CONGENITAL HEART VESSEL SEGMENTATION AND CLASSIFICATION USING DEEP LEARNING TECHNIQUES

A PROJECT REPORT
submitted to
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY, KERALA

by

BENSEN MATHEW (MUT20EC022)

in partial fulfilment of the requirements for the award of the Degree of
BACHELOR OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
MUTHOOT INSTITUTE OF TECHNOLOGY AND SCIENCE
VARIKOLI P.O, PUTHENCRUZ - 682308
MAY 2024

ACKNOWLEDGMENTS

First of all to the Great Almighty, the author of knowledge and wisdom for his countless love. With great respect, I express my sincere thanks to our Head of The Department **Dr. Abhilash Antony** for all the proper guidance and encouragement. I extent my gratitude to the project coordinators **Dr. Prathibha Sudhakaran**, **Dr. Arunkanth A Jose** and **Dr. Shajimon K John** for their timely advice, meticulous scrutiny, scholarly advice and scientific approach that helped to a very great extent throughout the project. I would like to sincerely thank my guides **Dr. Dhanya S** and **Ms. Anjali S.V**, for their support and valuable guidance.

I express my heartfelt veneration to all who had been helpful and inspiring throughout this endeavour.

Bensen Mathew

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

BENSEN MATHEW
(MUT20EC022)

Place : Varikoli

Date : 09/05/2024

MUTHOOT INSTITUTE OF TECHNOLOGY & SCIENCE

Varikoli P.O, Puthencruz-682308

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project final report entitled "**CONGENITAL HEART VESSEL SEGMENTATION AND CLASSIFICATION USING DEEP LEARNING TECHNIQUES**" submitted by **Bensen Mathew(MUT20EC022)** of Semester VIII is a bonafide account of the work done by her under our supervision.

Dr. Dhanya S.
(Project Guide)
Asso. Professor
Dept.of ECE
MITS
Varikoli

Ms. Anjali S.V
(Project Guide)
Asst. Professor
Dept.of ECE
MITS
Varikoli

Dr. Arunkanth A Jose
(Project Coordinator)
Asst. Professor
Dept.of ECE
MITS
Varikoli

Dr. Prathibha Sudhakaran
(Project Coordinator)
Asst. Professor
Dept.of ECE
MITS
Varikoli

Dr. Shajimon K John
(Project Coordinator)
Professor and Dean Academics
Dept.of ECE
MITS
Varikoli

Dr. Abhilash Antony
(Head of the Department)
Asso. Professor
Dept.of ECE
MITS
Varikoli

External Examiner

Contents

List of Figures

Abstract

Our project introduces an innovative strategy for diagnosing congenital heart diseases (CHDs) by leveraging advanced deep learning techniques. Specifically, we utilize a 3D U-Net architecture, which is a specialized neural network structure renowned for its efficacy in volumetric image segmentation tasks. This architecture enables precise delineation of heart vessels, a critical step in diagnosing and planning treatments for CHDs, which afflict 1 in every 110 births. Furthermore, we employ a convolutional neural network (CNN), a class of deep learning models adept at learning hierarchical features from image data, to accurately identify and classify sixteen distinct types of congenital heart defects (CHD). This integration highlights the CNN's effectiveness in medical diagnosis, promising improved patient care. Through automating both segmentation and classification processes, our project aims to enhance diagnostic precision, streamline treatment planning procedures, and ultimately elevate patient outcomes in CHDs.

Chapter 1

Introduction

Our project focuses on classifying different forms of congenital heart disease (CHD) and segmenting congenital heart vessels utilizing the cutting-edge 3D U-Net design. Congenital heart defects (CHDs) are intricate structural defects that impact cardiac function and may involve complex changes in heart arteries. Precise diagnosis, treatment planning, and patient monitoring for congestive heart failure depend heavily on accurate segmentation of these arteries. Our goal is to improve and automate the process of separating heart vessels from MRI or CT scans, giving medical practitioners precise and comprehensive anatomical insights. We will achieve this by utilizing deep learning and medical image analysis.

Moreover, our work goes beyond segmentation to include the critical work of categorizing various forms of CHD. By utilizing extensive datasets and machine learning methods, our goal is to create a reliable classification system that can classify CHD situations according to segmented vessel patterns. This classification capacity is extremely valuable for clinical practice because it allows physicians to quickly identify particular forms of CHD, which allows for more individualized treatment plans and better patient outcomes. Our project intends to transform pediatric cardiology's diagnostic and treatment landscape by integrating state-of-the-art technologies, such as machine learning for classification and deep learning for segmentation. Through the provision of automated, precise, and effective methods for heart vessel segmentation and CHD classification, our goal is to make a substantial contribution to improved clinical decision-making, patient care, and congenital heart disease research developments.

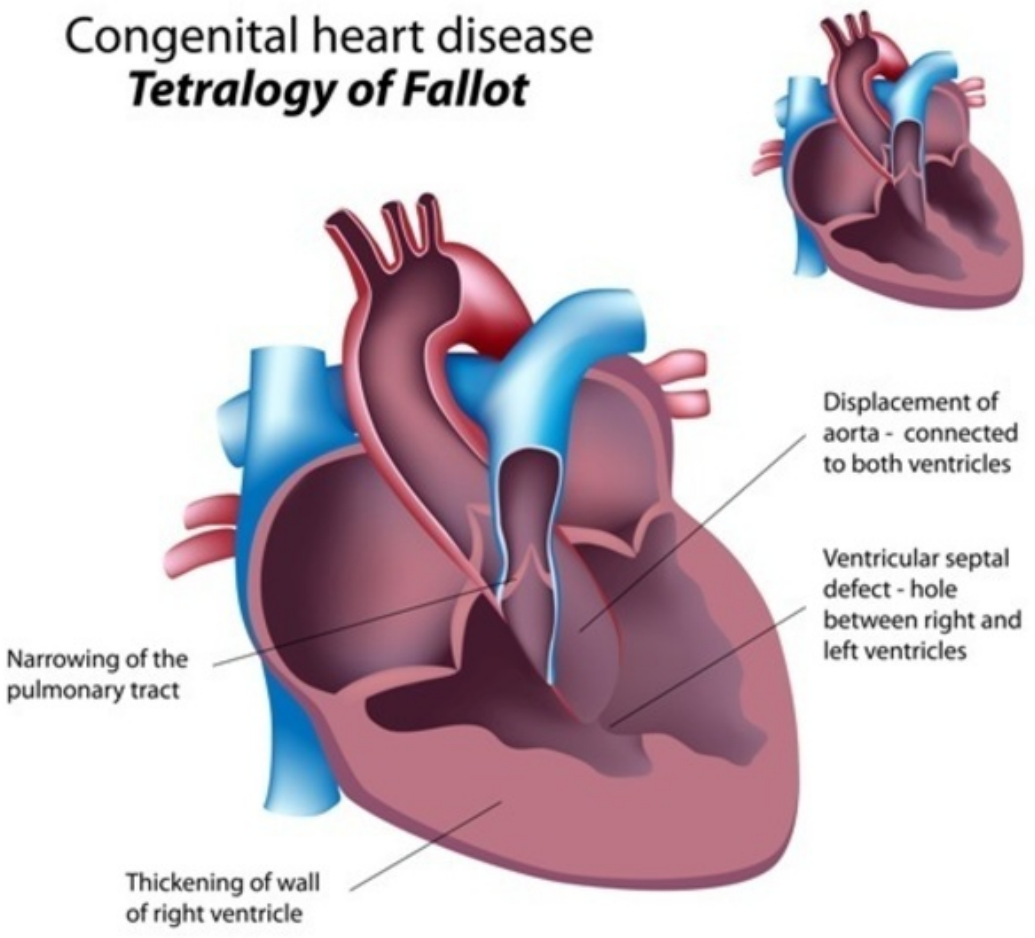

Figure 1.1: CHD- Tetralogy of Fallot [21]

1.1 Scope and Motivation

This project uses cutting edge deep learning and machine learning approaches to automate cardiac imaging analysis for congenital heart disease (CHD). For an immediate and precise diagnosis in order to improve patient outcomes, automation is essential. In order to improve patient care and healthcare technology, we concentrate on segmenting cardiac vessels and classifying different forms of congenital heart disease from 3D CT scans.

Our project is motivated by multiple important factors:

1. Clinical Impact: Improving patient outcomes and lowering mortality rates by addressing the urgent need for an accurate and prompt diagnosis of congestive heart failure (CHD).

2. Technological Progress: Advancing automated medical image analysis by utilizing cutting-edge deep learning and machine learning algorithms.

3. Efficiency and Precision: By automating CHD diagnosis processes, healthcare providers can save significant time while simultaneously improving the accuracy and consistency of their diagnoses.

4. Research Contribution: Encouraging additional study and development in the fields of medical imaging and CHD analysis by offering insightful approaches and methodologies.

