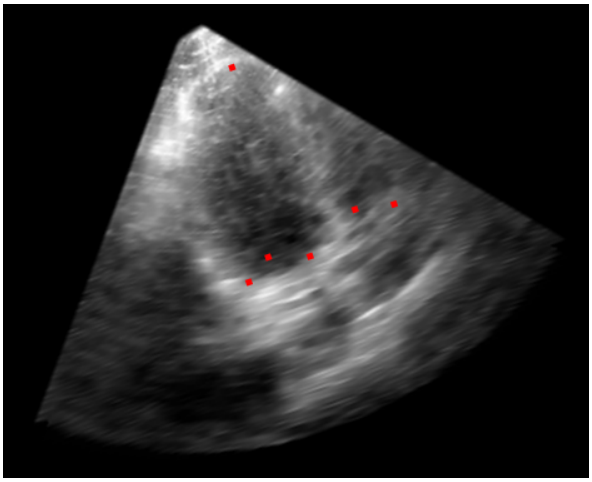

Figure 4.21: Reconstructed View with Incorrect Orientation

Figure 4.21 shows a reconstructed 4 chamber view with incorrect orientation. The reconstructed view has both incorrect rotation and normal. To solve this problem, similarity transform is proposed as a solution.

Similarity Transform

Similarity Transform is a transform operation $f : V^n \rightarrow V^n$ such that the reference and target points are matched or partially matched. To maintain the shape of the view, the similarity transform can only be a combined transformation of translation, rotation and flipping. To apply the similarity transformation, a set of reference points is stored.

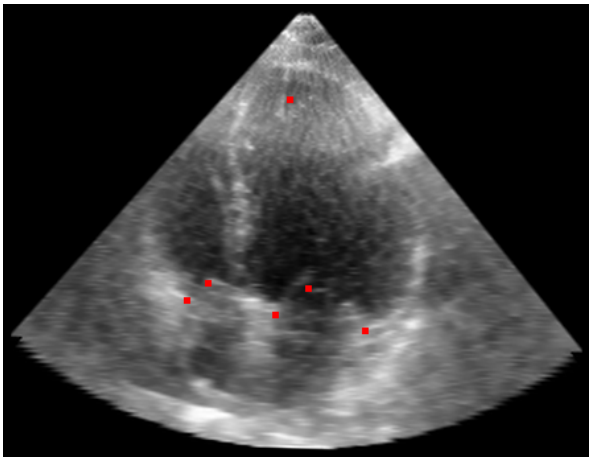

Figure 4.22: Reference Point for Reconstructed View

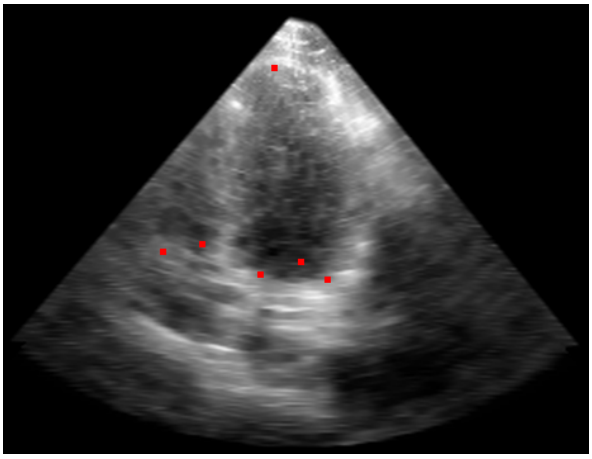

Figure 4.23: Reconstructed View After Applying Similarity Transform

Figure 4.22 shows the set of reference points of 4 chamber view. Figure 4.23 shows the results after applying similarity transform on Figure 4.21 using reference from Figure 4.22.

4.4.2 Summary

View reconstruction has provided a solution to the data labelling problem in developing view classification CNN. It can also be used as a visualisation tool for the output of landmark detection CNN.

4.5 3D Data Preprocessing for Landmark Detection CNN

To train the landmark detection CNN, labelled 3D data is needed. In this project, the 3D data is sent to the doctors for labelling using the software previously developed. Before the data is sent to be labelled and used to train in the landmark detection CNN, certain adjustments to the data would be made, including centring, flipping and spacing normalisation. The below-highlighted components in the project overview show the corresponding related parts in the project.

