



**THE UNIVERSITY OF HONG KONG**

**BEng(CompSc) Capstone Project**

**Speeding Up Examination of Heart Diseases  
with 3D Echocardiography and Machine Learning**

April 2023

Lu Meng 3035639379

lumeng@hku.hk

## Abstract

Modern hospitals use 2D echocardiography to obtain cross-section views of the heart. However, this process is time-consuming regarding the large number of cross-section views needed to be obtained and the insufficient training of transducer operators. The lengthy acquisition time also increases the wait time before a patient can receive an echocardiography examination. Therefore, the project aims to develop a machine learning algorithm to reconstruct the cross-section views from 3D echocardiography automatically. It consists of mainly two tasks. The first landmark localization task utilizes 3D echocardiography to shorten the acquisition time of the heart's structural information and extends the fully convolutional SpatialConfiguration-Net (SCN) to detect local and global features of cardiac landmarks in the 3D echocardiography. SCN excels at this job due to its two task-oriented components that can be trained in an end-to-end manner. Besides, the Adaptive Wing loss is incorporated into SCN for better convergence to the heatmap-based supervision. The second cross-section recovery task adopts the least squared distance fit to recover the cross-section plane parameters based on the predicted landmark locations and reconstructs the 2D cross-section views. The project achieved desirable results in both the landmark localization and cross-section recovery tasks by demonstrating small errors on multiple metrics and outputting satisfactory visualizations on the cardiac dataset with limited training data. As a result, the acquisition and analyzing time will be significantly shortened. The project will be the world's first AI software for automatically reconstructing cross-section views from 3D echocardiography. Furthermore, it is the prerequisite of many downstream 2D-based heart disease classifiers that can assist doctors in heart disease diagnoses and make mass heart disease screening possible.

**Keywords:** 3D echocardiography, convolutional neural network, landmark localization

## **Acknowledgment**

I would like to give my warmest thanks to my supervisor, Professor Kenneth Wong, who made this project possible. His guidance led me through the challenges of the project. I would also like to thank his research assistant, Jerry Tam, who offered extensive technical instructions throughout the project.

## Table of Contents

<b>1. Introduction</b>	8
<b>1.1. Background</b>	8
<b>1.2. Objectives</b>	8
<b>1.3. Outline</b>	9
<b>2. Related Work</b>	10
<b>2.1. Convolutional Neural Networks</b>	10
<b>2.2. Patch-based Fully Convolutional Neural Network</b>	10
<b>2.3. Two-stage Task-oriented Deep Learning</b>	11
<b>2.4. SpatialConfiguration-Net</b>	12
<b>3. Data Preparation</b>	14
<b>3.1. Description</b>	14
<b>3.2. Preprocessing</b>	16
<b>3.2.1. Extraction</b>	17
<b>3.2.2. Normalization</b>	17
<b>3.2.3. Resizing</b>	17
<b>3.2.4. Splitting</b>	18
<b>4. Methodology</b>	19
<b>4.1. Network Structure</b>	19
<b>4.2. Ground Truth Generation</b>	20
<b>5. Experiments</b>	22
<b>5.1. Setup</b>	22
<b>5.2. Hardware Limitation</b>	22
<b>5.3. Hyperparameter Tuning</b>	23
<b>5.4. Improvement with Data Augmentation</b>	25
<b>5.5. Improvement with Adaptive Wing Loss</b>	26
<b>5.6. Improvement with Learning Rate Decay</b>	30
<b>5.7. Improvement Analysis</b>	31
<b>6. Cross-section Recovery</b>	35

<b>6.1. Method Choice</b>	35
<b>6.2. Handling SAXB</b>	38
<b>6.3. Handling SAXA, SAXM, and SAXMV</b>	39
<b>7. Evaluation</b>	42
<b>7.1. Landmark Localization</b>	42
<b>7.2. Cross-section Recovery</b>	43
<b>7.3. Baselines</b>	44
<b>8. Visualization</b>	47
<b>9. Discussion</b>	50
<b>9.1. Limitations</b>	50
<b>9.1.1. Direct Prediction of Cross-section Parameters</b>	50
<b>9.1.2. Adaptive Learnable Ground Truth Heatmap</b>	50
<b>9.1.3. Sub-voxel Accuracy</b>	51
<b>9.1.4. Training Process</b>	51
<b>9.1.5. Cross-section Recovery Metrics</b>	52
<b>9.2. Future Work</b>	52
<b>10. Conclusion</b>	53
References	54

## List of Figures

<b>Figure 1.</b> Logical Structure of SpatialConfiguration-Net	12
<b>Figure 2.</b> 3D Echo Data Example	16
<b>Figure 3.</b> Data Preprocessing Workflow	16
<b>Figure 4.</b> SpatialConfiguration-Net Architecture	19
<b>Figure 5.</b> Ground Truth Heatmap Examples	21
<b>Figure 6.</b> Loss Plot after Hyperparameter Tuning	24
<b>Figure 7.</b> Data Augmentation Example	25
<b>Figure 8.</b> Loss Plot with Data Augmentation	26
<b>Figure 9.</b> Loss and Gradient of Loss Functions	27
<b>Figure 10.</b> Loss Plot with Adaptive Wing Loss	28
<b>Figure 11.</b> Loss Plot with Adaptive Wing Loss and Weighted Loss Map Mask	29
<b>Figure 12.</b> Loss Plot with Learning Rate Decay	30
<b>Figure 13.</b> Heatmap Visualization Example	33

## List of Tables

<b>Table 1.</b> Affiliation of Landmarks and Cross-section Views	14
<b>Table 2.</b> Number of Training, Validation, and Testing Data	18
<b>Table 3.</b> Runnable Configurations	23
<b>Table 4.</b> Point-to-point Error for Improvement Analysis	32
<b>Table 5.</b> Number of Parameters and MACCs	34
<b>Table 6.</b> Recovery Methods Comparison	37
<b>Table 7.</b> Improvement on SAXB Recovery	38
<b>Table 8.</b> Improvement on SAXA Recovery	40
<b>Table 9.</b> Improvement on SAXM and SAXMV Recovery	41
<b>Table 10.</b> Landmark Localization Evaluation	42
<b>Table 11.</b> Cross-section Recovery Evaluation	44
<b>Table 12.</b> Baseline Comparison of Planar Position	45
<b>Table 13.</b> Baseline Comparison of Planar Orientation	45
<b>Table 14.</b> Cross-section Visualization Examples	48

## Abbreviations

2D = Two-dimensional

3D = Three-dimensional

4D = Four-dimensional

A2C = 2 Chamber View

A4C = 4 Chamber View

ALAX = Long-axis View

CNN = Convolutional Neural Network

FCNN = Fully Convolutional Neural Network

JSON = JavaScript Object Notation

NRRD = Nearly Raw Raster Data

RANSAC = Random Sample Consensus

SAXA = Apical LV Short-axis View

SAXB = Basal Short-axis View

SAXM = Mid LV Short-axis View

SAXMV = MV Short-axis View

SCN = SpatialConfiguration-Net

SVD = Singular Value Decomposition

T<sup>2</sup>DL = Two-stage Task-oriented Deep Learning



# 1. Introduction

## 1.1. Background

According to the US National Center for Biotechnology Information, heart diseases are the leading cause of death globally, killing an estimated 13 million people annually, a quarter of the global total [1]. They are also the leading cause of hospitalization [2], which heavily burdens the global healthcare system.

Currently, hospitals mainly use 2D echocardiography to diagnose heart diseases [3]. This non-invasive technology enables medical practitioners to obtain a cross-section view of the heart structure with one scan. However, according to the American Society of Echocardiography, transducer operators need to seek 27 cross-section views to restore the complete structure of a heart [4]. Besides, the quality of the views is subject to the operators' individual experience. Therefore, the acquisition time for an echocardiography examination is as long as 30 minutes [5], causing patients to queue for 20 weeks to receive an examination [6].

## 1.2. Objectives

This project aims to speed up the examination of heart diseases by using 3D echocardiography and machine learning. 3D echocardiography can acquire a complete 3D representation of the heart with one scan in 5 seconds [7], significantly reducing the acquisition time. However, as 3D volumetric data is non-trivial for cardiologists to interpret, the project utilizes deep neural networks to reconstruct the cross-section views from 3D data automatically. This machine-learning approach can eliminate uncertainty from transducer operators' technical abilities.

The project is the world's first AI software for automatically reconstructing cross-section views from 3D echocardiography data. It is expected to reduce the echocardiography examination time to less than 3 minutes and shorten the wait time for an echocardiography examination to less than 10 weeks.

### **1.3. Outline**

This report will first review the literature and then discuss in detail the methods used in data preparation, landmark localization, and cross-section recovery. At last, it will provide evaluations and visualizations of the final results.

## 2. Related Work

As each cross-section view in the cardiac examination is defined by multiple cardiac landmarks, *i.e.*, multiple cardiac feature points lie on each cross-section plane, predicting the locations of the landmarks is essential to restoring the locations of the cross-section planes. In the literature, extensive machine-learning methods have been proposed for anatomical landmark detection. Most methods are based on convolutional neural networks (CNNs). Therefore, this section will first discuss the limitations of pure CNNs and review three advanced neural network frameworks.

### 2.1. Convolutional Neural Networks

CNNs are popular in landmark detection, as they outperform the state-of-the-art in many computer vision tasks, *e.g.*, ImageNet classification [8]. However, according to Payer *et al.* [9], pure CNNs do not perform satisfactorily on landmark prediction because, different from classification tasks, landmark prediction requires maintaining high-resolution feature maps to retain the accurate positions of the landmarks. As a result, CNNs perform poorly if they utilize pooling layers to reduce the resolution of feature maps [9].

This brings two major problems: larger feature maps mean a larger number of network parameters to be trained, which leads to greater memory consumption and lower efficiency; the lack of pooling layers restricts the receptive fields of convolutional kernels so that the network can only recognize patterns in small local areas but cannot draw connections between two far away areas. In landmark detection, not only the pattern of a landmark itself but also the relative positions between landmarks can help us locate the landmark. Therefore, many advanced methods aim to improve pure CNNs by incorporating local and global pattern recognition.

### 2.2. Patch-based Fully Convolutional Neural Network

Noothout *et al.* [10] proposed a patch-based fully convolutional neural network (FCNN) that combines regression and classification. FCNN first divides the input 3D

echo data into multiple isotropic voxels, *i.e.*, 3D patches of equal length, width, and height. Then, for each voxel, FCNN provides two outputs. The first output is a 3D displacement vector from the voxel's center to the landmark location. The second is a classification score indicating whether the voxel contains the landmark. To predict the actual location of a landmark, FCNN only selects voxels that contain or are near the landmark and averages the 3D vectors corresponding to these voxels. This way, it incorporates information from different voxels and avoids interference from far-away voxels.

However, FCNN suffers from both structural and computational shortcomings. Firstly, distance is not the only factor determining a voxel's usefulness in landmark prediction. Two parts of the heart may have structural relationships, even if not adjacent. Secondly, FCNN can only predict one landmark with one trained model. As our dataset contains 32 landmarks, it would be time-consuming to train 32 individual models.

### 2.3. Two-stage Task-oriented Deep Learning

Two-stage task-oriented deep learning (T<sup>2</sup>DL) framework proposed by Zhang *et al.* [11] is another novel framework that outperforms FCNN in that it can simultaneously predict multiple landmarks and automatically learn relationships among landmarks.

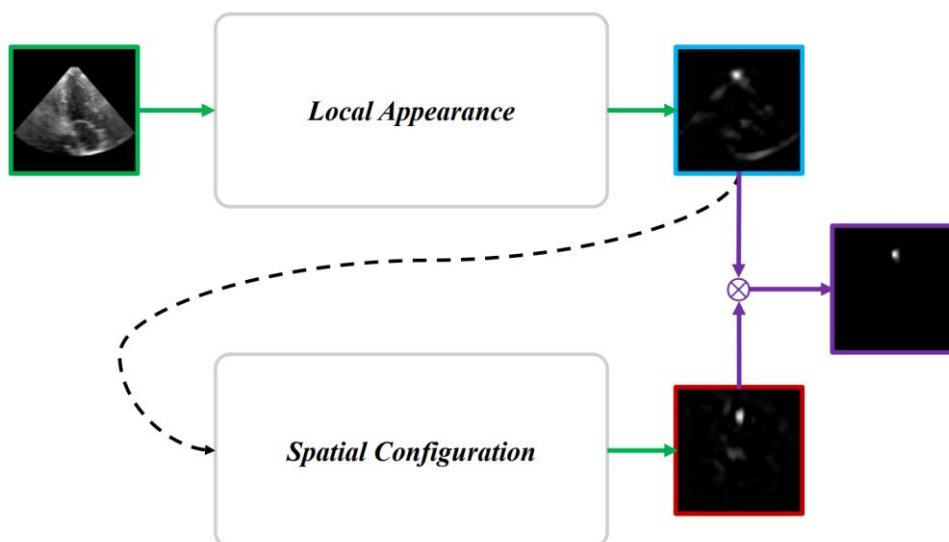
T<sup>2</sup>DL consists of two components, and they are trained separately in two stages. In the first stage, T<sup>2</sup>DL samples random voxels from 3D echo data and uses these sampled voxels to train the first component with voxel-based displacement vectors as supervision. In the second stage, the parameters in the first component are frozen, and the second component is concatenated to the end of the first component. To train the second component, T<sup>2</sup>DL feeds complete 3D echo data into the network and uses absolute landmark coordinates as supervision. T<sup>2</sup>DL is designed for large-scale landmarks [11], with the first component learning internal features within voxels and the second component considering correlation among voxels.

The method allows the network to automatically detect correlations among voxels instead of relying on distance, as in FCNN. However, the preprocessing of T<sup>2</sup>DL is

quite tedious. T<sup>2</sup>DL requires sampling millions of voxels as the training data for the first component, which significantly increases time complexity in both preprocessing and training.