5. Patient-Centric Approach: In the end, we want to improve the standard of care for patients with congestive heart failure (CHD) by offering tailored, efficient treatment regimens that are founded on precise diagnostic evaluations.

Our project's scope includes the following key areas:

1. Image Preprocessing: Improving 3D CT image quality by removing artifacts, enhancing contrast, and reducing noise.

2. Segmentation: Accurate and effective heart vessel segmentation is achieved by using cutting-edge deep learning architectures like U-Net.

3. Classification: Based on segmented vessel patterns, machine learning methods are used to classify different types of CHD.

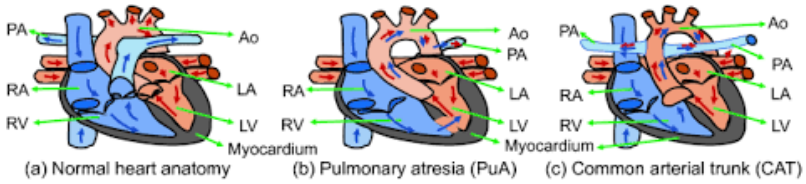

Figure 1.2: Examples of large structure variations in CHD. In normal heart anatomy (a), PA is connected to RV. However, in pulmonary atresia (b), PA is rather small and connected to descending Ao. In common arterial trunk (c), Ao is connected to both RV and LV, and PA is connected to Ao. [5]

1.2 Objectives

1. Data Preprocessing: To increase segmentation accuracy, create a pipeline for preprocessing 3D CT images that includes normalization, noise removal, and enhancement.

2. 3D U-Net segmentation: Develop and test a 3D U-Net model to identify anatomical structures from CT scans, as measured by a high Dice score or other pertinent metrics.

3. CHD Classification using CNN: Using a dataset of 16 different types of CHD, train a CNN model to correctly identify the type of CHD found in a particular CT scan, obtaining a high classification accuracy.

1.3 Applications

Medical imaging analysis for congenital heart disease using deep learning models presents a groundbreaking approach with diverse applications that revolutionize cardiovascular healthcare. Key applications include :

Enhancement of Clinical Diagnosis: With the use of medical imaging data, our initiative can help physicians correctly diagnose and categorize various forms of congenital heart disease (CHD). Better patient outcomes and more targeted treatment regimens may result from this.

Treatment Planning: Our project can assist in the development of individualized treatment plans for CHD patients, taking into account the unique characteristics and severity of their illness, by offering automated segmentation and classification of cardiac structures and defects.

Real-time Decision Support: Our models may be integrated into medical imaging software suites to allow for real-time analysis of 3D CT scans. This will allow physicians to provide quick feedback to patients during consultations or surgical operations.

Research Development: By using our project to evaluate enormous databases of cardiac imaging, scientists and physicians can better understand congenital heart disease (CHD) and create novel treatments by spotting trends, patterns, and predicting signs associated with the condition.

Education and Training: By using our initiative to teach medical students, residents, and healthcare professionals on CHD diagnosis, imaging interpretation, and treatment planning, educational institutions and training programs can improve the abilities and knowledge of cardiovascular healthcare providers.

Clinical trials and validation studies: The capabilities of our project can be applied to evaluate the effectiveness and dependability of deep learning-based methods in medical imaging analysis for congenital heart disease (CHD). This could result in regulatory approvals and broad implementation in clinical practice.

Chapter 2

Literature Survey Report

2.1 Rapid whole-heart CMR with single volume super-resolution[1]

Recent developments in cardiac magnetic resonance (CMR) imaging technology are examined in the study "Rapid whole-heart CMR with single volume super-resolution" by J. A. Steeden et al. The authors introduce a novel super-resolution method that improves whole-heart CMR imaging. This technique makes it possible to acquire photos more quickly without sacrificing image quality. The method's implementation and possible advantages for clinical applications are covered in the paper's technical section. The study shows that the method may produce high-resolution imaging in a single volume, which may have important ramifications for accurately and quickly diagnosing and evaluating cardiac disorders.

2.2 Graph matching and deep neural networks based whole heart and great vessel segmentation in congenital heart disease[2]

An innovative technique for precisely segmenting the whole heart and great vessels in patients with congenital heart disease is presented in the publication "Graph matching and deep neural networks based whole heart and great vessel segmentation in congenital heart disease" by Z. Yao et al. This method achieves accurate segmentation results by combining deep neural networks with graph matching. Deep neural networks enable reliable and effective image processing, while graph matching is used to assist build correspondences between various anatomical components. In order to diagnose and plan for congenital heart disease more effectively, superior segmentation performance is made possible by this combination. It is possible that this study will lead to the development of more sophisticated imaging methods for cardiac care because it shows how effective the suggested method is.

2.3 Automated heart segmentation using u-net in pediatric cardiac CT[3]

A deep learning strategy to accurately segment the heart in pediatric cardiac computed tomography (CT) images is shown in the publication "Automated heart segmentation using U-Net in pediatric cardiac CT" by A. Yoshida et al. The procedure is automated by the authors by using the U-Net architecture, a reputable convolutional neural network for picture segmentation. Because the anatomical structures of different age groups vary, manual segmentation can be difficult and time-consuming in pediatric cases. our is where our technique comes in handy. In order to demonstrate the U-Net model's potential to increase diagnostic accuracy and efficiency, the paper talks about how to optimize and implement it for pediatric cardiac CT scans. Strong results in automated heart segmentation are demonstrated by the technology, which may improve cardiac imaging and diagnostics in children.

2.4 Learned iterative segmentation of highly variable anatomy from limited data: Applications to whole heart segmentation for congenital heart disease[4]

A unique method for segmenting the complete heart using limited data in situations of congenital heart illness is presented in the publication "Learned iterative segmentation of highly variable anatomy from limited data: Applications to whole heart segmentation for congenital heart disease" by D. F. Pace et al. In order to accommodate the significant variation in cardiac anatomy among patients with congenital heart problems, the authors suggest an iterative learning approach. Through the application of deep learning techniques, the system gradually improves the segmentation process, making it more capable of handling a variety of anatomical structures. The study shows that even with a small amount of training data, the method has the potential to increase heart segmentation accuracy and efficiency. Better diagnosis and treatment planning tools for congenital heart disease could result from this development, which would be advantageous for patients as well as professionals.

2.5 Whole heart and great vessel segmentation in congenital heart disease using deep neural networks and graph matching [5]

A novel method for segmenting the whole heart and great vessels in patients with congenital heart disease is covered in the paper "Whole heart and great vessel segmentation in congenital heart disease using deep neural networks and graph matching" by X. Xu et al., which was presented at the International Conference on Medical Image Computing and Computer-Assisted Intervention. The authors increase the efficiency and accuracy of segmentation by combining deep neural networks with graph matching approaches. While graph matching helps create correspondences between various anatomical features, neural networks offer robust image processing that enables the technology to handle the heterogeneity in cardiac anatomy typically seen in congenital heart disease. The success of this hybrid technique is reported in the research, highlighting its

potential for use in clinical settings for tasks like diagnosis and treatment planning. With regard to cardiac imaging for congenital heart disease, this work marks a major advancement.

2.6 Simple and complex congenital heart disease in infants and children[6]

The use of cardiovascular magnetic resonance (CMR) in the diagnosis and assessment of congenital heart disease (CHD) in infants and children is the subject of A. J. Powell and T. Geva's chapter "Simple and complex congenital heart disease in infants and children" in the book Cardiovascular Magnetic Resonance, edited by R. Y. Kwong et al. The authors describe how CMR provides thorough information on cardiac architecture, function, and blood flow and provides a non-invasive, all-encompassing approach for diagnosing both basic and complicated CHD. They draw attention to the benefits of CMR in pediatric instances, such as its non-ionizing radiation source and capacity to produce crisp, high-resolution images. The chapter discusses the therapeutic uses of CMR in different forms of CHD, highlighting the role that CMR plays in directing management and therapy plans. Because CMR is safe and accurate in assessing congenital cardiac abnormalities, the authors generally support its increased usage in pediatric cardiology.

2.7 U-net:Convolutional networks for biomedical image segmentation [7]

A deep convolutional neural network specifically created for biomedical image segmentation, the U-Net architecture is introduced in the paper "U-Net: Convolutional Networks for Biomedical Image Segmentation" by O. Ronneberger, P. Fischer, and T. Brox. The paper was presented at the International Conference on Medical Image Computing and Computer-Assisted Intervention. The authors outline the novel architecture of the network, which consists of an encoder-decoder architecture with skip connections that let the model fuse the up-sampled output from later layers with the spatial information from previous levels. With little training data, this method allows U-Net to segment medical images with excellent accuracy. The model's efficiency across a range of biological applications is reported in the paper, showcasing its speed and efficiency in producing accurate segmentations. Since then, U-Net has expanded beyond its initial function to become a widely used fundamental model in medical image processing.