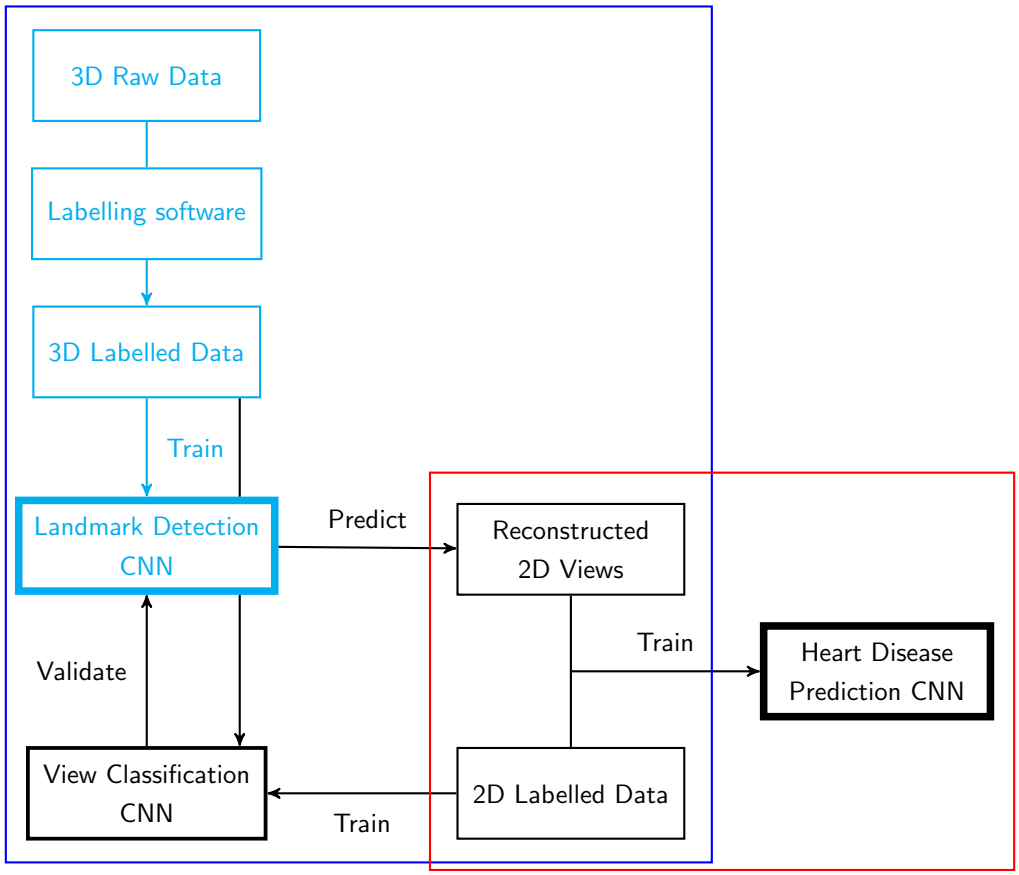

Figure 4.24: 3D Data Processing in Project Overview

4.5.1 Overview

The preprocessing of the 3D data involves two parts - before labelling and before training. Before labelling, the data is rotated and centred to have a better orientation for labelling. Before training, the data is normalised for consistent spacing for training the network.

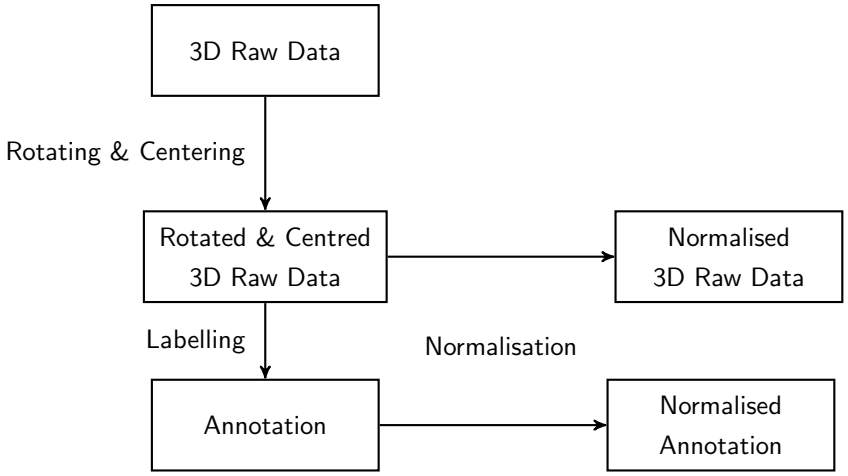

Figure 4.25: 3D Data Processing Overview

Figure 4.25 shows the overview of 3D Data Processing in this project. The 3D raw data is first rotated and centred for labelling. After labelling, the data and annotation files are normalised for network training.

4.5.2 Label Preprocessing

Before labelling, the data will be flipped and centred for labelling so that the data is displayed in conventional orientation and at centre. For a 3D data with dimensions d_x, d_y, d_z. Then, the following transformation is applied.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{d_x}{2} \\ 0 & 1 & 0 & -\frac{d_y}{2} \\ 0 & 0 & 1 & -\frac{d_z}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix on the right first translates the data's centre to the space origin. The data is then rotated around the x-axis by the transformation matrix on the left.