## 2.4. SpatialConfiguration-Net

Unlike the voxel-based methods above, Payer *et al.* [12] proposed the SpatialConfiguration-Net (SCN) that predicts landmarks using heatmaps. Instead of outputting coordinates, SCN outputs heatmaps where brightness indicates the possible appearance of landmarks. SCN is also composed of two components, as shown in Figure 1. The first Local Appearance component outputs heatmaps by considering features in local areas. The second Spatial Configuration component interrelates previous heatmaps of different landmarks and outputs new heatmaps based on relationships among landmarks. At last, the heatmaps from the two components are element-wisely multiplied such that common hotspots are enhanced, and different hotspots are suppressed.



**Figure 1. Logical Structure of SpatialConfiguration-Net.** ■: input echo data, ■: local appearance heatmaps, ■: spatial configuration heatmaps, ■: output heatmaps, ⊗: element-wise multiplication.

This method not only predicts multiple landmarks at once but also requires less

complex preprocessing, which is why I chose SCN as the basic network framework and optimized its performance on our cardiac dataset.

### 3. Data Preparation

This section will describe the dataset and then discuss different data preparation methods before feeding the data into the neural network.

#### 3.1. Description

Our research team collaborates with Prince of Wales Hospital to collect annotated echocardiography data for training the network. After discussion with expert cardiologists, we confirmed that 32 landmarks and 7 cross-sections are enough for diagnosing heart diseases and are dealt with in this project. The affiliation of landmarks and cross-sections is shown in Table 1.

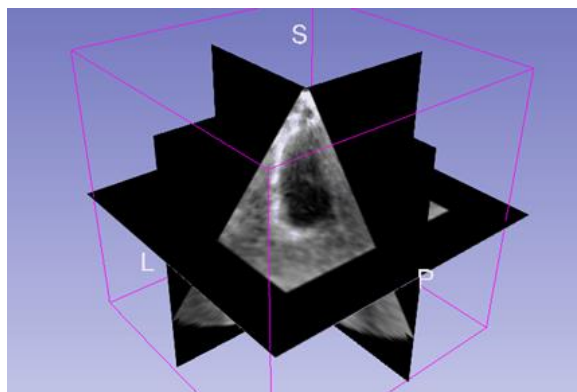
**Table 1. Affiliation of Landmarks and Cross-section Views**

View Name	Landmark Name
4 Chamber View (A4C)	MV tip ( $L_1^{(A4C)}$ )
	A4C-TV tip ( $L_2^{(A4C)}$ )
	A4C-LV apex ( $L_3^{(A4C)}$ )
	Medial mitral annulus ( $L_4^{(A4C)}$ )
	Lateral mitral annulus ( $L_5^{(A4C)}$ )
	Tricuspid annulus ( $L_6^{(A4C)}$ )
2 Chamber View (A2C)	A2C-LV apex ( $L_1^{(A2C)}$ )
	Anterolateral mitral annulus ( $L_2^{(A2C)}$ )
	Posteromedial mitral annulus ( $L_3^{(A2C)}$ )
Long-axis View (ALAX)	ALAX-LV apex ( $L_1^{(ALAX)}$ )
	Aortic annulus ( $L_2^{(ALAX)}$ )
	Anterior mitral annulus ( $L_3^{(ALAX)}$ )

	Posterior mitral annulus ( $L_4^{(ALAX)}$ )
Basal Short-axis View (SAXB)	Center of AV ( $L_1^{(SAXB)}$ )
	IAS ( $L_2^{(SAXB)}$ )
	SAXB-TV tip ( $L_3^{(SAXB)}$ )
	PV tip ( $L_4^{(SAXB)}$ )
MV Short-axis View (SAXMV)	MV anterior leaflet A1 ( $L_1^{(SAXMV)}$ )
	MV anterior leaflet A2 ( $L_2^{(SAXMV)}$ )
	MV anterior leaflet A3 ( $L_3^{(SAXMV)}$ )
	MV posterior leaflet P1 ( $L_4^{(SAXMV)}$ )
	MV posterior leaflet P2 ( $L_5^{(SAXMV)}$ )
	MV posterior leaflet P3 ( $L_6^{(SAXMV)}$ )
Mid LV Short-axis View (SAXM)	Anterolateral papillary muscle ( $L_1^{(SAXM)}$ )
	Posteromedial papillary muscle ( $L_2^{(SAXM)}$ )
	IVS ( $L_3^{(SAXM)}$ )
	IW ( $L_4^{(SAXM)}$ )
	LV ( $L_5^{(SAXM)}$ )
	RV ( $L_6^{(SAXM)}$ )
Apical LV Short-axis View (SAXA)	SAXA-LV apex ( $L_1^{(SAXA)}$ )
	Interventricular septum ( $L_2^{(SAXA)}$ )
	RV apex ( $L_3^{(SAXA)}$ )

$L_i^{(VIEW)}$  represents the  $i$ -th landmark on the cross-section  $VIEW$ , which will be referred to later.

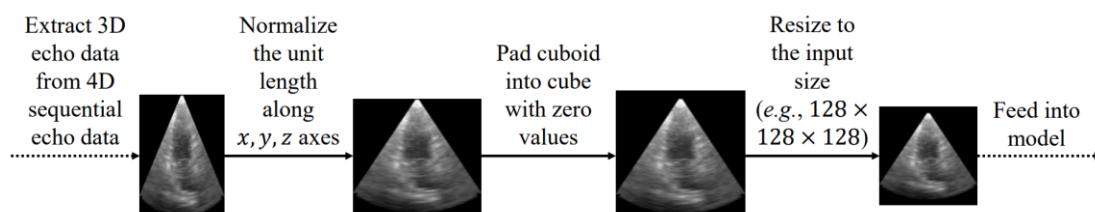




**Figure 2. 3D Echo Data Example.** This is a static frame of the 4D echo data rendered with the *3D Slicer* image computing platform.

The dataset consists of 371 data points collected from September 2021 to November 2021. Each data point comprises an NRRD file and its corresponding JSON file. The NRRD file is 4D, containing the complete 3D pyramid-shaped echo data of the heart over approximately 4 seconds, as rendered in Figure 2. Each intensity value in the 4D data ranges from 0 to 255, where 0 means no echo and 255 represents the strongest echo. The JSON file labels landmarks' information of the NRRD file. For each identifiable landmark, the JSON file describes its name, the cross-section view it belongs to, and its 4D coordinate, including time,  $x$ ,  $y$ , and  $z$ , relative to the NRRD data.

### 3.2. Preprocessing



**Figure 3. Data Preprocessing Workflow.** The extracted 3D echo data is first normalized, then padded into an isotropic cube, and finally resized to the desired input size.

This sub-section will discuss methods used in data preprocessing, including

extraction, normalization, and resizing, the workflow of which is shown in Figure 3. Besides it will also talk about splitting the training, validation, and testing dataset.

### 3.2.1. Extraction

My network uses 3D echo data with  $x$ ,  $y$ , and  $z$  axes as input. Therefore, I needed to extract 3D data from the 4D data at the time index when the landmarks are labeled. Notably, landmarks on different cross-sections were mostly labeled at different time indices within 4D data. Hence, in principle, I needed to extract seven 3D data from each 4D data and train separate models for each cross-section.

There are two points to note here. Firstly, some 4D data may only contain labels of some but not all seven cross-sections. As a result, such data was only used for training those labeled cross-sections, which caused the different number of training data for different cross-sections, as shown in Table 2. Secondly, landmarks on the SAXA, SAXM, and SAXMV views were labeled at the same time index because they are defined to be mutually parallel. Therefore, it is possible to use one model to predict all landmarks on the three cross-sections together. But I decided to train separate models for each cross-section for versatility. As a result, three copies of the same 3D data were generated for SAXA, SAXM, and SAXMV.

### 3.2.2. Normalization

After extracting the 3D echo data, I performed normalization to the data because the unit length along the  $x$ ,  $y$ , and  $z$  axes may be different due to the setting of transducers, as illustrated in the leftmost image of Figure 3. To correct this distortion, I used the inter-spacing information recorded in the NRRD header to restore the echo data's actual ratio using third-order spline interpolation. The unit length along all axes of the restored echo data is 1 millimeter.

### 3.2.3. Resizing

As the normalized data have different lengths, widths, and heights, I further resized

the data to the same size so that they can be fed into the neural network in batches. To achieve this purpose without distorting the echo data or introducing too much blank space, I first used zero values to pad each data into an isotropic cube according to its longest edge length and then zoomed the cube into a suitable size (e.g.,  $128 \times 128 \times 128$ ). Third-order spline interpolation was applied when zooming the data, and the scaling ratio was recorded for later evaluation. Additionally, a new set of annotations were generated to reflect the changes made to the data during normalization and resizing so that the locations of all landmarks were valid for the new echo data.

### 3.2.4. Splitting

For data splitting, I chose a fixed amount of 54 echo data as the testing set and adopted a training : validation = 8 : 2 ratio on the remaining data. However, although there are 371 labeled echo data in total, not all data contain annotations of all seven cross-sections, as discussed before. Therefore, the actual amount of data used for training and evaluating models of different cross-sections was slightly different, as shown in Table 2.

**Table 2. Number of Training, Validation, and Testing Data**

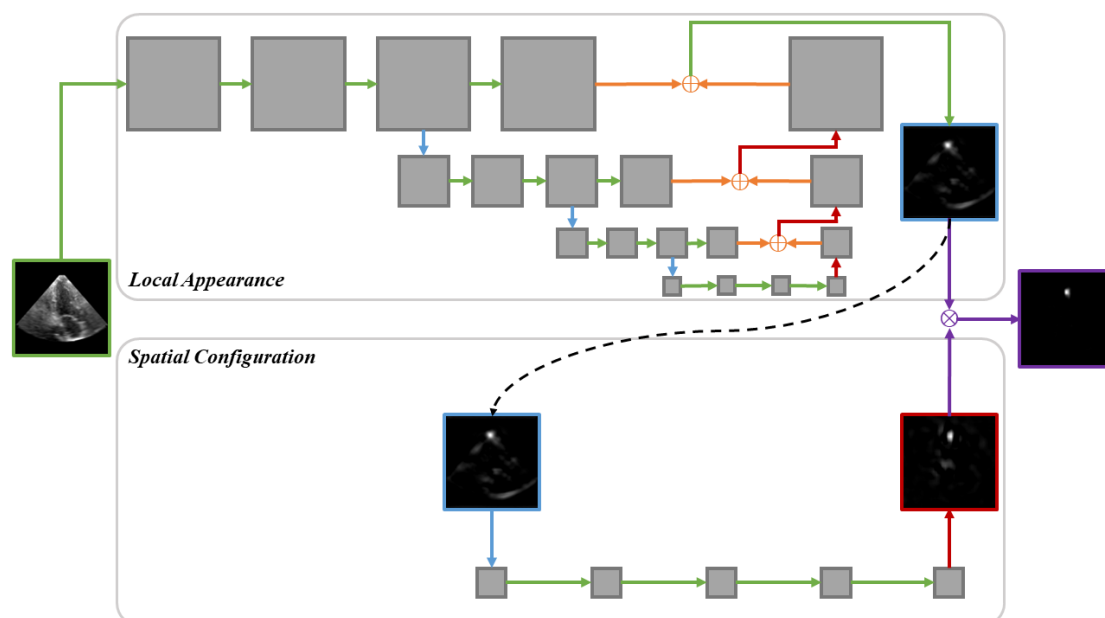
<b>Cross-section</b>	<b>Total</b>	<b>Training</b>	<b>Validation</b>	<b>Testing</b>
<b>A2C</b>	356	241	61	54
<b>A4C</b>	360	244	62	54
<b>ALAX</b>	353	240	60	53
<b>SAXA</b>	330	223	56	51
<b>SAXB</b>	305	207	52	46
<b>SAXM</b>	359	244	61	54
<b>SAXMV</b>	319	216	54	49

The number of testing data is at most 54, and the ratio of training and validation data is 8 : 2.

## 4. Methodology

### 4.1. Network Structure

My network reimplements the idea and the structure of the SpatialConfiguration-Net [12]. It is composed of two components, as shown in Figure 4. The first local appearance component outputs heatmaps by considering features in local areas. The second spatial configuration component interrelates previous heatmaps of different landmarks and outputs new heatmaps based on relationships among landmarks. At last, the heatmaps from the two components are element-wisely multiplied such that common hotspots are enhanced, and different hotspots are suppressed.



**Figure 4. SpatialConfiguration-Net Architecture.** The local appearance component generates heatmaps from the input echo data, and the spatial configuration component generates heatmaps from the local appearance heatmaps. The final heatmaps are the product of the local appearance and spatial configuration heatmaps. ■: input echo data, ■: local appearance heatmaps, ■: spatial configuration heatmaps, ■: output heatmaps, ■: intermediate feature maps;  $\rightarrow$ : convolution,  $\rightarrow$ : down-sampling,  $\rightarrow$ : up-sampling;  $\oplus$ : element-wise addition,  $\otimes$ : element-wise multiplication.

The local appearance component has four levels of different resolutions to capture different levels of detail. Each level consists of three  $3 \times 3 \times 3$  convolution layers with 128 intermediate channels where the small kernel size enables the network to

focus on local features. A dropout of 0.5 is included after the first convolution layer to increase generalizability. A  $2 \times 2 \times 2$  average pooling after the second convolution layer generates the level below. The local appearance heatmaps are generated from a  $3 \times 3 \times 3$  convolution layer with the number of output channels equal to the number of landmarks.

The spatial configuration component is calculated at  $\frac{1}{4}$  of the input resolution. It consists of three  $7 \times 7 \times 7$  convolution layers with 128 intermediate channels and an additional  $7 \times 7 \times 7$  convolution layer having the number of output channels equal to the number of landmarks. The low spatial resolution and the large kernel size enable the spatial configuration component to relate global features and output coarser heatmaps that can discriminate different local areas on the local appearance heatmaps. At last, the outputs are up-sampled back to the input resolution with tri-cubic interpolation.

Each intermediate convolution layer has a LeakyReLU activation with a negative slope of 0.1. The convolution layer generating local appearance heatmaps has no activation, while the convolution layer generating spatial configuration heatmaps has a TanH activation.