2.8 ImageCHD: A 3D Computed Tomography Image Dataset for Classi cation of Congenital Heart Disease[8]

The ImageCHD dataset, a carefully selected set of 3D CT scans with an emphasis on different forms of congenital heart disease (CHD), is presented in the paper "ImageCHD: A 3D Computed Tomography Image Dataset for Classification of Congenital Heart Disease". It highlights the diversity and therapeutic importance of the dataset by going into detail about its acquisition, preprocessing, and annotation procedures. The study reports performance metrics and presents a convolutional neural network (CNN) based approach

for CHD classification. It makes the dataset a useful tool for CHD diagnosis and research breakthroughs by discussing its implications for automated diagnostic tools, cardiac imaging research, and future enhancement opportunities.

2.9 The importance of segmental situs in the diagnosis of congenital heart disease[9]

Understanding segmental situs is critical to detecting congenital heart disease, as discussed by R. Van Praagh in the study "The Importance of Segmental Situs in the Diagnosis of Congenital Heart Disease" published in Seminars in Roentgenology. The spatial orientation and arrangement of the visceral and cardiac components is referred to as segmental situs. The position and orientation of the heart, atria, ventricles, and major vessels, among other segmental anatomy, must all be evaluated methodically, according to the paper. Accurate diagnosis and treatment of congenital cardiac abnormalities depend on an understanding of these interactions. Van Praagh offers a thorough explanation of the many kinds of segmental abnormalities and how they affect diagnosis and therapy. This foundational work, which has influenced methods in pediatric cardiology and radiology, emphasizes the significance of segmental analysis in the assessment of congenital heart disease.

2.10 Fully-automated deep-learning segmentation of pediatric cardiovascular magnetic resonance of patients with complex congenital heart diseases[10]

The use of deep learning for the automated segmentation of pediatric cardiovascular magnetic resonance (CMR) images in patients with complex congenital heart diseases is investigated in the paper "Fully-automated deep-learning segmentation of pediatric cardiovascular magnetic resonance of patients with complex congenital heart diseases" by S. Kanimi-Bidhendi et al. The authors describe an accurate and efficient deep-learning-based technique for segmenting cardiac tissues, including the heart and major vessels. The difficulty of managing juvenile patients' complex and variable anatomical features is addressed by this method. The study assesses the automated segmentation technique's efficacy and shows how it might improve pediatric cardiology's diagnostic and treatment planning procedures. According to the research, deep learning has the potential to significantly advance medical imaging technologies, especially for children with complex congenital heart diseases.

2.11 Deep learning for cardiac image segmentation: A review frontiers in cardiovascular medicine[11]

Chen, Chen, et al. provide a thorough analysis of deep learning methods specifically used for cardiac image segmentation in their 2020 review that was published in Frontiers in Cardiovascular Medicine. They examined many deep learning techniques for separating heart structures from medical imaging data, highlighting the

benefits and drawbacks of each approach. The review emphasized the benefits of deep learning, demonstrating how it may effectively and accurately segment cardiac structures to support accurate diagnosis and treatment planning. The authors also covered how deep learning models may be used to automate laborious manual segmentation chores and how flexible they are to different imaging modalities. However, the constraints included difficulties managing modest datasets, requiring significant computer power, and apprehensions regarding the interpretability and generalizability of the model in various patient populations or imaging scenarios.

Chapter 3

Problem Statement

Congenital heart disease (CHD) refers to a group of structural heart defects present at birth. These defects can affect the walls, valves, arteries, and veins of the heart, disrupting the normal flow of blood through the heart.

Common types of CHD include atrial septal defect, ventricular septal defect, tetralogy of Fallot, and transposition of the great arteries. CHD may manifest with symptoms such as cyanosis, difficulty breathing, poor feeding, and failure to thrive. Diagnosis typically involves a combination of physical examination, imaging studies (such as echocardiography and MRI), and sometimes genetic testing.Identifying the specific type of congenital heart disease (CHD) poses challenges due to the wide range of defects, variability in presentations, complex diagnostic imaging interpretation, the need for multidisciplinary approaches, and the interplay of genetic and environmental factors.

3.1 Statistics and Figuratives

Congenital heart diseases (CHDs) are the leading cause of mortality in the first year of life. These account for around 30 percent of total congenital anomalies. Large majority of these structural abnormalities of heart occur as an isolated anomaly, but around 33 percent have associated anomalies. There are few population-based studies in India about the prevalence of CHD and none in Himachal, hence the purpose of the study.

Table 1

Patient characteristics and pattern of congenital heart disease in the population.

Population screened	1882
Male	909
Female	973
Age group of population studied	
<1 year	28
1–5 years	192
5–18 years	540
18–30 years	349
>30 years	773
Congenital heart disease	12 (6.37/1000)
Male	04 (33.66%)
Female	08 (66.66%)
CHD at low altitude	9.25/1000
CHD at moderate altitude	5/1000
CHD in <18 years of age	12.95/1000
Mean age of CHD patients	19.5 ± 11.07 years
Pattern of CHD	
ASD	5 (41.66%)
VSD	4 (33.66%)
PDA	2 (16.66%)
TOF	1 (8.33%)
Patients already operated	04
VSD	02
ASD	01
PDA	01

Table 3.1: Population-based studies in India about the prevalence of CHD

Chapter 4

System Methodology

4.1 Methodology

The process entails gathering annotated CT scans, segmenting CHD regions in new images using a U-Net that has been trained for CHD segmentation. A CNN is trained on annotated pictures to identify illness kinds in fresh CT images by analysing their attributes.

4.2 Segmentation and Classification:

The process of using U-Net to segment coronary heart disease (CHD) from CT images entails multiple steps:

Image Augmentation:

This code provides a set of functions for performing random data augmentation on images and their corresponding masks, commonly used in deep learning tasks like image segmentation. The functions include random flipping, rotating, and scaling with cropping or padding to ensure consistency in input size. These augmentation techniques help diversify the training data, improving the robustness and generalization capability of deep learning models.

Preprocessing and Cross-Validation Split:

This script preprocesses and organizes a Congenital Heart Disease (CHD) segmentation dataset stored in NIfTI format. It includes functions to load and process the image and label pairs, performing tasks such as normalization, resizing, and saving as PNG images. Additionally, it implements a K-fold cross-validation split, generating separate training and testing sets for model evaluation.

Image and Mask Data Processing Utilities:

These utility functions are designed to preprocess image and mask data for segmentation tasks. get˙ids retrieves a list of file IDs from a directory. They crop and resize images based on a specified scale ensuring proper formatting and normalization. . These functions streamline the data preparation process for segmentation tasks.

UNet with Graph Reasoning (GloRe):

This code implements a U-Net architecture with the integration of graph reasoning (GloRe) blocks for enhanced feature extraction and semantic understanding. The network consists of encoder and decoder

paths with downscaling and upscaling operations, respectively, and utilizes graph reasoning modules within the downsampling path to capture long-range dependencies. The final output predicts segmentation masks for the input images.

Loss Functions for Semantic Segmentation:

These functions are tailored for semantic segmentation tasks, aiming to compute loss values that guide the training of deep learning models. soft˙dice˙loss calculates the Soft Dice Loss, a variation of the Dice coefficient loss, while dice˙loss computes the Sørensen–Dice loss, which measures the similarity between two samples. Finally, shape˙match˙loss introduces a loss term based on shape matching, leveraging OpenCV to compare the similarity between predicted and ground truth masks, enhancing segmentation accuracy.

Training U-Net:

U-Net model is trained for segmenting Congenital Heart Disease (CHD) images using shape matching loss. It supports command-line arguments for specifying training parameters such as epochs, batch size, learning rate, and validation percentage. The training progress is logged, and checkpoints are saved periodically.

3D Medical Image Evaluation:

These functions streamline the evaluation of 3D medical image segmentation models.It computes Dice similarity coefficients for each class in a dataset. It prepares 3D images and labels for evaluation by resizing and normalizing them. It assesses a model's performance across multiple images, generating segmentation masks and computing Dice similarity coefficients.

Prediction:

It evaluates the performance of a segmentation model trained for detecting CHD from medical images. It utilizes a pre-trained UNet˙GloRe model to segment CHD regions in a dataset of 3D medical images, calculating metrics such as Dice coefficient to measure the agreement between predicted and ground truth segmentations. Finally, the script reports the average Dice coefficient across all test images as a measure of the model's overall performance.

Reversing Normalization and Converting Stack of PNGs to NIfTI Format:

It loads a stack of PNG images representing slices of a 3D medical volume, reverses the normalization process by scaling pixel values back to their original range, and then saves the volume as a NIfTI (.nii.gz) file. It ensures proper ordering of slices and assumes an identity affine matrix for simplicity during conversion.