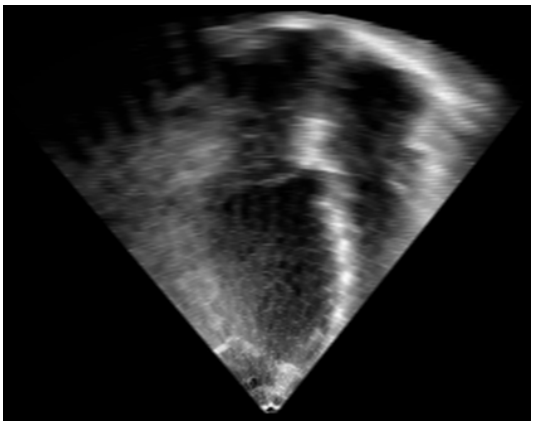

Figure 4.26: Incorrect Data Orientation

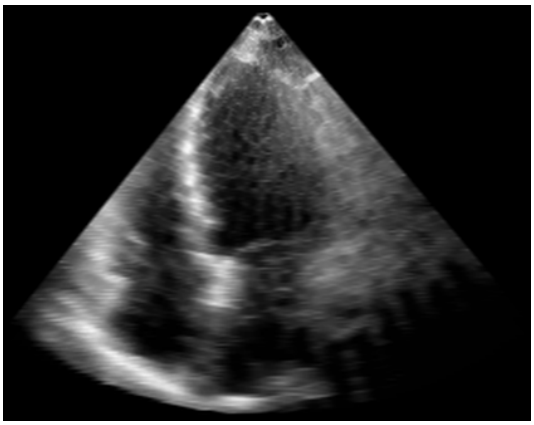

Figure 4.27: Correct Data Orientation

Figure 4.26 shows the original incorrect data orientation. Figure 4.27 shows the correct data orientation after the transformation.

4.5.3 Network Training Preprocessing

Each 3D data has a different spatial orientation, namely the spacing between two data points in the three axes in the 3D array. It means that the IJK-spaces between two different 3D data are different. This spacing difference is eliminated when it is transformed from IJK-space to RAS-space during reconstruction. The spatial difference between the 3D data has to be eliminated before using it for network training. The data is, therefore, transformed to RAS-space first, normalising the spacing between data points to 1. The following transformation is applied to the data.

$$\begin{bmatrix} \frac{1}{s_x} & 0 & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 & 0 \\ 0 & 0 & \frac{1}{s_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where s_k is the spatial orientation in k-axis

By applying the above transformation, the spatial orientation is set to 1 for the data, which is equivalent to transforming IJK-space to RAS-space. Then, the network can be trained with data inconsistent real-world space.

4.5.4 Summary

3D data preprocessing includes labelling preprocessing and network training preprocessing. The labelling preprocessing involves rotation and centring so that the data is shown in conventional orientation. The network training preprocessing involves data normalisation so that the spatial orientation is set to 1 and is consistent across all 3D data for network training.

5 Conclusion

This project proposes an expeditious way to do an examination using two CNN models with 3D echocardiography. The two models, Landmark Detection CNN and Heart Detection CNN, could use 3D raw echo data to reconstruct standard views and do predictions of heart disease. Different deep learning state-of-the-art architectures and methods will be implemented and tested to build the models. This final year project focuses on implementing the view classification CNN and the early development of the landmark detection CNN. It includes 2D data preprocessing for view classification CNN, proposed solution for view classification CNN, view reconstruction for view classification CNN and output visualisation of landmark detection CNN, and 3D data preprocessing for labelling and landmark detection CNN. Using the new method proposed, it is hoped that the examination time and waiting time could be reduced accordingly.

References

[1] W. H. Organization. "Cardiovasular diseases." (2021), [Online]. Available: https://www.who.int/health-topics/cardiovascular-diseases (visited on 09/26/2021).

[2] T. G. o. t. H. K. S. A. R. Centre for Health Protection Department of Health. "Heart diseases." (2019), [Online]. Available: https://www.chp.gov.hk/en/healthtopics/content/25/57.html (visited on 09/26/2021).

[3] C. Mitchell, P. S. Rahko, L. A. Blauwet, *et al.*, "Guidelines for performing a comprehensive transthoracic echocardiographic examination in adults: Recommendations from the american society of echocardiography," *Journal of the American Society of Echocardiography*, vol. 32, no. 1, 2019. DOI: 10.1016/j.echo.2018.06.004.

[4] E. S. of Cardiology. "Comparison between 2d and 3d." (2021), [Online]. Available: https://www.escardio.org/Education/Practice-Tools/EACVI-toolboxes/3D-Echo/comparison-between-2d-and-3d (visited on 09/27/2021).

[5] T. Ma, A. Gupta, and M. R. Sabuncu, "Volumetric landmark detection with a multi-scale shift equivariant neural network," *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 2020. DOI: 10.1109/isbi45749.2020.9098620.

[6] "3d slicer image computing platform." (), [Online]. Available: https://www.slicer.org/.

[7] "Opencv: Canny edge detection." (), [Online]. Available: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html/.

[8] S. Lee. "Lines detection with hough transform." (2021), [Online]. Available: https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549.

[9] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.