## 4.2. Ground Truth Generation

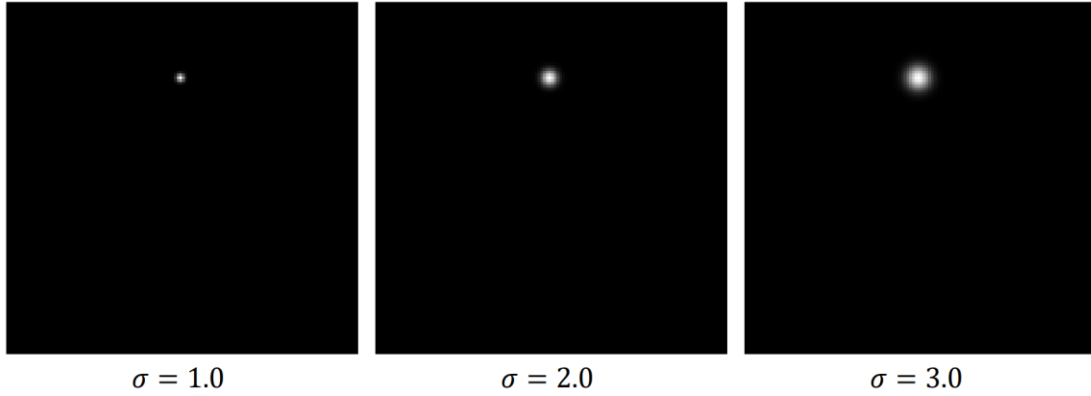
Since the output of SCN is a multi-channel heatmap, to supervise the network, I needed to generate a corresponding ground truth heatmap from the ground truth landmark locations for each echo data.

Let  $N$  be the number of landmarks on a cross-section view. A 3D channel  $\mathbb{H}_i^*(\mathbf{x}): \mathbb{R}^3 \rightarrow \mathbb{R}$  of a target landmark  $L_i, i \in \{1, \dots, N\}$  (see Table 1) is generated by placing a normalized 3D Gaussian function at the landmark location  $\mathbf{x}_i^* \in \mathbb{R}^3$ , as follows:

$$\mathbb{H}_i^*(\mathbf{x}; \sigma) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i^*\|^2}{2\sigma^2}\right)$$

where the standard deviation  $\sigma \in \mathbb{R}^+$  is a hyperparameter that controls the peak width

of the Gaussian function. Note that there is no coefficient preceding the exponent, as the peak value is always normalized to 1 to avoid small values of the Gaussian function.



**Figure 5. Ground Truth Heatmap Examples.** The images are 2D slices of 3D heatmap channels where Gaussian functions with different standard deviation  $\sigma$  are centered at the same location.

2D example heatmaps with different standard deviation  $\sigma$  are shown in Figure 5. Here  $\sigma$  represents the accuracy at which the network learns the landmark location. A large  $\sigma$  makes the network easier to fit to the heatmap, but the extracted maximum point would be less accurate. Meanwhile, a small  $\sigma$  makes it very hard to learn the landmark location as the foreground is too small compared to the background. To choose a reasonable  $\sigma$ , I combined my visualizations of heatmaps with different  $\sigma$  levels with experiments done by Payer *et al.*, who used  $\sigma \in [1.5, 3.0]$  for an average volume size of  $294 \times 512 \times 72$  [12]. I decided that  $\sigma = 2.0$  would be a suitable value if my volume size is  $128 \times 128 \times 128$ .

It is noteworthy that if there are more than three landmarks on a cross-section, some landmarks may not be labeled with the ground truth  $\mathbf{x}_i^*$ , possibly due to their vagueness. In such cases, the corresponding  $\mathbb{H}_i^*$  was just left blank with all zero values, and special handling during training will be introduced in Section 5.1.

Finally,  $\mathbb{H}_1^*, \dots, \mathbb{H}_N^*$  are stacked together as different channels of the heatmap  $\mathbb{H}^*$  for one cross-section.

## 5. Experiments

### 5.1. Setup

During experiments, I tested different hyperparameters, including spatial resolution, the number of intermediate channels, batch size, loss function, optimizer, learning rate, and L2 regularization.

In each experiment, the model was trained for at most 100 epochs, corresponding to approximately 23000 iterations. When loading the training data  $\mathbb{X}^{IN}$ , the intensity values of the original echo data  $\mathbb{X}$  were scaled and shifted from  $[0, 255]$  to  $[-1, 1]$  for faster convergence, as follows:

$$\mathbb{X}^{IN} = \frac{2\mathbb{X} - (\max(\mathbb{X}) + \min(\mathbb{X}))}{\max(\mathbb{X}) - \min(\mathbb{X})}$$

The loss is summed over voxels in each channel of the predicted heatmap and then averaged over all channels. Channels without ground truth values are left unsupervised by leaving them out of the averaged loss:

$$F_{loss}(\mathbb{H}, \mathbb{H}^*) = \frac{1}{|C^l|} \sum_{i \in C^l} \sum_{x,y,z} f_{loss}(\mathbb{H}_{i,(x,y,z)}, \mathbb{H}^*_{i,(x,y,z)})$$

where  $f_{loss} : \mathbb{R}^2 \rightarrow \mathbb{R}$  is a scalar loss function,  $\mathbb{H}$  is the predicted heatmap,  $\mathbb{H}^*$  is the ground truth heatmap, and  $C^l$  is the set of channels whose corresponding landmark is labeled.

### 5.2. Hardware Limitation

All models were trained on a NIVIDA GeForce RTX 2080Ti graphics card with 10GB of VRAM. The training time of 100 epochs was approximately 30 hours. Due to the limited VRAM, I first needed to determine the spatial resolution: the size of input echo data and output heatmaps, the number of intermediate channels during convolution, and the batch size. To test whether a configuration was runnable under the limited memory, I fed randomly initiated dummy data of different sizes into the SCN network of different structures. All runnable configurations are shown in Table 3.

The top priority is spatial resolution, as it directly determines the fineness of the

predicted landmark location. As shown in Table 3, the maximum attainable resolution is  $128 \times 128 \times 128$ . Under this resolution, the maximum number of intermediate channels is 128, the same as the recommended number by Payer *et al.* [12]. Although the batch size is only 1 in this configuration, larger batch sizes may not improve the result, according to experiments done by Payer *et al.* [12]. Therefore, I determined that the  $128 \times 128 \times 128$  resolution, 128 channels, and batch size of 1 were the optimal choices.

**Table 3. Runnable Configurations**

	B = 1			B = 2		
	C = 64	C = 128	C = 256	CH = 64	C = 128	C = 256
R = $64^3$	✓	✓	✓	✓	✓	✓
R = $128^3$	✓	✓	✗	✓	✗	✗
R = $256^3$	✗	✗	✗	✗	✗	✗

B: batch size, C: number of channels in intermediate feature maps, R: spatial resolution; ✓: runnable, ✗: not runnable.

### 5.3. Hyperparameter Tuning

To figure out the optimal model configuration, I performed experiments on the A2C view model to test the following hyperparameters:

- Loss function: MSE loss, L1 loss, smooth L1 loss
- Optimizer: Nesterov SGD, AdamW
- Learning rate:  $1e-6$ ,  $1e-5$ ,  $1e-4$
- L2 regularization:  $5e-4$ ,  $5e-3$

I first tested Nesterov’s accelerated SGD with 0.99 momentum, a learning rate of  $1e-6$ , and the MSE loss. It turned out that the training loss dropped very slowly, and there was no significant improvement even after the 20<sup>th</sup> epoch. However, when I increased the learning rate to  $1e-5$  or  $1e-4$ , there was a gradient explosion. I doubted this might be due to the large gradient of the MSE loss. Therefore, I changed the loss function to L1 loss and smooth L1 loss to see if they could mitigate gradient explosion.



However, the problem persisted. It was also possible that Nesterov's acceleration had some counter effects that led to gradient explosion on my network and dataset. So, I switched to the AdamW optimizer, which has an adaptive learning rate. Eventually, by setting an initial learning rate of  $1e-5$ , the training MSE loss could drop steadily. Besides, different values of L2 regularization did not seem to have a significant impact on the training loss, so I stuck to  $5e-4$  used by Payer *et al.* [12].



**Figure 6. Loss Plot after Hyperparameter Tuning.** The  $y$ -axis is the MSE loss. For the training loss, the  $x$ -axis is the number of iterations. For the validation loss, the  $x$ -axis is the number of epochs.

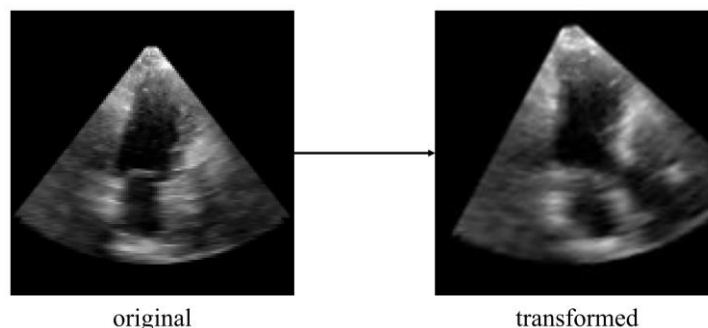
I trained a model for the A2C view under this optimal configuration for about 50 epochs, as shown in Figure 6. Note that the training loss was recorded for every iteration while the validation loss was recorded for every epoch, so their  $x$ -coordinates are not the same, but we can still compare their values at the same point of the training process from the figure. We can see that the training loss continues to drop to around 20 after 50 epochs, but the validation loss has reached its minimum of 35.11 at the 21<sup>st</sup> epoch and increases slightly in later epochs. The difference between the training and the

validation loss and the pattern of increase in the validation loss indicate the overfitting problem. The problem was possibly caused by the limited available training data, as there were only 241 training data for the A2C view (see Table 2). Therefore, I resorted to data augmentation to mitigate this problem.

## 5.4. Improvement with Data Augmentation

To solve the overfitting problem, I performed data augmentation that increased the training data size. The transformations performed on the input echo data are as follows:

1. Randomly zooming with a factor in  $[0.9, 1.1]$
2. Randomly shifting  $[-5, 5]$  voxels in the  $x$ ,  $y$ , and  $z$  axes
3. Randomly rotating  $[-15^\circ, 15^\circ]$  along the  $z$  axis and  $[-10^\circ, 10^\circ]$  along the  $x$  and  $y$  axes
4. Randomly multiplying the intensity with  $[0.75, 1.25]$  and then randomly adding  $[-0.25, 0.25]$

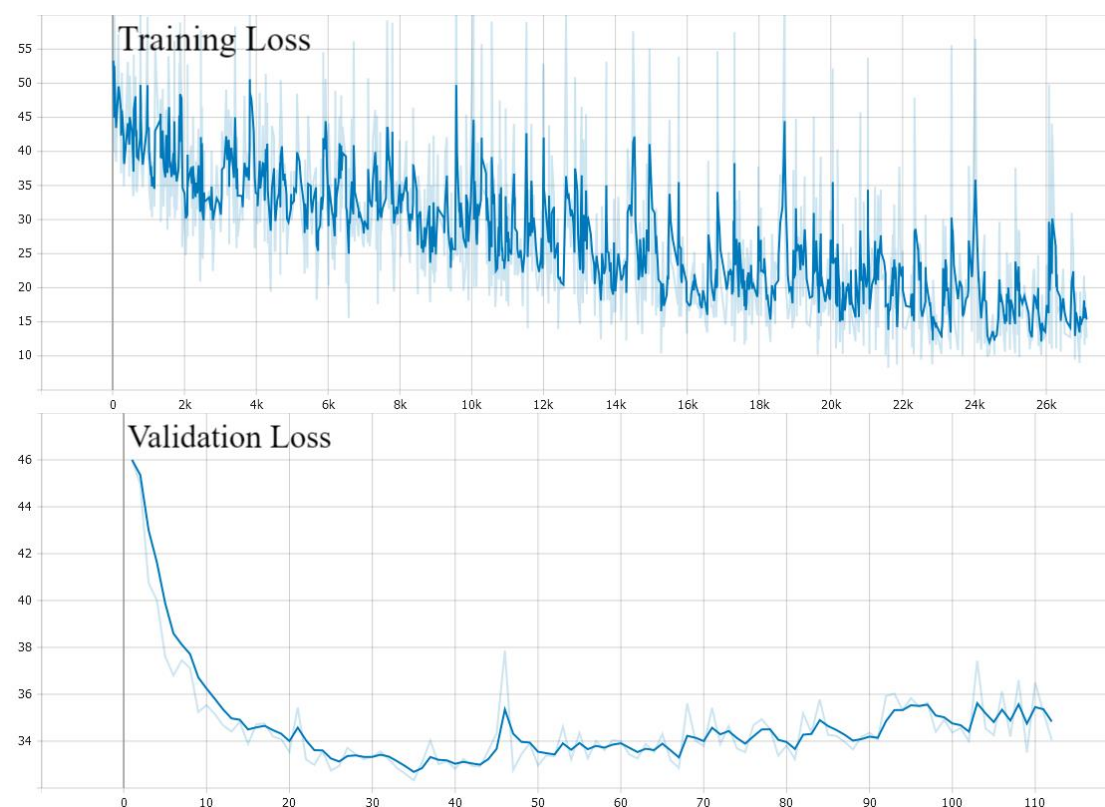


**Figure 7. Data Augmentation Example.** The two images are slices of 3D echo data. The left one is the original data. The right one is the transformed data by choosing the random parameters to extreme values.

The transforms were performed on-the-fly during training using background worker threads, which did not prolong the training time. The values used in each kind of transformation were carefully designed so that the central part of the echo data was still visible in the window, as shown in the example in Figure 7. Note that the rotational degree along the  $z$ -axis could be slightly larger than those of the other two axes because the echo data is pyramid-shaped and possesses some rotational symmetry along the  $z$ -

axis. Besides, the intensity values were manipulated after they were normalized to  $[-1, 1]$ . All random values were drawn from uniform distributions, and third-order spline interpolation was used to interpolate values.