The process of using U-Net to segment coronary heart disease (CHD) from CT images entails multiple steps:

Loading and Resizing NIfTI Images for CHD Dataset Analysis:

It defines a function to load and resize NIfTI images from the specified file path to a target size. This function is crucial for preprocessing the CHD dataset images before further analysis or model training.

Splitting Dataset into Training and Testing Sets:

A function is utilized to split the dataset into training and testing sets. X represents the input data (features), while y represents the corresponding labels. The random seed is fixed to 42. After execution, X˙train and X˙test contain the input features for training and testing, respectively, while y˙train and y˙test contain the corresponding labels.

CNN Architecture:

This code defines a CNN using Keras Sequential API with: Four convolutional layers followed by batch normalization, max-pooling, and dropout, flattening layer to convert 3D feature maps into a 1D vector, three fully connected layers with dropout for regularization, output layer with softmax activation for multi-class classification.

Model Training:

The model is compiled with Adam optimizer, binary cross entropy loss, and accuracy metric. Training is performed for 100 epochs on X˙train and y˙train, with validation on X˙test and y˙test.Training history is stored in history variable.

CHD Classification:

A new CT image is loaded and preprocessed using the previously defined function. The saved model is loaded.Predictions are made on the new image using the loaded model.Thresholding is applied to obtain binary predictions for each label. Finally, the predicted labels are extracted based on the thresholding and displayed.

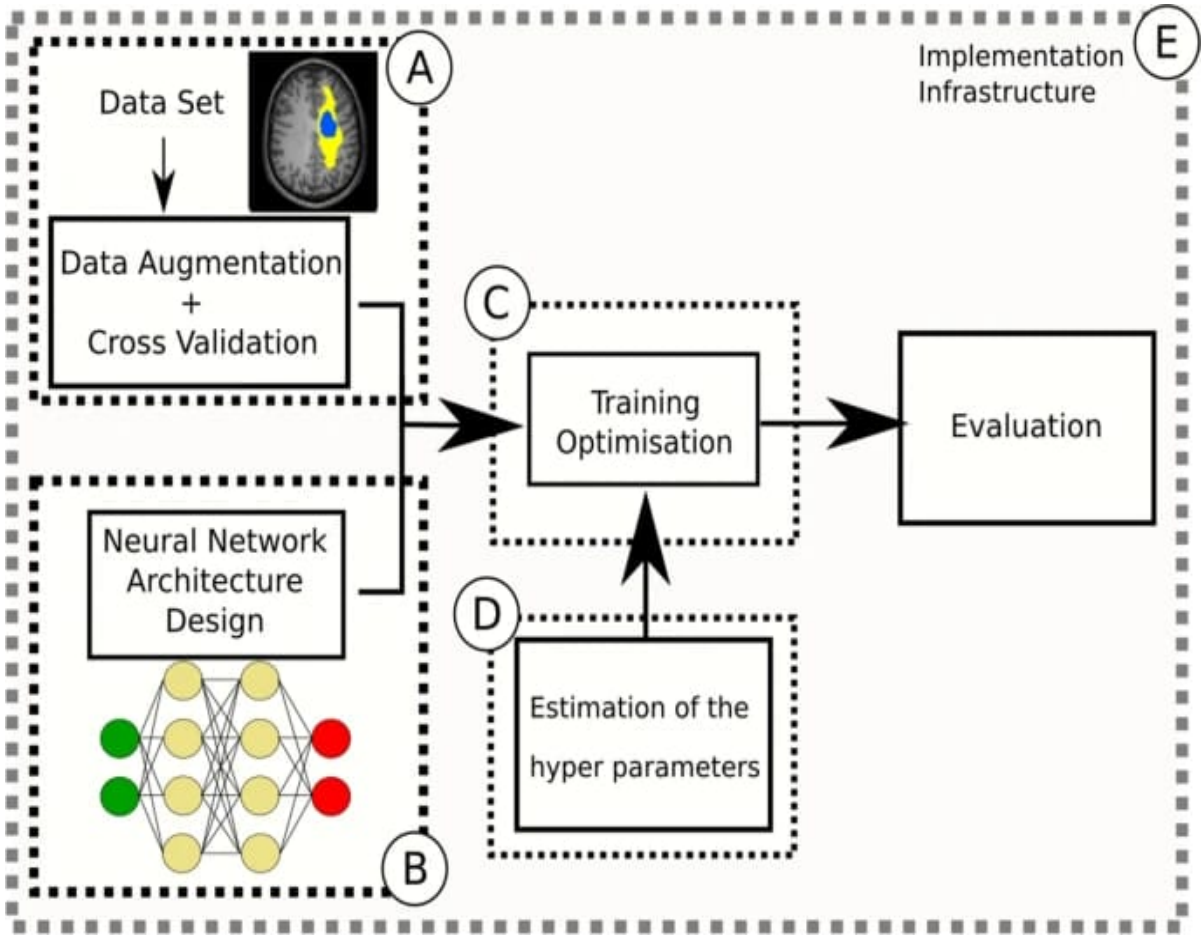

Figure 4.1: Flowchart of Classification [23]

Chapter 5

System Analysis

The segmentation and classification of congenital heart disease can be implemented by using the following algorithms.

1. U-Net
2. Convolutional Neural Network

Each component of the design is described in more details below:

5.1 U-Net

U-Net is a convolutional neural network architecture designed for semantic segmentation tasks, particularly in biomedical image analysis[22]. Its unique U-shaped architecture is made up of a corresponding decoder pathway that creates a segmentation mask with the same spatial dimensions as the input picture and an encoder pathway that extracts features from the input image. Skip links between the encoder and decoder pathways combine fine-grained spatial features with high-level semantic information to enable accurate localization of object boundaries. U-Net gains the ability to forecast pixel-wise probabilities for every class during training by utilizing an appropriate loss function, like dice loss or cross-entropy. U-Net has been widely used in medical image analysis due to its effective use of both local and global context. It has shown impressive performance in a variety of segmentation tasks.

Contracting Path (Encoder):
The encoder, also referred to as the contracting path, is made up of a sequence of convolutional layers that are succeeded by max-pooling layers. These convolutional layers extract characteristics at various spatial scales by applying learnable filters to the input image. In the early levels, these filters pick up basic features; in the deeper layers, they gradually pick up increasingly abstract and sophisticated features.By downsampling the feature maps, max-pooling layers save spatial dimensions without sacrificing significant features.

Bottleneck:
The bottleneck is the point when the input image's most abstract and advanced elements are captured. It is located at the end of the contracting path. It can preserve spatial information because it is made up

of several convolutional layers without the use of max-pooling techniques. The bottleneck layer facilitates accurate image processing by assisting the network in learning a compact and rich representation of the input image.

Expansive Path (Decoder):
The decoder, also known as the expansive path, is made up of convolutional layers after a sequence of upsampling layers.The feature maps' spatial dimensions are increased via upsampling layers, which frequently employ methods like transposed convolution or nearest neighbour interpolation.The decoder's convolutional layers use skip connections to merge features from the contracted path.By concatenating feature maps from the contracted path, these skip connections allow the network to restore spatial information that were lost during downsampling.

Output Layer:
An activation function is placed after a convolutional layer with a 1x1 kernel size. Each class's probability maps are created from the raw output by the activation function, which is usually softmax.Pixel by pixel categorization is made possible by the probability map, where each pixel indicates the possibility of falling into a particular class.

Loss Function:
A loss function, such cross-entropy loss, measures the difference between the ground truth and the anticipated segmentation during training. During optimisation, the loss function directs parameter changes and assesses the model's performance. By using backpropagation to minimise the loss function, the model is able to train to generate precise segmentations.

Training:
Annotated photographs, in which each pixel is labelled with the appropriate class, are used to train the U-Net model. The model uses optimisation techniques like Adam or stochastic gradient descent (SGD) to modify its parameters (weights and biases) during training. During the training phase, the network is fed input photos, the loss is calculated, and the parameters are updated to reduce the loss.

Prediction:
Once trained, the U-Net model can be deployed to predict segmentations for new input images.The input image is passed through the trained model, and the output segmentation map is generated.The model's predictions can be post-processed to refine the segmentation masks and improve the final results.

