Bachelor's Thesis

Bachelor's degree in Industrial Technology Engineering

Segmentation of Heart Chambers in 2-D Heart Ultrasounds with Deep Learning

Author: Ferran Craven-Bartle Corominas
Directors: Alicia Casals Gelpi; Maria Farahi
Date: June 2022

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

Echocardiography is a non-invasive image diagnosis technique where ultrasound waves are used to obtain an image or sequence of the structure and function of the heart. The **segmentation** of the heart chambers on ultrasound images is a task usually performed by experienced cardiologists, in which they delineate and extract the shape of both atriums and ventricles to obtain important indexes of a patient's heart condition.

However, this task is usually hard to perform accurately due to the poor image quality caused by the equipment and techniques used and due to the variability across different patients and pathologies. Therefore, **medical image processing** is needed in this particular case to avoid inaccuracy and obtain proper results.

Over the last decade, several studies have proved that **deep learning** techniques are a possible solution to this problem, obtaining good results in automatic segmentation. The major problem with deep learning techniques in medical image processing is the **lack of available data** to train and test these architectures.

In this work we have trained, validated, and tested a convolutional neural network based on the architecture of **U-Net** for 2D echocardiogram chamber segmentation. The data used for the training of the convolutional neural network was the B-Mode 4-chamber apical view **Echogan** dataset with data augmentation techniques applied. The novelty of this work is the hyperparameter and architecture **optimizations** to reduce the computation time while obtaining significant training and testing accuracies.

Acknowledgments

I would like to show my deep appreciation to my thesis directors Alicia Casals and Maria Farahi for always providing me with help and guidance to be able to develop this project. I would also like to express my gratitude to all the people who have supported me during this project.

Contents

1. Introduction

In this chapter, the theoretical fundamentals of the work are presented to the reader to understand the purpose of the work. These concepts come from two different areas: the medical area and the engineering area.

In the first place, the **medical context** consists of a brief introduction to the heart anatomy to understand which are the different parts of the heart. This section is followed by an explanation of the most used echocardiography modalities and the parts of the heart that can be seen in each of them. In the final part of the medical context, the most common heart diseases will be enumerated and related to each part of the heart.

Once the medical area is approached, the focus will be on the engineering area: **Deep Learning**. The basic concepts such as layers, activation function, weights, and others will be presented. Subsequently, convolutional neural networks will be explained briefly, and finally, the most used architectures for echocardiogram segmentation will be analyzed.

1.1. Medical Context

In this section, a medical approach to the heart anatomy and ultrasound diagnosis will be presented to further understand the purpose, objectives, methodology, and results of this work.

1.1.1. Heart Anatomy

The **heart** is one of the most important parts of a living being. This organ is the one in charge to pump the blood to all parts of the body through the circulatory system. Blood contains oxygen and other vital nutrients to live. Blood also carries metabolic waste such as carbon dioxide to the lungs. In humans, the heart is approximately the size of a closed fist and is located between the lungs, in the middle compartment of the chest [1] [2].

The human heart has four chambers (see Figure 1): two superior **atria** and two inferior **ventricles**. The atria are the blood receiving chambers, and the ventricles are the blood discharging chambers. A wall of muscle called the **septum** separates the left and right atria and the left and right ventricles. Each chamber (atriums and ventricles) is connected through a **valve** to the next cavity the blood flows into. These valves are: Tricuspid valve or right atrioventricular valve, between the right atrium and right ventricle; Mitral valve or bicuspid valve, between the left atrium and left ventricle; Pulmonary valve, located at the opening

between the right ventricle and the pulmonary trunk; Aortic valve, located at the opening between the left ventricle and the aorta.

The heart is enclosed in a triple-walled sac called the **pericardium**. The pericardium is composed of three layers. The outer layer is called the **epicardium**, the outermost protective layer of the heart. The epicardium is composed of mesothelium, a cell type that covers and protects the heart. The middle layer is called the **myocardium** and is composed of contractile cardiac muscle. The inner layer is called the **endocardium** and is in contact with the blood that the heart pumps.