I then trained a model with this data augmentation for 100 epochs. Figure 8 shows that the best validation loss is 32.33, achieved at the 35<sup>th</sup> epoch, which is improved from previous results. However, the overall trend still reveals the overfitting problem as the validation loss goes up from the best epoch to more than 36, much greater than the training loss of less than 20. Therefore, I decided to explore other methods to make the network learn better and further alleviate overfitting.



**Figure 8. Loss Plot with Data Augmentation.** The  $y$ -axis is the MSE loss. For the training loss, the  $x$ -axis is the number of iterations. For the validation loss, the  $x$ -axis is the number of epochs.

## 5.5. Improvement with Adaptive Wing Loss

A problem with the current heatmaps is that the foreground area, which contains a landmark, only accounts for a small portion of the entire heatmap, as shown in Figure

5. Therefore, identically calculating the loss of every voxel may misdirect the network to the unimportant background area. To solve the problem, I introduced the Adaptive Wing loss proposed by Wang *et al.* [13] to my training process.

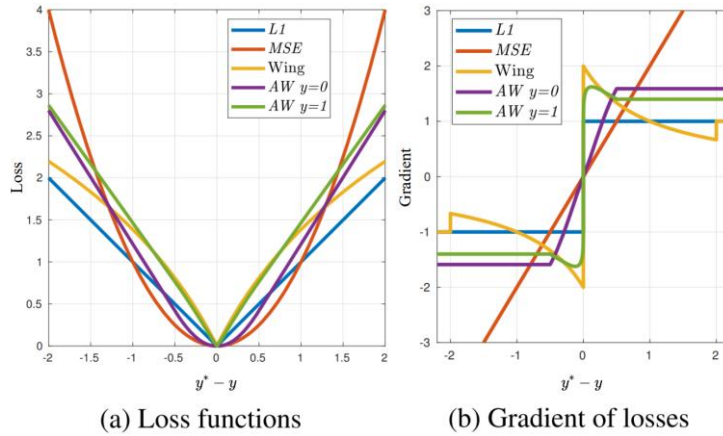
The Adaptive Wing loss [13] takes the form of

$$\text{AWing}(y, y^*) = \begin{cases} \omega \ln \left( 1 + \left| \frac{y^* - y}{\epsilon} \right|^{\alpha - y^*} \right), & \text{if } |y^* - y| < \theta \\ A|y^* - y| - C, & \text{otherwise} \end{cases}$$

where  $y$  is the prediction,  $y^*$  is the ground truth, and

$$A = \omega \left( \frac{1}{1 + \left(\frac{\theta}{\epsilon}\right)^{(\alpha - y^*)}} \right) (\alpha - y^*) \left( \left(\frac{\theta}{\epsilon}\right)^{(\alpha - y^* - 1)} \right) \left(\frac{1}{\epsilon}\right)$$

$$C = \left( \theta A - \omega \ln \left( 1 + \left(\frac{\theta}{\epsilon}\right)^{\alpha - y^*} \right) \right)$$



**Figure 9. Loss and Gradient of Loss Functions.** Retrieved from Wang *et al.* [13]. The  $x$ -axis is  $(y^* - y)$ . In (a), the  $y$ -axis is the loss. In (b), the  $y$ -axis is the gradient of the loss.

This is an adaptive version of the Wing loss [14], aiming to tackle the imbalance of foreground and background in heatmaps. When  $y^*$  is large, *i.e.*, close to a landmark, the Adaptive Wing loss behaves like a Wing loss, imposing a stronger gradient than the MSE loss, as shown in the green curve in Figure 9(b). But when  $y^*$  is small, *i.e.*, belongs to the background, the Adaptive Wing loss behaves like an MSE loss, as shown in the violet curve in Figure 9(b). This loss makes the network focus more on areas

close to landmarks and learn the foreground better.

Furthermore, Wang *et al.* [13] combined the Adaptive Wing loss with a weighted loss map mask in the form of

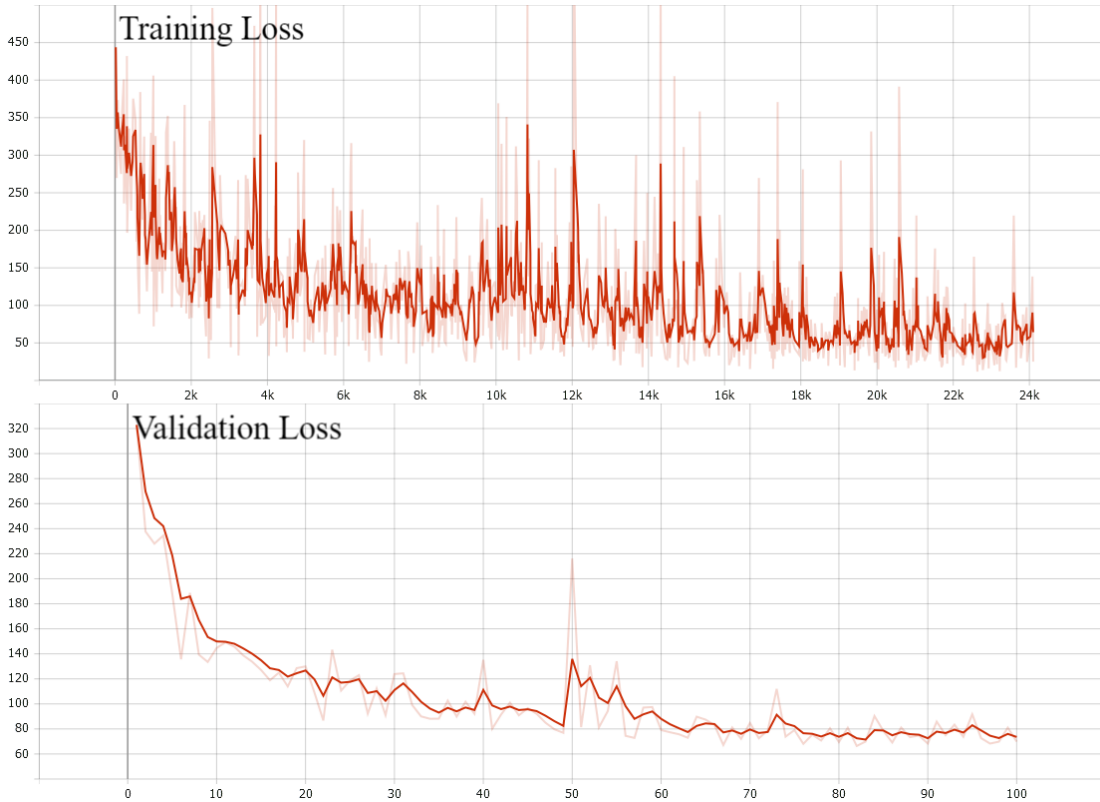
$$M = \begin{cases} 1, & \text{where } \mathbb{H}^d \geq 0.2 \\ 0, & \text{otherwise} \end{cases}$$

where  $\mathbb{H}^d$  is generated from the ground truth heatmap  $\mathbb{H}^*$  by a  $3 \times 3 \times 3$  gray dilation. And then, the weighted loss is calculated as

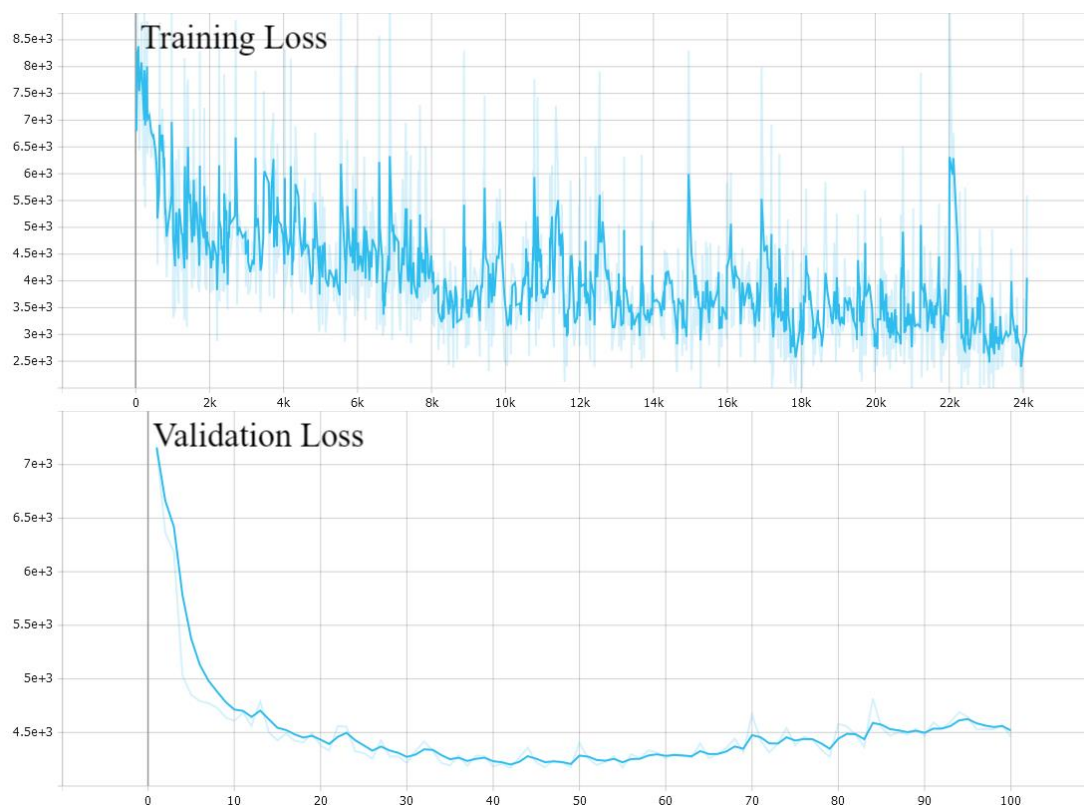
$$F_{weighted\ loss}(\mathbb{H}, \mathbb{H}^*) = F_{loss}(\mathbb{H}, \mathbb{H}^*) \otimes (w \cdot M + 1)$$

where  $\otimes$  is element-wise multiplication and  $w \in \mathbb{R}^+$  is a scalar hyperparameter to control how much weight to add to the foreground.

As a result, areas near a landmark further receive a loss of  $w + 1$  times the original one. This weighted loss puts more emphasis on the foreground region. It can enhance the effect of the Adaptive Wing loss in counterbalancing the fact that the foreground is relatively small compared to the background.



**Figure 10. Loss Plot with Adaptive Wing Loss.** The vertical axis is the Adaptive Wing loss. For the training loss, the horizontal axis is the number of iterations. For the validation loss, the horizontal axis is the number of epochs.



**Figure 11. Loss Plot with Adaptive Wing Loss and Weighted Loss Map Mask.** The vertical axis is the weighted Adaptive Wing loss. For the training loss, the horizontal axis is the number of iterations. For the validation loss, the horizontal axis is the number of epochs.

I tested the Adaptive Wing loss with and without the weighted loss map mask for 100 epochs, and the results are promising, as shown in Figures 10 and 11. Although the validation loss value cannot be compared to previous losses as the loss function has been changed, we can still see that for Adaptive Wing loss without the weighted loss map mask (Figure 10), the best epoch is achieved at the 82<sup>nd</sup> epoch, which is much later than the previous results, and there are almost no signs of overfitting as the validation loss is comparable to the training loss. The result is similar for Adaptive Wing loss with the weighted loss map mask (Figure 11). Still, it is noticeable that the network converges much faster when the loss is weighted, indicating that it can learn the foreground better. However, the validation loss again goes up at the end of the training, which may imply that the learning rate is too large and needs to be manually adjusted in later stages of training.

## 5.6. Improvement with Learning Rate Decay

To avoid the fluctuations and increases in the loss that occurred in later stages of training, I further introduced a learning rate scheduler to the trainer, which automatically decreased the learning to  $1/10$  after every 30 epochs because I noticed that the validation loss reached its minimum in around 30 epochs in previous experiments. With an initial learning rate of  $1e-5$ , the scheduler dropped the learning rate to  $1e-6$ ,  $1e-7$ , and  $1e-8$  at the 30<sup>th</sup>, 60<sup>th</sup>, and 90<sup>th</sup> epoch, respectively. As shown in Figure 12, the validation loss is much more stable and convergent than that in Figure 11, as the loss no longer goes up or fluctuates at the end of training. The final loss of around  $4.2e3$  is about the same as the training loss and much lower than the previous  $4.5e3$  in Figure 10. Such a pattern appearing in the validation loss indicates that the training process is stable and reliable.



**Figure 12. Loss Plot with Learning Rate Decay.** The vertical axis is the weighted Adaptive Wing loss. For the training loss, the horizontal axis is the number of iterations. For the validation loss, the horizontal axis is the number of epochs.



## 5.7. Improvement Analysis

To prove the effectiveness of my improvements, I compared the performance of the previous models trained on the A2C view at the 50<sup>th</sup> epoch on the test dataset. To convert a predicted heatmap to predicted landmarks, I extracted the coordinate of the maximum value on the  $i$ -th channel of the predicted heatmap as the predicted coordinate  $\mathbf{x}_i \in \mathbb{R}^3$  of landmark  $L_i$ , as follows:

$$\mathbf{x}_i = \operatorname{argmax} \mathbb{H}_i \text{ for } i \in \{1, \dots, N\}$$

where  $N$  is the number of landmarks on the A2C view, which also equals the number of channels in  $\mathbb{H}$ .