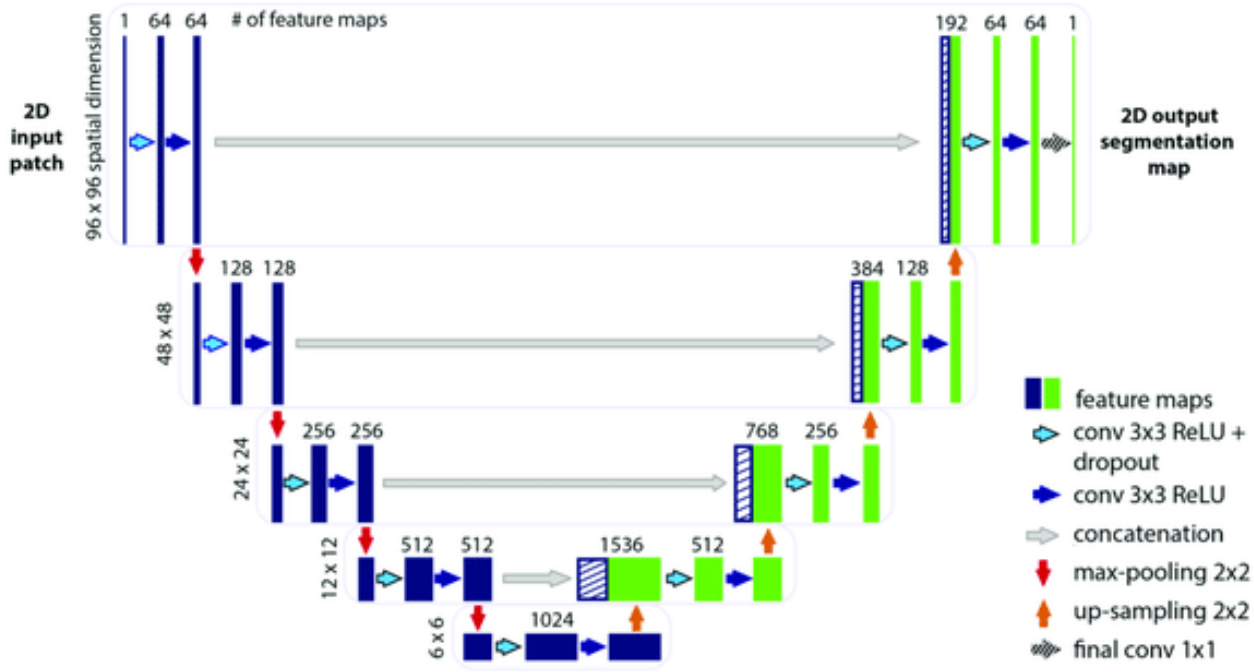

Figure 5.1: U-Net architecture [20]

5.2 CNN

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing structured grid data such as images.They are made up of several layers, such as fully connected, pooling, and convolutional layers. These convolutional layers use convolutions to extract features from input images by applying learnable filters. By downsampling the feature maps, pooling layers save valuable information while decreasing spatial dimensions. All of the image's features are combined via fully connected layers, allowing for high-level abstraction and classification. With the use of labeled data and backpropagation, CNNs are trained to minimize a predetermined loss function by varying the layer weights. CNNs have demonstrated great effectiveness in image classification, object recognition, and picture segmentation due to their hierarchical structure and capacity to automatically learn hierarchical features.

Input Layer:
At the beginning of the CNN, the input layer receives raw data, often in the form of images.The dimensions of each image are represented as a grid of pixel values, with width, height, and channels (such as RGB channels for color images) corresponding to each dimension.

Convolutional Layer:
Learnable filters, often referred to as kernels, move over the input image in this layer to carry out convolutional operations. By doing element-wise multiplication and summing with the local regions of the input image, each filter finds certain patterns or characteristics. Several features of the input data are captured

by the feature maps produced by these processes.

Activation Function:
To add non-linearities to the network, an activation function is applied element-by-element following the convolution procedure. ReLU (Rectified Linear Unit), which replaces negative values with zero, thereby adding non-linearities and allowing the network to model complex relationships.

Pooling layer:
In order to downsample the feature maps without losing any important information, pooling layers come after convolutional layers. While average pooling calculates the average, max pooling chooses the maximum value inside each pooling window. Pooling concentrates on the most important features, which helps lower computational complexity, memory consumption, and overfitting.

Additional Convolutional Layers:
Activation functions and pooling layers are usually stacked after one or more convolutional layers in CNN systems. As input moves through the layers, the network's hierarchical structure enables it to acquire progressively abstract and complicated properties.

Flattening:
The feature maps are reduced to a one-dimensional vector following a number of convolutional and pooling layers. The spatial data is flattened and formatted such that it may be included into fully connected layers.

Fully Connected Layers:
The flattened feature vector is fed into fully connected layers. Every neuron in the layer above it is coupled to every other neuron in a fully connected layer. The network can understand intricate correlations between characteristics thanks to the high-level feature extraction performed by these levels.

Output Layer:
Using the features that have been learned, the output layer generates the final predictions or classifications. The number of classes in the classification task is matched by the number of neurons in the output layer. For instance, multi-class tasks include numerous neurons whereas binary classification tasks have one neuron for each class.

Softmax Activation:
For classification tasks, the softmax activation function is applied to the output layer to convert raw scores into class probabilities. Softmax ensures that the predicted probabilities sum up to 1, making them interpretable as probabilities

Loss Function:
A loss function calculates the variation between the true labels and the expected outputs during training. The task will determine which loss function is best; for multi-class classification, categorical cross-entropy is

frequently utilized, while binary cross-entropy is best for binary classification.

Optimization Algorithm:
To minimize the loss function, an optimization algorithm modifies the network's weights. Weights are adjusted by backpropagation by algorithms such as Adam and Stochastic Gradient Descent (SGD), where weights are changed based on the gradients of the loss function that are computed with respect to the network parameters.

Training:
Labeled training data is used to train the CNN. Batches of input data are fed into the network during training, and iterative updates to the weights are made to reduce loss. The model is trained until it converges.

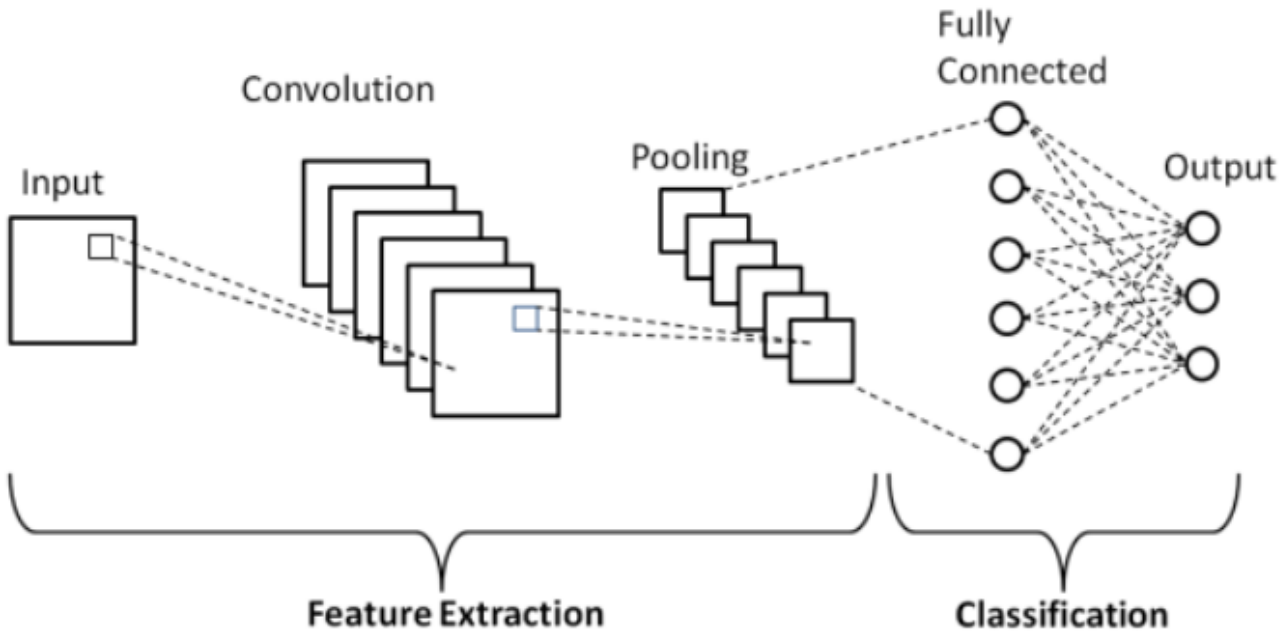

Figure 5.2: CNN Architecture [21]

Chapter 6

Result

In this work, we used a 3D U-Net architecture for congenital heart vessel segmentation and a CNN-based classifier for illness classification. A dataset containing 178 3D CT scans representing different forms of congenital heart disease (CHD) was used to conduct the performance evaluation.

Segmentation Results: The segmentation of congenital cardiac vessels was accomplished by the 3D U-Net model with a Dice score of 80 percent. To illustrate the efficacy of our segmentation method, Figure 6.5 presents reconstructed segmented image.