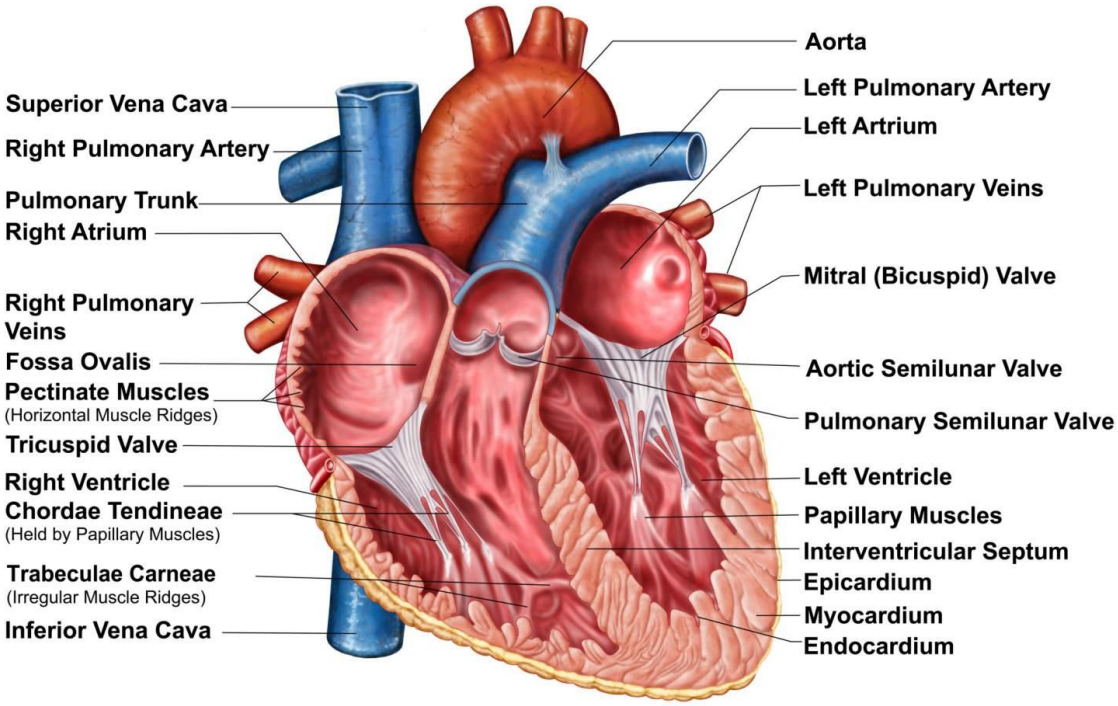


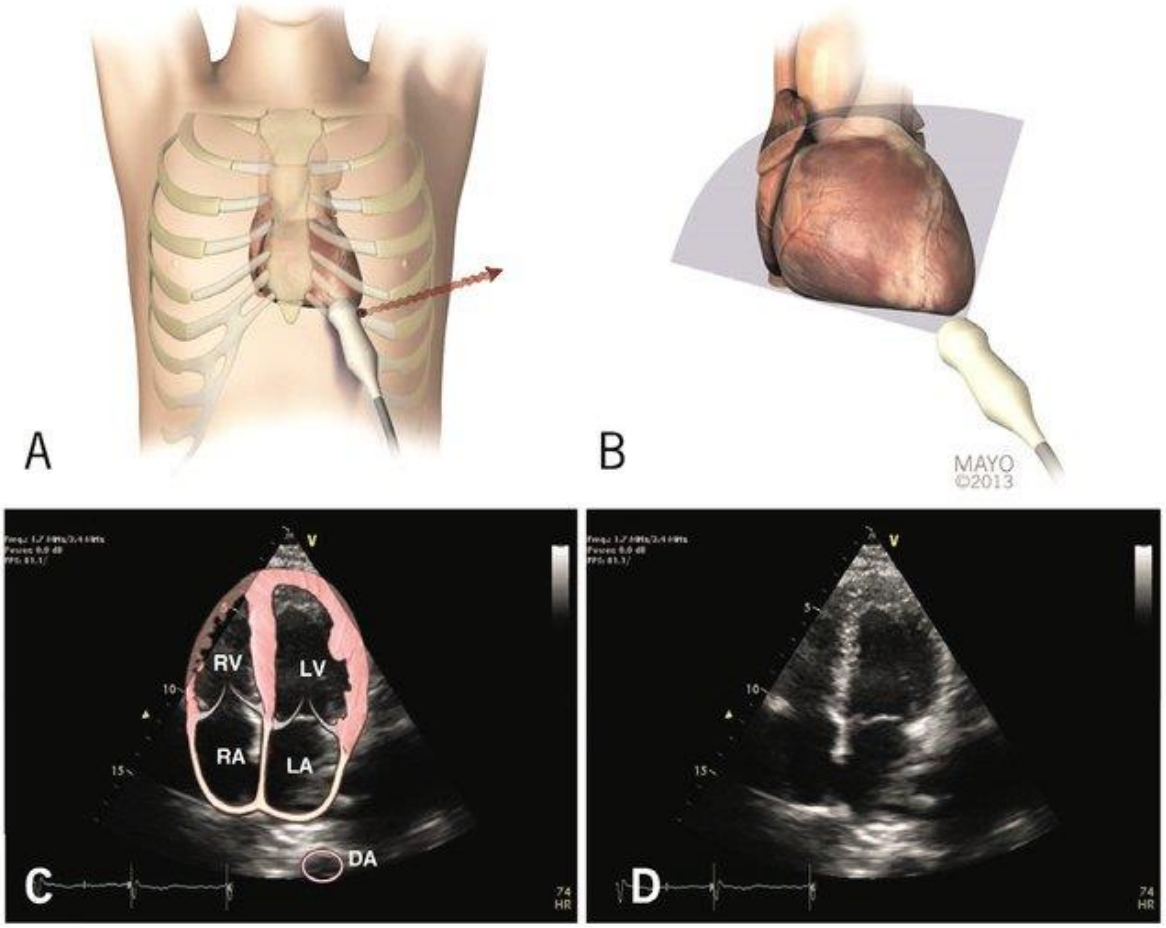
Figure 1. Heart Anatomy.