Then I calculated  $\mathbf{x}_i$ 's Euclidean distances to their corresponding ground truth landmark coordinates  $\mathbf{x}_i^*$ 's. Let  $S$  be the amount of data in the testing dataset and  $j \in \{1, \dots, S\}$ , then  $\mathbf{x}_i^{(j)} \in \mathbb{R}^3$  and  $\mathbf{x}_i^{*(j)} \in \mathbb{R}^3$  are just the predicted and ground truth landmark coordinates of landmark  $L_i$  on the  $j$ -th testing echo data, respectively. The point-to-point error  $E_{p2p}^{(j)}$  of the  $j$ -th testing data is defined as:

$$E_{p2p}^{(j)} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i^{*(j)} - \mathbf{x}_i^{(j)}\|$$

The median, mean, and standard deviation of all  $E_{p2p}^{(j)}$  on the testing dataset are shown in Table 4. Remarkably, the model trained with data augmentation, weighted Adaptive Wing loss, and learning rate decay achieved the best mean error 4.39 with the lowest standard deviation 4.10. In fact, we can see decrements in the mean  $E_{p2p}$  after each improvement method is implemented. From method §5.4 to method §5.6, the mean  $E_{p2p}$  drops by 4.64%, 10.69%, and 11.49% compared to method §5.3. And despite a slight increase in the standard deviation of  $E_{p2p}$  in method §5.4, its best value in method §5.6 is still 14.94% better than that of method §5.3. It indicates that my model was more accurate and stable after these improvements. Additionally, it is also noticeable that method §5.6's median  $E_{p2p}$  is not as good as that of method §5.4, but it does not necessarily mean that method §5.6 performs worse. Because we can see that the distributions of  $E_{p2p}$  in all methods are all skewed to the right as the median  $E_{p2p}$



is smaller than the mean  $E_{p2p}$ . Such skewness indicates that the model performs well on most data but has large  $E_{p2p}$  on some data, which may be a sign of overfitting. The shortening of the gap between the median and mean  $E_{p2p}$  may imply that my improvements increased the model's generalizability, which echoes the observation in Section 5.6. Therefore, I believe that the improvements were effective.

**Table 4. Point-to-point Error for Improvement Analysis**

Method	$E_{p2p}$ (in voxel)	
	Median	Mean $\pm$ SD
§5.3	3.45	4.96 $\pm$ 4.82
§5.4	<b>3.13</b>	4.73 $\pm$ 4.95
§5.5	3.21	4.43 $\pm$ 4.18
§5.6	3.30	<b>4.39 <math>\pm</math> 4.10</b>

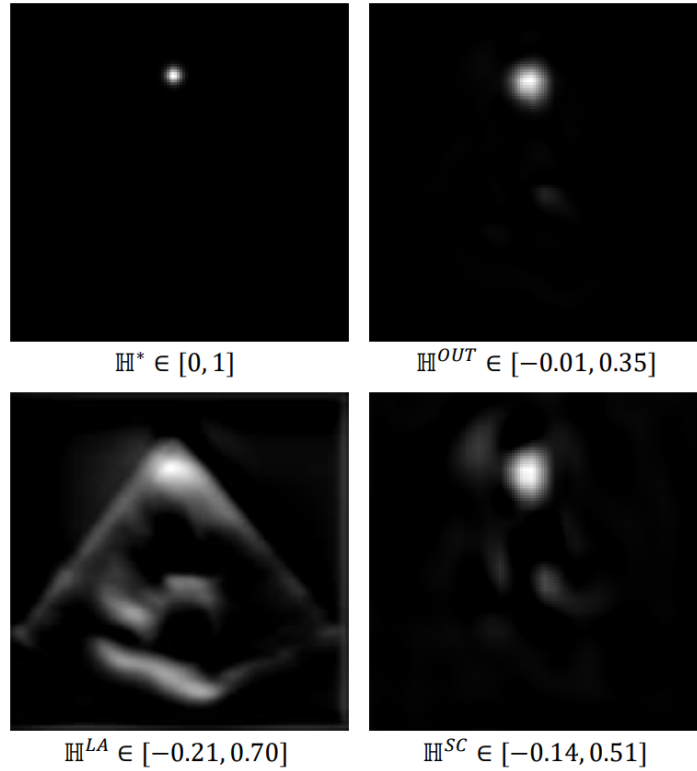
§5.3: model with MSE loss, §5.4: model with MSE loss and data augmentation, §5.5: model with weighted Adaptive Wing loss and data augmentation, §5.6: model with weighted Adaptive Wing loss, data augmentation, and learning rate decay. The best values are in **bold**.

Furthermore, to visually evaluate the performance of the model, I rendered the predicted heatmaps from the best-performing model, an example of which is shown in Figure 13. Here, the local appearance heatmap  $\mathbb{H}^{LA}$  and the spatial configuration heatmap  $\mathbb{H}^{SC}$  were extracted from the last intermediate layers of the model. Since the activation function of  $\mathbb{H}^{LA}$  and  $\mathbb{H}^{SC}$  are linear and TanH, respectively, the output heatmaps may contain negative values. If I directly map the value range to  $[0, 255]$ , the heatmaps may contain large grey areas. Therefore, I clamped the negative values to 0 and rescaled positive values to  $[0, 255]$  for better visualization, as the negative values do not help predict landmarks.

Furthermore, since the activation functions of  $\mathbb{H}^{LA}$  and  $\mathbb{H}^{SC}$  are symmetric about the origin, if the network inverts them simultaneously, the final output heatmap  $\mathbb{H}^{OUT}$  or  $\mathbb{H} := \mathbb{H}^{LA} \otimes \mathbb{H}^{SC}$  keeps the same. Therefore, I needed to judge whether to invert  $\mathbb{H}^{LA}$  and  $\mathbb{H}^{SC}$  for visualization based on the absolute values of the peaks and

troughs of the intensities. If the absolute value of the trough is larger than that of the peak, the heatmap probably needs to be inverted before going through the clamp-and-rescale process. The whole procedure can be summarized as:

1.  $inverted \mathbb{H} = \begin{cases} \mathbb{H}, & \text{if } |\max(\mathbb{H})| \geq |\min(\mathbb{H})| \\ -\mathbb{H}, & \text{otherwise} \end{cases}$  (only invert  $\mathbb{H}^{LA}, \mathbb{H}^{SC}$ )
2.  $clamped \mathbb{H}_{i,(x,y,z)} = \begin{cases} inverted \mathbb{H}_{i,(x,y,z)}, & \text{if } inverted \mathbb{H}_{i,(x,y,z)} > 0 \\ 0, & \text{otherwise} \end{cases}$
3.  $rendered \mathbb{H} = \frac{255 \cdot clamped \mathbb{H}}{\max(clamped \mathbb{H})}$



**Figure 13. Heatmap Visualization Example.** The images are 2D slices of the same channel of the heatmaps.  $\mathbb{H}^*$ : ground truth heatmap,  $\mathbb{H}^{OUT}$ : predicted heatmap,  $\mathbb{H}^{LA}$ : local appearance heatmap,  $\mathbb{H}^{SC}$ : spatial configuration heatmap. The original value ranges are shown below each heatmap.

We can see that the bright area on  $\mathbb{H}^{LA}$  is distributed across the heatmap as it considers every local point that can be the landmark. Meanwhile, on  $\mathbb{H}^{SC}$ , the bright area is centralized to one large region as it is the most likely region to contain the landmark. When  $\mathbb{H}^{LA}$  and  $\mathbb{H}^{SC}$  are combined,  $\mathbb{H}^{SC}$  discriminates those false positive areas on  $\mathbb{H}^{LA}$  and only retains one prominent area. The final output heatmap

$\mathbb{H}^{OUT}$  resembles the ground truth heatmap  $\mathbb{H}^*$  as their peaks lie in the same area. This pattern proves that my network worked as designed.

**Table 5. Number of Parameters and MACCs**

View	Landmarks	Params	MACCs
A2C	3	16.83 M	3588.90 G
A4C	6	17.1 M	3619.3 G
ALAX	4	16.92 M	3599.03 G
SAXA	3	16.83 M	3588.90 G
SAXB	4	16.92 M	3599.03 G
SAXM	6	17.1 M	3619.3 G
SAXMV	6	17.1 M	3619.3 G

M:  $\times 10^6$ , G:  $\times 10^9$ .

The numbers of parameters and MACCs of the models for different views are also shown in Table 5. These statistics are calculated on the final structure with an input size of  $128 \times 128 \times 128$  and 128 channels in each intermediate layer. The different number of landmarks on different views affects the number of parameters and MACCs in that the number of convolutions in the output layers is different.

## 6. Cross-section Recovery

### 6.1. Method Choice

As shown in Table 1, some cross-sections may have more than three landmarks. Therefore, I may or may not use all predicted landmarks to fit the cross-section plane. To find the best way to recover the cross-sections, I have proposed three methods:

1. **Least Square Fit:** Perform singular value decomposition on all predicted landmarks to fit a plane with the least squared distance to all landmarks.

The centroid  $\mathbf{c} \in \mathbb{R}^3$  that lies on the plane is calculated as:

$$\mathbf{c} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$$

The unit normal vector  $\mathbf{n} \in \mathbb{R}^3$  where  $\|\mathbf{n}\| = 1$  is calculated as:

$$U\Sigma V^T \stackrel{\text{SVD}}{=} \begin{bmatrix} \mathbf{x}_1 \\ \dots \\ \mathbf{x}_N \end{bmatrix} - \begin{bmatrix} \mathbf{c} \\ \dots \\ \mathbf{c} \end{bmatrix}$$

$$\mathbf{n} = v_3 / \|v_3\| \text{ where } [v_1 \ v_2 \ v_3] = V$$

2. **Top Confidence Fit:** Interpret the intensity value of the brightest voxel on the predicted heatmap as its confidence level, and select the three landmarks with the top confidence levels to fit a plane. Note that this fit is deterministic, so any fitting methods are equivalent. In practice, I just used the Least Square Fit.
3. **RANSAC Fit:** Run the random sample consensus algorithm [15] to randomly find the best fitting plane. There are four parameters in the RANSAC algorithm:
  - $n$ : the minimum number of landmarks required to fit a plane.
  - $k$ : the maximum number of iterations allowed in the algorithm.
  - $t$ : a threshold determining whether a landmark is close to a plane, which is calculated as the squared distance to the plane  $((\mathbf{x}_i - \mathbf{c}) \cdot \mathbf{n})^2$  in this case.
  - $d$ : the number of additional close landmarks required to assert that a plane fits well to the landmarks.

For preliminary evaluation, I started with a relaxed condition using  $n = 3$ ,  $k = 100$ ,  $t = 0.5$ ,  $d = 1$ , which requires only 4 landmarks within 0.5 squared distance to the plane to have consensus. Again, singular value decomposition is

used when fitting to a subset of landmarks.

To test the effects of different methods, I recovered cross-sections from the model trained on the A4C view, as it contains six landmarks. Since this postprocessing method selection does not involve machine learning techniques, I trained the model using all training and validation data and tested the performance on the testing data. The recovered cross-section is represented by two parameters  $(\mathbf{c}, \mathbf{n})$  where  $\mathbf{c}$  is a point on the plane and  $\mathbf{n}$  is a unit normal vector of the plane. I used two metrics to measure the difference between the predicted plane  $\Pi := (\mathbf{c}, \mathbf{n})$  and the ground truth plane  $\Pi^* := (\mathbf{c}^*, \mathbf{n}^*)$ .

The first metric is  $E_{shift}$ , which measures the displacement from the ground truth  $\mathbf{c}^*$  to the predicted  $\mathbf{c}$  projected onto the unit ground truth normal  $\mathbf{n}^*$  where  $\|\mathbf{n}^*\| = 1$ , as follows:

$$E_{shift} = \|((\mathbf{c} - \mathbf{c}^*) \cdot \mathbf{n}^*)\mathbf{n}^*\|$$

The second metric is  $E_{angle}$ , which measures the angle between the  $\mathbf{n}^*$  and  $\mathbf{n}$ , as follows:

$$E_{angle} = \begin{cases} \cos^{-1}(\mathbf{n} \cdot \mathbf{n}^*), & \text{if } \cos^{-1}(\mathbf{n} \cdot \mathbf{n}^*) \leq 90^\circ \\ 180^\circ - \cos^{-1}(\mathbf{n} \cdot \mathbf{n}^*), & \text{otherwise} \end{cases}$$

Note that  $E_{angle}$  is always in the range  $[0^\circ, 90^\circ]$  no matter the directions of the two normal vectors.

Additionally, I designed another metric  $E_{fit}$  that measures the mean squared distance from predicted landmarks  $(\mathbf{x}_1, \dots, \mathbf{x}_N)$  to the predicted plane  $(\mathbf{c}, \mathbf{n})$ , as follows:

$$E_{fit} = \frac{1}{N} \sum_{i=1}^N ((\mathbf{x}_i - \mathbf{c}) \cdot \mathbf{n})^2$$

Note that  $E_{fit}$  measures how well the predicted cross-section plane fits the predicted landmarks.

For each echo data in the testing dataset, I obtained the predicted landmarks  $(\mathbf{x}_1, \dots, \mathbf{x}_N)$  from the predicted heatmaps by extracting the maximum points following  $\mathbf{x}_i = \operatorname{argmax} \mathbb{H}_i$ . And then, I used one of the three recovery methods listed above to obtain  $(\mathbf{c}, \mathbf{n})$ . For the ground truth cross-section plane  $\Pi^*$ , I just used the Least Square

Fit method to obtain  $(\mathbf{c}^*, \mathbf{n}^*)$  from the ground truth landmarks location  $(\mathbf{x}_1^*, \dots, \mathbf{x}_N^*)$ . Since the ground truth landmarks were just picked from the ground truth cross-section by annotators, the ground truth fit always had  $E_{fit} = 0$ .

Then, I could obtain  $E_{shift}^{(j)}$ ,  $E_{angle}^{(j)}$  from  $(\mathbf{c}^{(j)}, \mathbf{n}^{(j)})$ ,  $(\mathbf{c}^{*(j)}, \mathbf{n}^{*(j)})$  and  $E_{fit}^{(j)}$  from  $(\mathbf{c}^{(j)}, \mathbf{n}^{(j)})$ ,  $(\mathbf{x}_1^{(j)}, \dots, \mathbf{x}_N^{(j)})$  for the  $j$ -th echo data in the testing dataset where  $j \in \{1, \dots, S\}$  and  $S$  is the size of the testing dataset. At last, statistics were obtained from  $(E_{shift}^{(1)}, \dots, E_{shift}^{(S)})$ ,  $(E_{angle}^{(1)}, \dots, E_{angle}^{(S)})$ , and  $(E_{fit}^{(1)}, \dots, E_{fit}^{(S)})$ , as shown in Table 6.