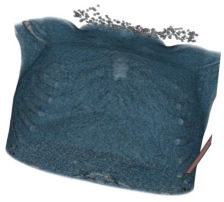

Figure 6.1: Input cardiac image [20]

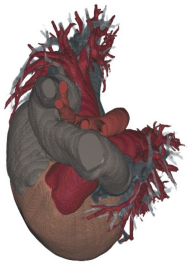

Figure 6.2: Labelled input cardiac image [20]

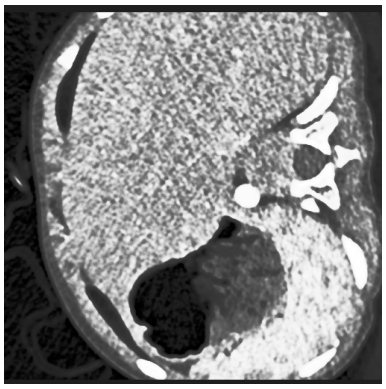

Figure 6.3: 2D sliced cardiac image

Figure 6.4: Predicted 2D sliced image

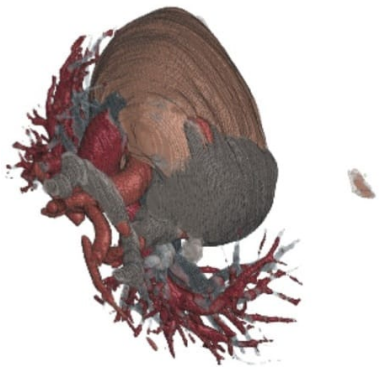

Figure 6.5: Reconstructed 3D cardiac image

Classification Results: With an accuracy of 90 percent for disease categorization, the CNN model showed strong performance. Based on the predictions made by our CNN classifier, Table shows the classification results for eight prevalent forms of CHD and eight less frequent types.

In general, the segmentation and classification pipeline we developed produced encouraging outcomes, demonstrating the promise of deep learning methods in medical image analysis for the identification of congenital heart disease.

	index	ASD	VSD	AVSD	ToF	TGA	DORV	CAT	CA	AAH	DAA	IAA	PA	APVC
0	1001	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1002	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1004	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	1005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	DSVC	PDA	PAS
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0

Figure 6.6: CHD type classification results

Chapter 7

Conclusion and Scope for Future Work

By combining CNN-based classification with 3D U-Net segmentation, our effort made major strides in the analysis of congenital heart disease (CHD) using medical imaging data. An 80 percent Dice score during the segmentation phase indicated accurate vessel delineation, which is essential for anatomical assessment. With a 90 percent accuracy rate in disease categorization, our CNN model improved the precision of diagnosis and treatment planning for various forms of congestive heart failure.

Our deep learning pipeline's achievement highlights how AI-driven medical imaging analysis has the potential to revolutionize cardiovascular treatment. Precise segmentation enables comprehensive anatomic assessment, and strong disease categorization enhances patient matching and tailored treatment plans. These developments have the potential to improve patient outcomes, clinical decision-making, and our comprehension of intricate cardiac diseases.

Future developments in segmentation robustness and classification generalization will keep driving the incorporation of deep learning into standard clinical practice. With regard to using AI for accurate and individualized cardiovascular diagnosis and therapy, our approach represents a major advancement.

7.1 Future Expansion

The following ideas can be considered for the future expansion of the model:

1. Advanced Segmentation Techniques: To increase the precision and resilience of vessel delineation, investigate more sophisticated segmentation methods and structures than 3D U-Net, such as active contour models, graph neural networks, and attention mechanisms.

2. Multi-Modal Fusion: Create a more thorough and detailed representation for analysis and diagnosis by combining data from several imaging modalities, such as MRI, CT, and ultrasound, using fusion techniques.

3. Predictive modeling for clinical outcomes, such as disease progression, response to treatment, or patient prognosis, can be extended to the project by utilizing patient-specific attributes and longitudinal imaging data.

4. Real-Time Analysis: Provide real-time analysis capabilities to support healthcare practitioners with prompt feedback and ongoing monitoring, allowing for timely intervention and decision-making.

5. Interactive Visualization: Make advantage of interactive visualization platforms or tools that enable doctors to easily engage and explore segmentation structures and classification results, improving interpretation and comprehension.

7.2 Future Scope

The future scope of our medical imaging analysis project for congenital heart disease using deep learning models is quite promising and offers several avenues for further development and impact:

1. Improved Accuracy and Robustness: By enhancing algorithms, fine-tuning hyperparameters, and implementing cutting-edge deep learning approaches, segmentation and classification models can be made more accurate and resilient over time.

2. Multi-Modality Integration: Investigate how to combine various imaging modalities, such CT, MRI, and echocardiography, to provide a comprehensive diagnostic platform that makes use of a variety of data sources to produce assessments that are more thorough and reliable.

3. Point-of-care and Real-Time Applications: Provide tools for real-time analysis that may be used at the point of care. This will help physicians make better decisions faster by using automated imaging analysis to improve patient outcomes and efficiency.

4. Clinical Decision Support Systems (CDSS): Construct a thorough clinical decision support system that combines patient data, clinical guidelines, imaging analysis, and predictive analytics to help medical professionals make well-informed treatment decisions.

5. Predictive modeling and Longitudinal Monitoring: Expand the project to incorporate predictive modeling for identifying high-risk patients, refining treatment strategies, and forecasting results, as well as longitudinal monitoring capabilities that track the course of the disease over time.

6. Telemedicine and Remote Monitoring: To enable remote consultations, ongoing patient monitoring, and collaborative treatment across geographic distances, integrate the project with telemedicine platforms and remote monitoring tools.

7. AI-Driven Personalized Medicine: Personalized and precision medicine techniques are made possible by using machine learning algorithms to customize treatment plans and interventions based on the unique characteristics of each patient, their genetic profiles, and how they respond to therapy.

8. Data Sharing and Collaborative Research: Establish shared datasets, benchmarking methodologies, and collaborative research projects with healthcare organizations and research groups to promote medical imaging analysis and innovation in cardiovascular healthcare.
Overall, the future scope involves enhancing accuracy, integrating modalities, real-time applications, clinical decision support, longitudinal monitoring, telemedicine, personalized medicine, collaborative research, regulatory compliance, and education initiatives. These advances aim to improve patient care, drive innovation, and responsibly deploy AI in cardiovascular healthcare.

Bibliography

[1] RJ. A. Steeden et al., "Rapid whole-heart CMR with single volume super resolution," Journal of Cardiovascular Magnetic Resonance, vol. 22, no. 1, pp.1-13, 2020.

[2] Z. Yao et al., "Graph matching and deep neural networks based whole heart and great vessel segmentation in congenital heart disease," Scientific Reports, vol. 13, no. 1,pp.1-11, 2023

[3] A. Yoshida et al., "Automated heart segmentation using u-net in pediatric cardiac CT," Measurement and Sensing, vol. 18, pp.100127, 2021.

[4] D. F. Pace et al., "Learned iterative segmentation of highly variable anatomy from limited data: Applications to whole heart segmentation for congenital heart disease," Medical Image Analysis, vol. 80, pp.102469, 2022

[5] X. Xu et al., "Whole heart and great vessel segmentation in congenital heart disease using deep neural networks and graph matching," in International Conference on Medical Image Computing and Computer Assisted Intervention, pp. 477-485, Springer, 2019.

[6] A. J. Powell and T. Geva, "Simple and complex congenital heart disease in infants and children," in Cardiovascular Magnetic Resonance, R. Y. Kwong et al., Eds., pp. 469-485, Elsevier, 2019.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234-241, Springer, 2015.

[8] X. Xu et al., "ImageCHD: A 3D Computed Tomography Image Dataset for Classification of Congenital Heart Disease," in Medical Image Computing and Computer Assisted Intervention – MICCAI 2020, ser. Lecture Notes in Computer Science, vol. 12264, 2020.

[9] R. Van Praagh, "The importance of segmental situs in the diagnosis of congenital heart disease," Seminars in Roentgenology, vol. 20, pp.254-271, 1985.

[10] S. Kanimi-Bidhendi et al., "Fully-automated deep-learning segmentation of pediatric cardiovascular magnetic resonance of patients with complex congenital heart diseases," Journal of Cardiovascular Magnetic Resonance, vol.22, no.80, 2020.

[11] C. Chen et al., "Deep Learning for Cardiac Image Segmentation: A Review," Frontiers in Cardiovascular Medicine, vol. 7, p. 25, 2020.

[12] A. Saxena, "Congenital heart disease in India: A status report," Indian Pediatrics, vol. 55, no. 12, pp. 1075-1082, 2018.

[13] L. Yu et al., "Automatic 3D cardiovascular MR segmentation with densely-connected volumetric convnets," in International Conference on Medical Image.

[14] J. M. Woltennk et al., "Dilated convolutional neural networks for cardiovascular MR segmentation in congenital heart disease," in Reconstruction, Segmentation, and Analysis of Medical Images, R. P. Poudel et al., Eds., pp. 95-102, Springer, 2016.

[15] A. Mukhopadhyay, "Total variation random forest: Fully automatic segmentation in congenital heart diseases," in Reconstruction, Segmentation, and Analysis of Medical Images, R. P. Poudel et al., Eds., pp. 165-171, Springer, 2016.

[16] J. Hofman and S. Kaplan, "The incidence of congenital heart disease," Journal of the American College of Cardiology, vol. 39, pp. 1890-1900, 2002.

[17] S-J. Yoo et al., "3D printing in medicine of congenital heart diseases," 3D Printing in Medicine, vol. 2, no. 3, 2016.

[18] J. Torrents-Barrena et al., "Segmentation and classification in moi and us fetal imaging: Recent trends and future prospects," Medical Image Analysis, vol. 51, pp. 61-88, 2019.

[19] Y. Jang et al., "Automatic segmentation of IV and LV in cardiac MRI," in International Workshop on Statistical Atlases and Computational Models of the Heart, pp. 161-169, Springer, 2017.

[20] Livne, Michelle Rieger, Jana Aydin, Orhun Taha, Abdel Aziz Akay, Ela Kossen, Tabea Sobesky, Jan Kelleher, John Hildebrand, Kristian Frey, Dietmar Madai, Vince. (2019). A U-Net Deep Learning Framework for High Performance Vessel Segmentation in Patients With Cerebrovascular Disease. Frontiers in Neuroscience. 13. 10.3389/fnins.2019.00097.

[21] Phung, Rhee,. (2019). A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. Applied Sciences. 9. 4500. 10.3390/app9214500.

[22] Yin XX, Sun L, Fu Y, Lu R, Zhang Y. U-Net-Based Medical Image Segmentation. J Healthc Eng. 2022 Apr 15;2022:4189781. doi: 10.1155/2022/4189781. Retraction in: J Healthc Eng. 2023 Oct 18;2023:9890389. PMID: 35463660; PMCID: PMC9033381.

[23] Renard, F., Guedria, S., Palma, N.D. et al. Variability and reproducibility in deep learning for medical image segmentation. Sci Rep 10, 13724 (2020). https://doi.org/10.1038/s41598-020-69920-0

[24] https://www.kaggle.com/datasets/xiaoweixumedicalai/imagechd

[25] https://blog.pregistry.com/tetralogy-fallot/

Chapter 8

Appendix

8.1 Segmentation Code

8.1.1 Imageload.py

```
from vedo import load

# Load the 3D image data
image = load("D:\\Users1\\HP1\\Downloads1\\CHD-Seg\\CHD Seg\\dataset
\\CHD_segmentation_dataset\\_ct_1001_image.nii.gz")

# Render the image
image.show()
```

8.1.2 Evaluation.py

```
import torch
from tqdm import tqdm
import numpy as np
from medpy.metric.binary import hd, dc
from utils import batch, plot_img_and_mask
import nibabel as nib
import re
import cv2
import os

# def normalize(image):
```

```
#     MIN_BOUND = np.min(image)
#     MAX_BOUND = np.max(image)
#     image = (image - MIN_BOUND) / (MAX_BOUND - MIN_BOUND)
#     image[image > 1] = 1.
#     image[image < 0] = 0.
#     return image

def eval_net(net, dataset, device, n_val):
    net.eval()
    tot = 0

    for i, b in tqdm(enumerate(dataset), total=n_val, desc='Validation round', unit='img'):
        img = b[0]
        true_mask = b[1][0]

        img = torch.from_numpy(img).unsqueeze(0)
        true_mask = torch.from_numpy(true_mask)

        img = img.to(device=device)
        true_mask = true_mask.to(device=device)

        mask_pred = net(img).squeeze(dim=0)

        mask_pred = mask_pred.cpu().detach().numpy()
        true_mask = true_mask.cpu().detach().numpy()
        mask_pred = np.argmax(mask_pred, 0)
        # print(np.unique(true_mask), np.unique(mask_pred))

        if (len(np.unique(true_mask)) == 1 and np.unique(true_mask) == [0]):
            n_val -= 1
        else:
            current_dice = 0
            classes = np.unique(true_mask)
            for c in [1, 2, 3, 4, 5, 6, 7]:
                gt_c_i = np.copy(true_mask)
                gt_c_i[gt_c_i != c] = 0
                pred_c_i = np.copy(mask_pred)
                pred_c_i[pred_c_i != c] = 0
                gt_c_i = np.clip(gt_c_i, 0, 1)
                pred_c_i = np.clip(pred_c_i, 0, 1)
                dice = dc(gt_c_i, pred_c_i)
```

```
                current_dice += dice
            # print(current_dice / (len(classes)-1))
            tot += (current_dice / (len(classes)-1))
            # plot_img_and_mask(true_mask, mask_pred)

    return tot / n_val

def get_3d_image_and_label(dir_path, image_name):

    map_need_crop_images = {"ct_1016_image.nii.gz": [50, 30, 480, 450],
                            "ct_1070_image.nii.gz": [40, 0, 480, 430],
                            "ct_1072_image.nii.gz": [0, 0, 512, 460],
                            "ct_1074_image.nii.gz": [0, 0, 512, 460],
                            "ct_1106_image.nii.gz": [40, 40, 430, 430],
                            "ct_1111_image.nii.gz": [20, 0, 480, 440],
                            "ct_1116_image.nii.gz": [40, 40, 450, 450],
                            "ct_1124_image.nii.gz": [50, 0, 470, 370],
                            "ct_1126_image.nii.gz": [120, 130, 430, 410],
                            }

    map_need_fill_images = {"ct_1004_image.nii.gz": [210, 0, 210, 0],
                            "ct_1021_image.nii.gz": [130, 0, 130, 0],
                            "ct_1022_image.nii.gz": [40, 0, 40, 0],
                            "ct_1027_image.nii.gz": [150, 0, 150, 0],
                            "ct_1070_image.nii.gz": [100, 0, 100, 0],
                            }

    image_path = os.path.join(dir_path, image_name)
    nimg = nib.load(image_path)
    img = np.asanyarray(nimg.dataobj)
    img = img.astype('float')
    img[img > 2000] = 2000

    label_name = image_name.replace('image', 'label')
    image_path = os.path.join(dir_path, label_name)
    nimg = nib.load(image_path)
    lab = np.asanyarray(nimg.dataobj)
    lab[lab > 7] = 0        #### some labels have values greater than 7

    result_img = np.zeros([512, 512, img.shape[2]], dtype='float')
    result_lab = np.zeros([512, 512, lab.shape[2]])
```

```
    for i in range(img.shape[2]):
        norm = img[:, :, i]
        crop_label = lab[:, :, i]

        if image_name in map_need_crop_images.keys():
            location = map_need_crop_images[image_name]
            norm = norm[location[0]:location[2], location[1]:location[3]]
            crop_label = crop_label[location[0]:location[2], location[1]:location[3]]

        if image_name in map_need_fill_images.keys():
            location = map_need_fill_images[image_name]
            norm = cv2.copyMakeBorder(norm, location[0], location[1],
            location[2], location[3], cv2.BORDER_CONSTANT, 0)
            crop_label = cv2.copyMakeBorder(crop_label, location[0],
            location[1], location[2], location[3], cv2.BORDER_CONSTANT, 0)

        norm = np.uint8(cv2.normalize(norm, None, 0, 255, cv2.NORM_MINMAX))
        norm = cv2.equalizeHist(norm)
        # norm = normalize(img[:, :, i])
        norm = cv2.resize(norm, dsize=(512, 512), interpolation=cv2.INTER_NEAREST)
        result_img[:, :, i] = norm / 255.0
        result_lab[:, :, i] = cv2.resize(crop_label, dsize=(512, 512),
        interpolation=cv2.INTER_NEAREST)

    return result_img, result_lab

#### reference from: https://www.creatis.insa-lyon.fr/Challenge/acdc/code/metrics_acdc.py
def eval_net_3D(net, test_image_names, device, dir_path, save_dir):
    net.eval()
    tot = []
    for image_name in test_image_names:
        img, lab = get_3d_image_and_label(dir_path, image_name)
        for i in range(lab.shape[2]):
            lab[:, :, i] = cv2.resize(lab[:, :, i], dsize=(512, 512),
            interpolation=cv2.INTER_NEAREST)
            current_image = img[:, :, i]
            current_image = torch.from_numpy(current_image)
            current_image = current_image.to(device=device)
            current_image = current_image.unsqueeze(dim=0)
            current_image = current_image.unsqueeze(dim=0)
            current_image = current_image.float()
```

```
            mask_pred = net(current_image).squeeze(dim=0)
            mask_pred = mask_pred.cpu().detach().numpy()

            mask_pred = np.argmax(mask_pred, 0)
            img[:, :, i] = mask_pred
            if save_dir is not None:
                num = re.findall('\d+', image_name)
                save_name = os.path.join(save_dir, str(num[0]).zfill(4) + '.' +
                str(i+1).zfill(4) + '.png')
                cv2.imwrite(save_name, mask_pred * 30)

        res = []
        for c in [1, 2, 3, 4, 5, 6, 7]:
            gt_c_i = np.copy(lab)
            gt_c_i[gt_c_i != c] = 0
            pred_c_i = np.copy(img)
            pred_c_i[pred_c_i != c] = 0

            gt_c_i = np.clip(gt_c_i, 0, 1)
            pred_c_i = np.clip(pred_c_i, 0, 1)

            dice = dc(gt_c_i, pred_c_i)

            res += [dice]
        print(image_name, res, sum(res)/len(res))
        tot += [res]
    tot = np.sum(tot, axis=0)
    tot = np.divide(tot, len(test_image_names))
    return tot
```