**Table 6. Recovery Methods Comparison**

A4C	$E_{shift}$		$E_{angle}$		$E_{fit}$	
	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD
<b>Least Square Fit</b>	<b>1.45</b>	<b>1.53 <math>\pm</math> 0.97</b>	<b>1.87</b>	<b>2.48 <math>\pm</math> 2.60</b>	0.11	<b>0.61 <math>\pm</math> 1.29</b>
<b>Top Confidence Fit</b>	1.58	1.68 $\pm$ 1.09	10.78	11.33 $\pm$ 4.18	<b>0.00</b>	51.38 $\pm$ 271.87
<b>RANSAC Fit</b>	1.68	1.74 $\pm$ 1.29	8.00	7.73 $\pm$ 3.89	<b>0.00</b>	13.16 $\pm$ 82.22

The best values are in **bold**.

We can see that the Least Square Fit is clearly the optimal method, obtaining the best result in all statistics of the three metrics (note that the median of  $E_{fit}$  is not comparable because the Top Confidence Fit and the RANSAC Fit only fit part of the landmarks). Although the difference of  $E_{shift}$  is not so large across different methods, the mean  $E_{shift}$  of the Least Square Fit is still 8.93% and 12.07% better than that of the Top Confidence Fit and the RANSAC Fit, respectively. Meanwhile,  $E_{angle}$  of the Least Square Fit is much better than the other two, with only an error of around 2°. The mean  $E_{angle}$  of the Least Square Fit is 78.11% and 69.92% better than that of the Top Confidence Fit and the RANSAC Fit, respectively.

The RANSAC Fit is unsuitable for this task because the maximum number of

landmarks on a cross-section is only six, so it is unlikely for the algorithm to distinguish outliers from inliers through random selection. The Confidence Fit performed the worst on recovering the normal vector probably because the brightness of the voxel does not carry explicit information about its confidence level, and the importance of all landmarks does not differ too much in determining the cross-section. Therefore, the Least Square Fit method was chosen to recover cross-sections in the following tasks.

## 6.2. Handling SAXB

The SAXB view contains four landmarks, including the Center of AV ( $L_1^{(SAXB)}$ ), the IAS ( $L_2^{(SAXB)}$ ), the SAXB-TV tip ( $L_3^{(SAXB)}$ ), and the PV tip ( $L_4^{(SAXB)}$ ). However, the PV tip is not labeled in most training data. In fact, only one training data from 2 November 2021 contains the annotation of the PV tip, which means that the model could barely learn anything about the location of the PV tip. As a result, including the PV tip in the recovery process resulted in poor performance, as shown in the first row of Table 7. Therefore, I chose to ignore the predicted location of the PV tip and only use the Center of AV, the IAS, and the SAXB-TV tip to fit the SAXB view, *i.e.*,  $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ . The result is shown in the second row of Table 7. As expected, excluding the useless information made both  $E_{shift}$  and  $E_{angle}$  better with improvements of 20% to 30% on both the median and the mean.

**Table 7. Improvement on SAXB Recovery**

SAXB	$E_{shift}$		$E_{angle}$	
	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD
<b>Before</b>	2.24	2.98 $\pm$ <b>2.68</b>	14.72	21.31 $\pm$ 17.64
<b>After</b>	<b>1.66</b>	<b>2.36</b> $\pm$ 2.72	<b>11.76</b>	<b>14.37</b> $\pm$ <b>8.96</b>

**Before:** use the predicted locations of the Center of AV, the IAS, the SAXB-TV tip, and the PV tip to fit the SAXB cross-section plane; **After:** use the predicted locations of the Center of AV, the IAS, and the SAXB-TV tip to fit the SAXB cross-section plane. The best values are in **bold**.

### 6.3. Handling SAXA, SAXM, and SAXMV

When using the Least Square Fit to recover the SAXA view, I found that  $E_{angle}$  was exceptionally large, with the median value being around  $60^\circ$ , while  $E_{shift}$  was comparable to those of other cross-section views, as shown in the first row of Table 8. After a closer examination of the landmarks of SAXA: the SAXA-LV apex ( $L_1^{(SAXA)}$ ), the Interventricular septum ( $L_2^{(SAXA)}$ ), and the RV apex ( $L_3^{(SAXA)}$ ), I noticed that their ground truth locations are almost colinear, so a slight shift in any of the predicted landmark locations would result in a significant degree of rotation of the predicted SAXA plane.

To solve this problem, I explored alternative definitions of the SAXA view and found that the SAXA, SAXM, and SAXMV views are defined to be mutually parallel. Therefore, I could use the collective landmark information from the SAXA, SAXM, and SAXMV views to predict the normal vector of the SAXA view, as these landmarks are no longer colinear. To achieve this purpose, I first translated the centroids of the predicted landmarks of SAXA, SAXM, and SAXMV to the origin, respectively:

$$centered \mathbf{x}_i^{(VIEW)} = \mathbf{x}_i^{(VIEW)} - \frac{1}{N^{(VIEW)}} \sum_{j=1}^{N^{(VIEW)}} \mathbf{x}_j^{(VIEW)}$$

for  $VIEW \in \{SAXA, SAXM, SAXMV\}$

And then, I concatenated the translated landmark locations together as

$$centered \mathbf{X}^{(SAXA, SAXM, SAXMV)} = \left\{ centered \mathbf{x}_i^{(SAXA)}, centered \mathbf{x}_j^{(SAXM)}, centered \mathbf{x}_k^{(SAXMV)} \right\}$$

for  $i \in \{1, \dots, N^{(SAXA)}\}$ ,  $j \in \{1, \dots, N^{(SAXM)}\}$ ,  $k \in \{1, \dots, N^{(SAXMV)}\}$ , which consisted of predicted locations of all 15 landmarks on the three cross-sections and performed singular value decomposition collectively on  $centered \mathbf{X}^{(SAXA, SAXM, SAXMV)}$ :

$$U \Sigma V^T \stackrel{SVD}{=} \begin{bmatrix} \mathbf{x}_1 \\ \dots \\ \mathbf{x}_N \end{bmatrix} \text{ where } \mathbf{x}_i \in centered \mathbf{X}^{(SAXA, SAXM, SAXMV)}$$

$$\mathbf{n}^{(SAXA)} = v_3 / \|v_3\| \text{ where } [v_1 \ v_2 \ v_3] = V$$

The result would be a plane  $(\mathbf{0}, \mathbf{n}^{(SAXA)})$  that has the least squared error over  $centered \mathbf{X}^{(SAXA, SAXM, SAXMV)}$ .



At last, I calculated the  $\mathbf{c}^{(SAXA)}$  as usual, using only SAXA's landmarks  $\mathbf{X}^{(SAXA)} = \{\mathbf{x}_1^{(SAXA)}, \dots, \mathbf{x}_{N^{(SAXA)}}^{(SAXA)}\}$  and shifted the plane back to  $\mathbf{c}^{(SAXA)}$ :

$$\mathbf{c}^{(SAXA)} = \frac{1}{N^{(SAXA)}} \sum_{i=1}^{N^{(SAXA)}} \mathbf{x}_i^{(SAXA)}$$

As a result, I could obtain the predicted SAXA plane  $\Pi^{(SAXA)} = (\mathbf{c}^{(SAXA)}, \mathbf{n}^{(SAXA)})$ .

For calculating the ground truth plane of SAXA, a similar approach was adopted by fitting the ground truth normal vector using  $centered \mathbf{X}^{*(SAXA, SAXM, SAXMV)}$  because some of the ground truth landmark locations of the SAXA view are exactly colinear as well. This approach is feasible because the SAXA, SAXM, and SAXMV views were labeled at the same time index and on the same echo data. However, it is noticeable that some echo data labeled with SAXA's landmarks were not labeled with SAXM's or SAXMV's landmarks. In such cases, I used the ground truth landmark locations of only two views to calculate the normal vector, *i.e.*,  $centered \mathbf{X}^{*(SAXA, SAXM)}$  or  $centered \mathbf{X}^{*(SAXA, SAXMV)}$ .

The result after this postprocessing is shown in the second row of Table 8.  $E_{shift}$  is the same, as its prediction method was not changed. We can see that the median of  $E_{angle}$  improved significantly to  $5.43^\circ$ , which is pretty small compared to the unreliable value before. Such improvement proved the effectiveness of my approach.

**Table 8. Improvement on SAXA Recovery**

SAXA	$E_{shift}$		$E_{angle}$	
	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD
<b>Before</b>	1.96	$2.85 \pm 3.10$	59.09	$54.54 \pm 28.77$
<b>After</b>	1.96	$2.85 \pm 3.10$	<b>5.43</b>	<b><math>7.29 \pm 6.67</math></b>

**Before:** use the predicted locations of landmarks on the SAXA view to fit the normal vector of the SAXA cross-section plane; **After:** use the predicted locations of landmarks on the SAXA, SAXM, and SAXMV views to fit the normal vector of the SAXA cross-section plane. The best values are in **bold**.

Although the SAXM and SAXMV views do not suffer from the collinearity

problem, it is presumed that utilizing the additional mutually parallel information among the SAXA, SAXM, and SAXMV views would improve the recovery results. Therefore, similar operations were performed on SAXM and SAXMV by using *centered*  $\mathbf{X}^{(SAXA,SAXM,SAXMV)}$  to recover the predicted normal vector. However, the ground truth normal vectors of SAXM and SAXMV were computed using only  $\mathbf{X}^{(SAXM)}$  and  $\mathbf{X}^{(SAXMV)}$  respectively because their locations do not suffer from the colinear problem, and hence the computation is accurate. The results are shown in Table 9. Here, only  $E_{angle}$  is shown for comparison since the two methods only differ in predicting the normal. We can observe that on both views, the results obtained using the SAXA, SAXM, and SAXMV's collective information are much better and stabler, as there are improvements in the median, mean, and standard deviation in both cases. For SAXM, the median and mean have been improved by 18.50% and 16.02%, and the standard deviation has also decreased by 41.40%. For SAXMV, the results are even better, with improvements of 46.97%, 34.68%, and 36.84% for the median, mean, and standard deviation, respectively. Therefore, this method was adopted to recover the SAXM and SAXMV views in the final evaluation.

**Table 9. Improvement on SAXM and SAXMV Recovery**

SAXM	$E_{angle}$		SAXMV	$E_{angle}$	
	Median	Mean $\pm$ SD		Median	Mean $\pm$ SD
<b>Before</b>	7.35	9.05 $\pm$ 11.28	<b>Before</b>	11.56	11.42 $\pm$ 7.41
<b>After</b>	<b>5.99</b>	<b>7.60 <math>\pm</math> 6.61</b>	<b>After</b>	<b>6.13</b>	<b>7.46 <math>\pm</math> 4.68</b>

**Before:** use the predicted locations of landmarks on the SAXM/SAXMV view to fit the normal vector of the SAXM/SAXMV cross-section plane; **After:** use the predicted locations of landmarks on the SAXA, SAXM, and SAXMV views to fit the normal vector of the SAXM/SAXMV cross-section plane. The best values are in **bold**.

## 7. Evaluation

### 7.1. Landmark Localization

To directly evaluate the models trained for the seven cross-sections on the testing dataset, I first considered the points-to-points error  $E_{p2p}$  on the predicted locations of the landmarks, as described in Section 5.7. In addition to the distance calculated in voxel side length, I also recovered the distance in millimeters  $E_{realp2p}$  according to the scaling ratio of the echo data recorded during preprocessing, as discussed in Section 3.2.3:

$$E_{realp2p}^{(VIEW,j)} = \frac{1}{N^{(VIEW)}} \sum_{i=1}^{N^{(VIEW)}} r^{(VIEW,j)} \left\| \mathbf{x}_i^{*(VIEW,j)} - \mathbf{x}_i^{(VIEW,j)} \right\|$$

where  $VIEW$  is a cross-section view;  $j$  is the  $j$ -th data in  $VIEW$ 's testing dataset;  $(VIEW, j)$  represents a specific testing data;  $N^{(VIEW)}$  is the number of landmarks on  $VIEW$ , which is the same for all  $j$ ;  $r^{(VIEW,j)}$  is the ratio of the real data to the resized data for the testing data  $(VIEW, j)$ ;  $\mathbf{x}_i^{*(VIEW,j)}$  is the ground truth coordinate of landmark  $L_i^{(VIEW)}$  on the testing data, and  $\mathbf{x}_i^{(VIEW,j)}$  is the predicted coordinate of landmark  $L_i^{(VIEW)}$  on the testing data.

**Table 10. Landmark Localization Evaluation**

View (#landmarks)	$E_{p2p}$ (in voxel)		$E_{realp2p}$ (in mm)	
	Median	Mean $\pm$ SD	Median	Mean + SD
<b>A2C (3)</b>	3.27	4.42 $\pm$ 4.26	5.35	7.47 $\pm$ 7.66
<b>A4C (6)</b>	2.94	3.91 $\pm$ <b>3.48</b>	<b>4.76</b>	6.70 $\pm$ 7.05
<b>ALAX (4)</b>	3.08	4.02 $\pm$ 3.91	4.84	<b>6.69 <math>\pm</math> 6.67</b>
<b>SAXA (3)</b>	3.51	4.77 $\pm$ 4.84	5.83	8.30 $\pm$ 10.29
<b>SAXB (4)</b>	3.49	4.96 $\pm$ 4.61	5.86	8.63 $\pm$ 9.40
<b>SAXM (6)</b>	3.03	4.28 $\pm$ 4.60	5.40	7.50 $\pm$ 10.20
<b>SAXMV (6)</b>	<b>2.86</b>	<b>3.90 <math>\pm</math> 3.99</b>	4.81	6.82 $\pm$ 8.77

The best values are in **bold**.