8.1.3 Prediction.py

```
import torch
from tqdm import tqdm
import numpy as np
from medpy.metric.binary import hd, dc
from net import UNet_GloRe
from eval import eval_net, eval_net_3D
from utils import get_imgs_and_masks, plot_img_and_mask, get_ids
```

```
import os
from sklearn.model_selection import KFold

def get_test_Dice():
    dir_path = './dataset/CHD_segmentation_dataset'
    save_dir = './dataset/CHD_predict'
    image_names = [f for f in os.listdir(dir_path) if f.endswith('image.nii.gz')
    and not f.startswith('._')]
    image_names = sorted(image_names)
    cross_validation = 4
    current_cross_validation = 1        ####0, 1, 2, 3
    save_dir = save_dir + '.' + str(current_cross_validation)
    if not os.path.isdir(save_dir):
        os.mkdir(save_dir)
    kf = KFold(n_splits=cross_validation)
    for i, (train_idx, test_idx) in enumerate(kf.split(image_names)):
        if i == current_cross_validation:
            train_names = np.array(image_names)[train_idx]
            test_names = np.array(image_names)[test_idx]
            break
    print('test image names:', test_names)
    if current_cross_validation == 2:
        test_names = np.delete(test_names, [7, 8, 9, 12], axis=0)   #### dirty data
        print('test image names:', test_names)


    net = UNet_GloRe(n_channels=1, n_classes=8)
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    net.to(device=device)
    net.load_state_dict(torch.load('./CHD-Seg/checkpoints/CP_epoch20.pth', map_location=device))
    test_score = eval_net_3D(net, test_names, device, dir_path, save_dir)

    print('test_score:', test_score)
    print('mean:', sum(test_score)/len(test_score))


if __name__ == '__main__':
    get_test_Dice()
```

8.1.4 Join.py

```
import numpy as np
import os
import nibabel as nib
import cv2
import re

def load_one_png(image_path):
    return cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

def stack_png_to_nii(png_dir, nii_save_path):
    png_files = [f for f in os.listdir(png_dir) if f.endswith('.png')]
    png_files.sort()  # Sort files to ensure proper order

    img = []
    for png_file in png_files:
        png_path = os.path.join(png_dir, png_file)
        img_slice = load_one_png(png_path)
        img.append(img_slice)

    img = np.array(img)
    img = img.transpose((1, 2, 0))  # Transpose to (width, height, depth)

    # Assuming you need to reverse the normalization process
    img = (img / 255.0) * 2000  # Assuming the original maximum value was 2000

    # Save as NIfTI
    nii_img = nib.Nifti1Image(img, np.eye(4))  # Assuming affine matrix is identity
    nib.save(nii_img, nii_save_path)

def _main():
    png_dir = './sample'  # Directory containing PNG images
    nii_save_path = './reversed.nii.gz'  # Path to save the reversed NIfTI file

    stack_png_to_nii(png_dir, nii_save_path)

if __name__ == '__main__':
    _main()
```

8.2 Classification Code

8.2.1 Classi.py

```
import os
import numpy as npr
import pandas as pd
import matplotlib.pyplot as plt
import nibabel as nib
from sklearn.model_selection import train_test_split
from tqdm import tqdm
from tensorflow.keras.models import Sequential
from keras.layers import Conv3D, MaxPooling3D, Flatten, Dense, Dropout
from tensorflow.keras.layers import BatchNormalization
from skimage.transform import resize
from google.colab import drive
drive.mount('/content/drive')

df=pd.read_csv('/content/drive/MyDrive/CHD/imageCHD_dataset_info_modified.csv')
print(df.head())  # Printing first five rows of the file

#df = df.iloc[:2000]  # Loading only first 2000 data points for memory reasons

# Define image directory
image_directory = "/content/drive/MyDrive/CHD/ImageCHD_dataset"
# Define function to load and resize NIfTI images
def load_and_resize_nifti_image(file_path, target_size):
    img = nib.load(file_path)
    data = img.get_fdata()
    # Resize the image to the target size
    resized_data = resize(data, target_size)
    return resized_data

    TARGET_SIZE = (100, 100, 100)
X_dataset = []
for i in tqdm(range(df.shape[0])):
    # Assuming IDs are in the format "ct_1001", "ct_1002", etc.
    file_name = 'ct_' + str(df['index'][i]) + '_label.nii.gz'
    img = load_and_resize_nifti_image(os.path.join(image_directory, file_name), TARGET_SIZE)
    X_dataset.append(img)
```

```
X = np.array(X_dataset)

y = np.array(df.drop(['index'], axis=1))

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

batch_size = X_test.shape[0]
num_classes = y_test.shape[1]

# Display the shape
print("Shape of (batch_size, num_classes):", (batch_size, num_classes))

model = Sequential()

model.add(Conv3D(filters=16, kernel_size=(3, 3, 3), activation="relu",
input_shape=TARGET_SIZE + (1,)))
model.add(BatchNormalization())
model.add(MaxPooling3D(pool_size=(2, 2, 2)))
model.add(Dropout(0.2))

model.add(Conv3D(filters=32, kernel_size=(3, 3, 3), activation='relu'))
model.add(MaxPooling3D(pool_size=(2, 2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv3D(filters=64, kernel_size=(3, 3, 3), activation="relu"))
model.add(MaxPooling3D(pool_size=(2, 2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv3D(filters=64, kernel_size=(3, 3, 3), activation='relu'))
model.add(MaxPooling3D(pool_size=(2, 2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(16, activation='softmax'))
```

```
model.summary()

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=100, validation_data=(X_test,
y_test), batch_size=22)

from keras.models import save_model

# Define the file path where you want to save the model
model_path = "/content/drive/MyDrive/CHD/image_classification_label.h5"

# Save the model
model.save(model_path)

print("Model saved successfully at:", model_path)

from keras.models import load_model
df = pd.read_csv('/content/drive/MyDrive/CHD/imageCHD_dataset_info_modified.csv')

new_image_path = "/content/drive/MyDrive/CHD/Copy of ct_1003_image.nii.gz"
model_path = "/content/drive/MyDrive/CHD/image_classification.h5"
TARGET_SIZE = (100, 100, 100)

def load_and_resize_nifti_image(file_path, target_size):
    img = nib.load(file_path)
    data = img.get_fdata()
    # Resize the image to the target size
    resized_data = resize(data, target_size)
    return resized_data

# Load and preprocess the new CT image
new_img = load_and_resize_nifti_image(new_image_path, TARGET_SIZE)
new_img = np.expand_dims(new_img, axis=0)

# Load the saved model
loaded_model = load_model(model_path)

# Make predictions using the loaded model
```

```
predictions = loaded_model.predict(new_img)

# Threshold the predictions to obtain binary predictions for each label
binary_predictions = (predictions > 0.3).astype(int)

# Get the names of the labels from your dataframe
label_names = df.drop(['index'], axis=1).columns

# Extract the predicted labels based on thresholding
predicted_labels = [label_names[i] for i in range(len(label_names)) if
binary_predictions[0][i] == 1]

# Display the predicted labels
print("Predicted Labels:", predicted_labels)

new_image_path = "/content/drive/MyDrive/CHD/data/ct_1003_image.nii.gz"

# Load and preprocess the new CT image
new_img = load_and_resize_nifti_image(new_image_path, TARGET_SIZE)
new_img = np.expand_dims(new_img, axis=0)  # Add batch dimension

# Make predictions using the trained model
predictions = model.predict(new_img)
print("Probability Scores:")
for i, score in enumerate(predictions[0]):
    print(f"Label {i}: {score}")

# Threshold the predictions to obtain binary predictions for each label
binary_predictions = (predictions > 0.3).astype(int)  # Using built-in int
instead of np.int

# Get the names of the labels from your dataframe
label_names = df.drop(['index'], axis=1).columns

# Extract the predicted labels based on thresholding
predicted_labels = [label_names[i] for i in range(len(label_names)) if
binary_predictions[0][i] == 1]

# Display the predicted labels
print("Predicted Labels:", predicted_labels)
```