The results are shown in Table 10. Although the scaling ratios of individual data are different, we can see that the statistics of  $E_{realp2p}$  are approximately 1.6 times those of  $E_{p2p}$ . To evaluate the models' performance in real life, we shall focus on the statistics of  $E_{realp2p}$  and observe that the results on different cross-sections are similar, which indicates that the network worked stably on different cross-section views. The mean  $E_{realp2p}$  ranges from 6.69 mm to 8.63 mm, which is approximately 3% to 4% of the side length of the echo data. It is noticeable that the model of the ALAX view achieves both the minimum mean and the minimum standard deviation on  $E_{realp2p}$ , which may reveal that the structural features of landmarks on the ALAX view are more easily distinguishable. On the contrary, the models of SAXA and SAXB have poorer performance, possibly because their landmarks are more concentrated and harder to recognize. Furthermore, it is noticeable that the median  $E_{realp2p}$  is smaller than mean  $E_{realp2p}$  for all the models, which may imply that the network suffers from high errors on a small amount of the testing data. Nonetheless, the localization results are within my expectations and are basically satisfactory.

## 7.2. Cross-section Recovery

To analyze the quality of the reconstructed cross-sections. I used the same metrics  $E_{shift}$  and  $E_{angle}$  described in Section 6.1. For  $E_{shift}$ , I further recovered the distance in millimeters  $E_{realshift}$  similarly to  $E_{realp2p}$ :

$$E_{realshift}^{(VIEW,j)} = r^{(VIEW,j)} \left\| \left( \mathbf{c}^{(VIEW,j)} - \mathbf{c}^{*(VIEW,j)} \right) \cdot \mathbf{n}^{(VIEW,j)} \right\|$$

where  $r^{(VIEW,j)}$  is the ratio of the real data to the resized data for the testing data  $(VIEW, j)$ .

The results are shown in Table 11. Firstly, we can see that, like  $E_{realp2p}$ ,  $E_{realshift}$  does not differ too much across different cross-section views. The mean  $E_{realshift}$  ranges from 1.93 mm to 2.98 mm, with its best value attained on the ALAX view again. This  $E_{realshift}$  indicates that the centroid of the predicted landmarks is at most 2.98 mm away from the centroid of the ground truth landmarks along the ground truth normal direction, which is not so large considering that the side length of the echo data

is almost 300 mm. Secondly,  $E_{angle}$  is quite different across different cross-section views, with a minimum mean of  $6.38^\circ$  and a maximum mean of  $14.37^\circ$ . For the cross-sections fitted using three landmarks (*i.e.*, A2C and SAXB; recall that SAXB does not use the PV tip), the median and mean are relatively large, being around  $12^\circ$  and  $14^\circ$ , respectively. For the cross-section fitted using four landmarks (*i.e.*, ALAX), the median and mean are around  $9^\circ$  and  $11^\circ$ , respectively, which are comparably smaller. While for the cross-sections fitted using more than or equal to six landmarks (*i.e.*, A4C, SAXA, SAXM, and SAXMV; recall that SAXA, SAXM, and SAXMV use all 15 landmarks to fit their cross-sections), the median and mean are even smaller, which are approximately  $5^\circ$  and  $7^\circ$ , respectively. From this pattern, we can conclude that  $E_{angle}$  is negatively correlated to the number of predicted landmarks used for fitting, which implies that the algorithm can achieve its best performance on cross-sections containing a relatively large number of identifiable landmarks.

**Table 11. Cross-section Recovery Evaluation**

View (#landmarks)	$E_{shift}$ (in voxel)		$E_{realshift}$ (in mm)		$E_{angle}$ (in degree)	
	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD
A2C (3)	1.51	$1.84 \pm 1.70$	2.30	$3.18 \pm 3.30$	11.64	$13.60 \pm 10.96$
A4C (6)	1.44	$1.53 \pm \mathbf{0.97}$	2.29	$2.55 \pm \mathbf{1.76}$	<b>5.03</b>	$\mathbf{6.38} \pm 4.69$
ALAX (4)	<b>1.14</b>	$\mathbf{1.49} \pm 1.81$	<b>1.93</b>	$\mathbf{2.45} \pm 2.95$	8.53	$11.45 \pm 12.19$
SAXA (3)	1.96	$2.85 \pm 3.10$	2.98	$4.91 \pm 5.94$	5.43	$7.29 \pm 6.67$
SAXB (4)	1.66	$2.37 \pm 2.73$	2.66	$4.05 \pm 5.32$	11.76	$14.37 \pm 8.97$
SAXM (6)	1.76	$2.38 \pm 3.78$	2.85	$4.30 \pm 8.61$	5.99	$7.60 \pm 6.61$
SAXMV (6)	1.30	$2.17 \pm 3.82$	2.03	$3.97 \pm 8.44$	6.13	$7.46 \pm \mathbf{4.68}$

The best values are in **bold**.

### 7.3. Baselines

This sub-section compares my results with two other baseline networks: the Patch-based Fully Convolutional Neural Network (FCNN) [10] and the Down-sampling Net (DSN) [9], which were implemented by other members in our group, and the details

shall be found in their respective reports.

**Table 12. Baseline Comparison of Planar Position**

View (#landmarks)	$E_{realshift}$ (in mm)					
	SCN (My)		FCNN		DSN	
	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD
A2C (3)	<b>2.30</b>	<b>3.18 <math>\pm</math> 3.30</b>	3.73	4.94 $\pm$ 4.70	3.17	4.21 $\pm$ 3.89
A4C (6)	2.29	2.55 $\pm$ 1.76	<b>1.58</b>	<b>2.52 <math>\pm</math> 2.72</b>	1.97	2.70 $\pm$ 2.53
ALAX (4)	<b>1.93</b>	<b>2.45 <math>\pm</math> 2.95</b>	2.71	4.07 $\pm$ 3.80	2.44	4.00 $\pm$ 4.10
SAXA (3)	<b>2.98</b>	<b>4.91 <math>\pm</math> 5.94</b>	3.64	5.26 $\pm$ 5.68	3.62	5.21 $\pm$ <b>5.43</b>
SAXB (4)	<b>2.66</b>	<b>4.05 <math>\pm</math> 5.32</b>	5.59	7.21 $\pm$ 6.98	6.08	7.67 $\pm$ <b>1.17</b>
SAXM (6)	<b>2.85</b>	<b>4.30 <math>\pm</math> 8.61</b>	3.96	5.10 $\pm$ <b>6.01</b>	4.38	5.44 $\pm$ 6.07
SAXMV (6)	<b>2.03</b>	<b>3.97 <math>\pm</math> 8.44</b>	4.39	6.12 $\pm$ 6.93	4.73	6.33 $\pm$ <b>6.83</b>

The best value in each row is in **bold**.

**Table 13. Baseline Comparison of Planar Orientation**

View (#landmarks)	$E_{angle}$ (in degree)					
	SCN (My)		FCNN		DSN	
	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD	Median	Mean $\pm$ SD
A2C (3)	11.64	13.60 $\pm$ 10.96	9.56	<b>11.51 <math>\pm</math> 10.56</b>	<b>8.01</b>	12.40 $\pm$ 11.36
A4C (6)	5.03	6.38 $\pm$ <b>4.69</b>	<b>2.95</b>	<b>6.13 <math>\pm</math> 7.00</b>	4.42	6.19 $\pm$ 6.27
ALAX (4)	<b>8.53</b>	<b>11.45 <math>\pm</math> 12.19</b>	10.71	12.78 $\pm$ <b>11.75</b>	10.39	13.25 $\pm$ 13.16
SAXA (3)	<b>5.43</b>	<b>7.29 <math>\pm</math> 6.67</b>	6.44	7.50 $\pm$ <b>4.70</b>	7.89	9.13 $\pm$ 6.96
SAXB (4)	11.76	14.37 $\pm$ <b>8.97</b>	<b>11.01</b>	<b>13.51 <math>\pm</math> 11.49</b>	14.47	16.50 $\pm$ 11.08
SAXM (6)	<b>5.99</b>	<b>7.60 <math>\pm</math> 6.61</b>	6.97	7.84 $\pm$ <b>4.86</b>	7.90	9.66 $\pm$ 6.66
SAXMV (6)	<b>6.13</b>	<b>7.46 <math>\pm</math> 4.68</b>	7.38	7.92 $\pm$ <b>3.92</b>	8.57	10.84 $\pm$ 9.89

The best value in each row is in **bold**.

Since FCNN and DSN only predict one landmark's location in one forward pass, the localization evaluation was calculated on a landmark-by-landmark basis but not on

my view-by-view basis. Although it is possible to coalesce the landmark location data on each cross-section view, the group members did not perform such statistical analysis, so the landmark localization comparison is omitted here. And we shall focus on the cross-section recovery comparison.  $E_{realshift}$  is shown in Table 12, and  $E_{angle}$  is shown in Table 13.

We can observe that SCN excels at predicting the plane position, which is based on the centroid of landmarks on the cross-section. It obtains the best median and mean  $E_{realshift}$  on all cross-section planes except for A4C. This advantage is explainable because SCN predicts all landmarks' locations on one cross-section with one model, enabling it to learn the relationships between different landmarks. Meanwhile, FCNN and DSN treat each landmark individually, so they may not predict the centroid of all landmarks on one cross-section so accurately. For  $E_{angle}$ , SCN achieves the best results on four cross-sections: ALAX, SAXA, SAXM, and SAXMV, which is still overall advantageous over the other two baselines. In summary, SCN performs better than FCNN and DSN in cross-section recovery.

## 8. Visualization

To visually examine the quality of the recovered cross-sections, I rendered the 2D cross-section images on both the predicted plane  $\Pi = (\mathbf{c}, \mathbf{n})$  and the ground truth plane  $\Pi^* = (\mathbf{c}^*, \mathbf{n}^*)$  for comparison. To render an image, I needed to calculate the coordinates (relative to the 3D echo data) of the vertices on a 2D mesh grid which lies on the plane  $\Pi$  (or  $\Pi^*$  in a similar way).

The first step of this approach is to work out two direction vectors  $(\mathbf{dx}, \mathbf{dy})$ ,  $\mathbf{dx} \in \mathbb{R}^3$ ,  $\mathbf{dy} \in \mathbb{R}^3$  of the plane  $\Pi = (\mathbf{c}, \mathbf{n})$ , where  $\|\mathbf{dx}\| = \|\mathbf{dy}\| = 1$  and  $\mathbf{dx}, \mathbf{dy}, \mathbf{n}$  are mutually perpendicular, as follows:

$$\text{unnormed } \mathbf{dx} = \mathbf{u} - \frac{\mathbf{u} \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n}$$

$$\mathbf{dx} = \frac{\text{unnormed } \mathbf{dx}}{\|\text{unnormed } \mathbf{dx}\|}$$

$$\text{unnormed } \mathbf{dy} = \mathbf{n} \times \mathbf{dx}$$

$$\mathbf{dy} = \frac{\text{unnormed } \mathbf{dy}}{\|\text{unnormed } \mathbf{dy}\|}$$

where  $\mathbf{u}$  is a unit vector used to control the directions of  $\mathbf{dx}$  and  $\mathbf{dy}$ , which is also the orientation of the image. In practice, I simply adopted  $\mathbf{u} = (1, 0, 0)^\top$ . Additionally, in case  $\|\text{unnormed } \mathbf{dx}\|$  is very close to 0, any other unit vector can be chosen as  $\mathbf{u}$ .

Secondly, to decide the field of view and the resolution of the image, I introduced two sets of parameters  $(Vx, Vy) \in \mathbb{R}_+^2$  and  $(Rx, Ry) \in \mathbb{Z}_+^2$ , where  $Vx$  and  $Vy$  determine the size of the viewport (*i.e.*, the size of the mesh grid) along the width and height, and  $Rx$  and  $Ry$  determine the number of pixels (*i.e.*, the density of the mesh grid) along the width and height. In practice, I used  $(Vx, Vy) = (128, 128)$  and  $(Rx, Ry) = (512, 512)$ . Then, I obtained the origin  $\mathbf{o}$  of the mesh grid as follows:

$$\mathbf{o} = \mathbf{c} - \frac{Vx}{2} \mathbf{dx} - \frac{Vy}{2} \mathbf{dy}$$

Hence, the set vertices on the mesh grid are

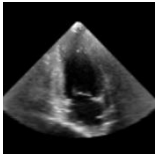
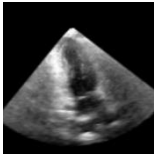
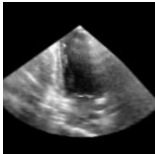
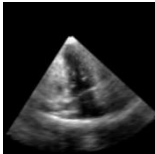
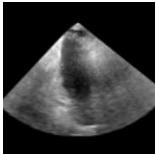
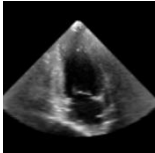
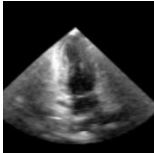
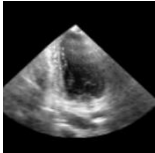
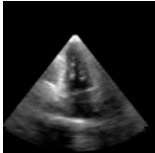
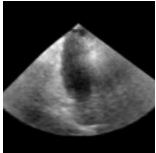
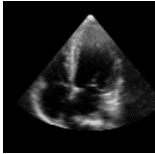
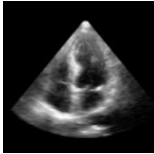
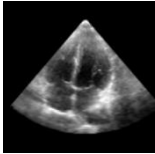
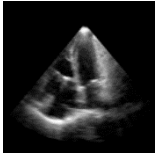
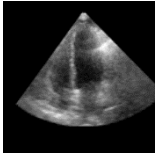
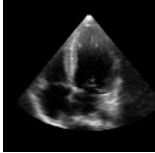
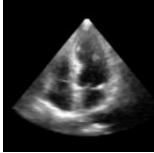
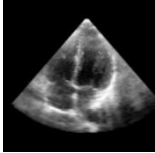
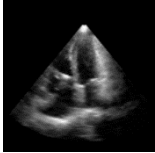
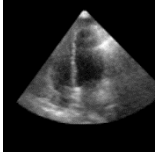
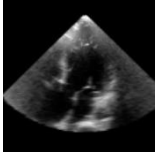
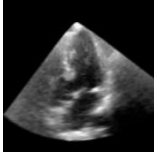
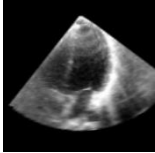
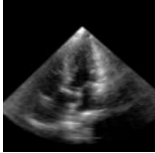
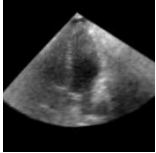
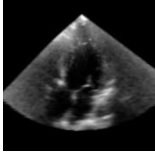
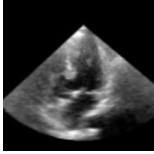
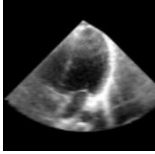
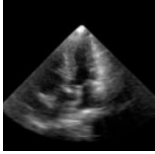
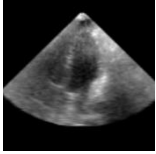
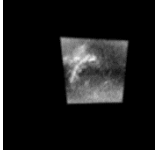
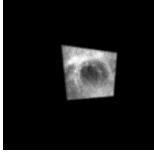
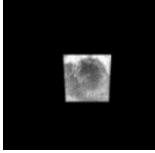
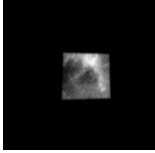
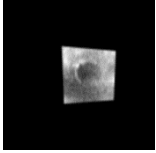
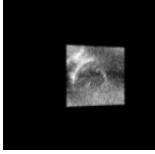
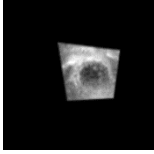
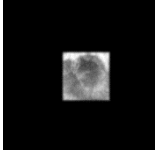
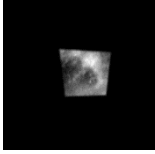
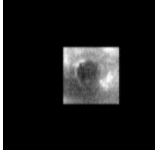
$$\left\{ \mathbf{o} + i \frac{Vx}{Rx} \mathbf{dx} + j \frac{Vy}{Ry} \mathbf{dy} \right\}$$

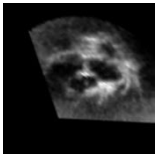
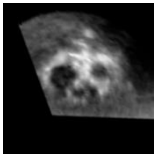
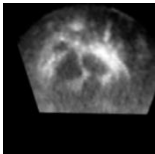
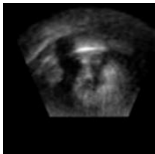
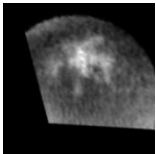
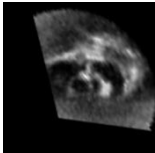
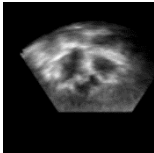
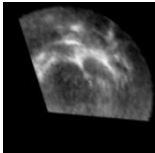
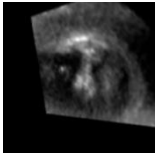
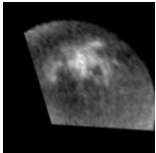
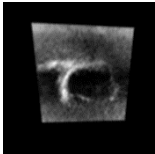
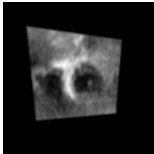
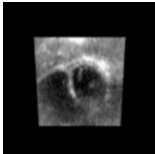
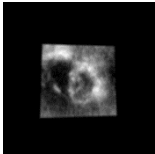
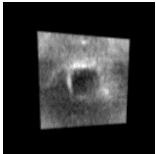
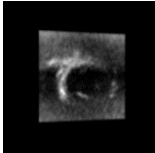
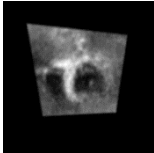
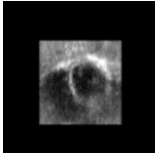
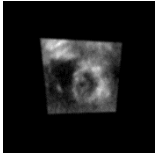
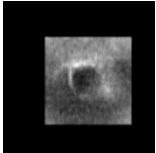
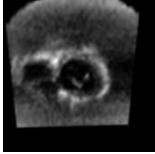
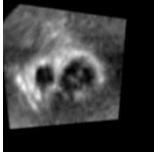
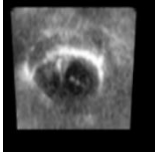
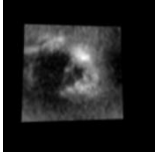
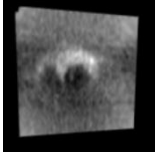
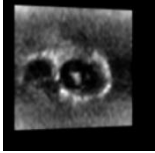
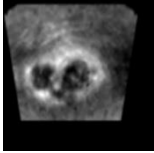
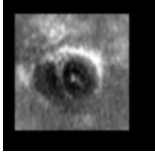
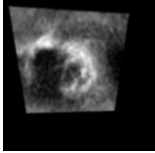
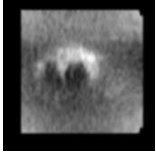
for  $i \in \{1, 2, \dots, Rx\}$  and  $j \in \{1, 2, \dots, Ry\}$ . At last, the intensity values of the vertices



on the mesh grid are calculated from the 3D echo data using linear interpolation, and vertices outside the space of the 3D data are assigned 0 values. Uniformly randomly selected examples of the rendered images are shown in Table 14.

**Table 14. Cross-section Visualization Examples**

	PWHOR2120462 98Q_12Oct2021_ CX0479AK_3DQ	PWHOR2120067 04W_6Oct2021_I M_0041_3DQ	PWHOR2121206 67N_22Oct2021_ D1LHKKHC_3D Q	PWHOR2106219 59U_31Mar2021_ 485M2C5A_3DQ	PWHOR1907802 42U_27Oct2021_ D3W6GH9O_3D Q
A2C (P)					
A2C (T)					
A4C (P)					
A4C (T)					
ALAX (P)					
ALAX (T)					
SAXA (P)					
SAXA (T)					

SAXB (P)					
SAXB (T)					
SAXM (P)					
SAXM (T)					
SAXMV (P)					
SAXMV (T)					

(P): predicted cross-sections, (T): ground truth cross-sections. The column header is the identifier of the echo data.

## 9. Discussion

### 9.1. Limitations

#### 9.1.1. Direct Prediction of Cross-section Parameters

This project uses a heatmap-based neural network to predict the locations of landmarks on a cross-section and then utilizes geometric methods to reconstruct cross-section parameters from the landmark locations. However, this method may suffer from some drawbacks like the collinearity problem that we have seen on the SAXA view and the problem that the number of landmarks on a cross-section affects the quality of the recovered cross-section even though the localization errors of these landmarks are similar across different cross-sections. Therefore, it is possible to discover other neural network architectures that directly predicted the cross-section parameters from input echo data, which eliminates shortcomings of traditional geometric calculations.

#### 9.1.2. Adaptive Learnable Ground Truth Heatmap

In later research, Payer *et al.* [12] improved the SpatialConfiguration-Net by making the standard deviation  $\sigma$  in each  $\mathbb{H}_i^*$  a learnable parameter so that the network can decide by itself how accurately it can learn about each landmark's location. The learned value of  $\sigma$  tells us how confident the network is about the landmark's location; the smaller, the more confident. This adaptive  $\sigma$  is better than the manually selected fixed  $\sigma = 2.0$  used in this project in that the network can better fit the ground truth heatmaps and make a more accurate prediction by making  $\sigma$  smaller.

Furthermore, the learnable  $\sigma$  is also a more informative indicator of the confidence level of each landmark for recovering the cross-section plane. If  $\sigma$  is too large for a landmark, which means that the network is uncertain about its location, we can exclude the landmark from the cross-section recovery. This approach can be adapted to the Top Confidence Fit in Section 6.1 by using  $\sigma$  instead of the intensity value as the confidence indicator, which may produce a better method than the Least Square Fit.

### 9.1.3. Sub-voxel Accuracy

The neural network in this project adopts a spatial resolution of  $128 \times 128 \times 128$ , which is coarser than the original echo data. By extracting the maximum points from the predicted heatmaps, the predicted landmarks are confined to the vertices of a discrete grid, limiting the attainable accuracy of the network. Therefore, it is better to achieve sub-voxel accuracy by performing a quadratic approximation using voxels adjacent to the maximum voxel. This approach is possible as the center of the Gaussian function on the ground truth heatmaps also has a sub-voxel accuracy, so the network may be able to learn more accurate locations from the ground truth heatmaps.

### 9.1.4. Training Process

The project adopts a fixed 8 : 2 ratio to split the training and the validation dataset, and hyperparameter tuning was carried out solely on the validation dataset. However, considering the limited data in the dataset, it is better to perform a 5-fold cross validation on the dataset so that the validation results would be more stable and reliable. Besides, it is also better to validate all cross-sections using the same configuration instead of relying solely on the results of the A2C view.

When tuning the hyperparameters in the first stage, I simply chose the set of options through experience and experiments done by Payer *et al.* [12]. However, it would be better to explore more choices of loss functions and optimizers and perform a grid search on the hyperparameters to figure out the best configuration. In addition, I experienced gradient vanishing and gradient explosion when using the Nesterov SGD optimizer. Although the problem disappeared when I changed to the AdamW optimizer, the cause of this problem is still worth investigating.

Furthermore, the data augmentation used in the project consisted of some simple geometric transformations. It is foreseeable that adding some other transformations would further improve the generalizability of the network. For example, Payer *et al.* [12] employed elastic deformations on 2D echo data, which may be extended to 3D for this project.

At last, training a model for one cross-section is time-consuming, taking up to 30 hours. Therefore, it is better to speed up the training of other models by employing transfer learning because the first few layers of different models should share common parameters as they are used for preliminary feature extraction.

### 9.1.5. Cross-section Recovery Metrics

When evaluating the reconstructed cross-sections, I mainly use two metrics  $E_{realshift}$  and  $E_{angle}$ .  $E_{realshift}$  tells the distance between the centroids of the predicted and the ground truth landmarks along the ground truth plane normal, and it can be interpreted as the distance of the two planes only when  $E_{angle}$  is very small. Hence,  $E_{angle}$  seems to be a dominant metric in evaluating the quality of the reconstructed plane. Therefore, combining the two metrics into one for more intuitive evaluation is better.

## 9.2. Future Work

Based on the reconstructed 2D cross-section view, we can engineer a downstream neural network that takes in these 2D cross-sections and automatically diagnoses heart diseases. This would require designing deep CNNs to classify heart diseases and extra 2D echo data labeled with the diagnosis of different heart diseases. Once this downstream neural network is completed, the combined neural network will have a complete workflow that takes in 3D echo data obtained by transducers and automatically suggests possible heart diseases of the patient, which can assist doctors in making diagnoses and make public screening of common structural diseases feasible.

## 10. Conclusion

The project innovatively applied deep learning methods to speed up the acquisition of cross-section views of the heart. It used the SpatialConfiguration-Net framework [12] that combines the local appearance and spatial configuration information into a single network trained in an end-to-end manner. The network's ability to combine local and global features and multiple improvement techniques enabled the model to accurately predict the landmark locations with a median point-to-point error down to 4.76 mm.

The cross-section views were then reconstructed automatically from the predicted landmarks using the least squared distance fit combined with other constraints on the cross-sections. The median of the angle difference between the reconstructed cross-sections and the ground truth cross-sections was down to 5.03°.

This process saves the trouble of manually locating cross-sections to obtain 2D echocardiography and is expected to improve the efficiency of heart disease examination. In the future, this project can be used by other downstream neural networks to automatically diagnose heart diseases, which not only saves time for cardiologists but also makes mass heart disease screening possible.

## References

- [1] K. Mc Namara, H. Alzubaidi, and J. K. Jackson, “Cardiovascular disease as a leading cause of death: how are pharmacists getting involved?,” *Integrated pharmacy research & practice*, vol. 8, p. 1, 2019.
- [2] H. M. Salah et al., “Causes of hospitalization in the USA between 2005 and 2018,” *European Heart Journal Open*, vol. 1, no. 1, p. oeab001, 2021.
- [3] J. S. Gottdiener et al., “American Society of Echocardiography recommendations for use of echocardiography in clinical trials: a report from the american society of echocardiography’s guidelines and standards committee and the task force on echocardiography in clinical trials,” *Journal of the American Society of Echocardiography*, vol. 17, no. 10, pp. 1086–1119, 2004.
- [4] C. Mitchell 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, pp. 1–64, 2019.
- [5] A. Parthiban et al., “Improving Wait Time for Patients in a Pediatric Echocardiography Laboratory-a Quality Improvement Project,” *Pediatric Quality & Safety*, vol. 3, no. 3, 2018.
- [6] B. Munt et al., “Treating the right patient at the right time: Access to echocardiography in Canada,” *Canadian Journal of Cardiology*, vol. 22, no. 12, pp. 1029–1033, 2006.
- [7] M. J. Monaghan, “Role of real time 3D echocardiography in evaluating the left ventricle,” *Heart*, vol. 92, no. 1, pp. 131–136, 2006.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [9] C. Payer, D. Štern, H. Bischof, and M. Urschler, “Regressing heatmaps for multiple landmark localization using CNNs,” in *International conference on medical image computing and computer-assisted intervention*, 2016, pp. 230–238.

- [10] J. M. Noothout, B. D. de Vos, J. M. Wolterink, T. Leiner, and I. Išgum, “CNN-based landmark detection in cardiac CTA scans,” *arXiv preprint arXiv:1804.04963*, 2018.
- [11] J. Zhang, M. Liu, and D. Shen, “Detecting anatomical landmarks from limited medical imaging data using two-stage task-oriented deep neural networks,” *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 4753–4764, 2017.
- [12] C. Payer, D. Štern, H. Bischof, and M. Urschler, “Integrating spatial configuration into heatmap regression based CNNs for landmark localization,” *Medical image analysis*, vol. 54, pp. 207–219, 2019.
- [13] X. Wang, L. Bo, and L. Fuxin, “Adaptive wing loss for robust face alignment via heatmap regression,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6971–6981.
- [14] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, “Wing loss for robust facial landmark localisation with convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2235–2245.
- [15] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.