Source: Stocktrek Images/Getty Images

1.1.2. Echocardiography

An **echocardiogram**, cardiac echo, or simply an echo, is a type of medical imaging of the heart, using standard ultrasound technique or Doppler ultrasound technique. Ultrasound waves produced by piezoelectric crystals travel from the **transducer** through the inside of the body to the inside of the heart, and they return to the transducer, where these waves are

converted into electrical signals that will eventually result in the ultrasound image or sequence (see Figures 2).



Figures 2. Ultrasound waves are sent by the transducer to the heart to obtain the echocardiogram.
Source: Diaz-Gomez, Jose & Via, Gabriele & Ramakrishna, Harish. (2016). Focused cardiac and lung ultrasonography: Implications and applicability in the perioperative period.

Echocardiography can provide a wealth of helpful information, including the **size and shape** of the heart (internal chamber size quantification), pumping capacity, the location and extent of any tissue damage, and assessment of valves and cardiac masses. An echocardiogram can also give physicians other estimates of **heart function**, such as a calculation of the cardiac output, ejection fraction, and diastolic function (how well the heart relaxes).

Echocardiography can be classified into two different classes: types and modes. Types depend on where the transducer obtains the images or sequences from. Modes depend on the characteristics of the results obtained [3]. The four types are:

- **Transthoracic echocardiogram**, which is the most common type of echocardiogram and is non-invasive, taking place entirely outside the body. The gel is applied to the patient and a handheld transducer is used to scan the heart.

- **Intracardiac echocardiography** represents a newer form of testing, with images taken inside the heart. It involves the placement of thin tubes called catheters inside the patient's arteries.

- **Stress echocardiogram** uses ultrasound imaging of the heart to assess the wall motion in response to physical stress. First, images of the heart are taken "at rest" to acquire a baseline of the patient's wall motion at a resting heart rate. The patient then exercises to increase the heart rate. Finally, images of the heart are taken "at stress" to assess wall motion at the peak heart rate.

- **Transesophageal echocardiogram** is an alternative way to perform an echocardiogram. A specialized probe containing an ultrasound transducer at its tip is passed into the patient's esophagus. This allows image and Doppler evaluation from a location directly behind the heart.

Depending on the mode:

- **B-mode / 2D**: Brightness mode is often synonymous with "2-D" and is very commonly used in echocardiography.

- **M-mode**: Motion mode is infrequently used in modern echocardiography. It has specific uses and has the benefit of very high temporal fidelity (e.g., measuring LV size at end-diastole).

- **Doppler echocardiogram**: This technique is used to measure and assess the flow of blood through the heart's chambers and valves. Also, Doppler can detect abnormal blood flow within the heart, which can indicate a problem with one or more of the heart's four valves, or with the heart's walls.

- **3-D echo**: Echocardiography typically shows a flat picture, but our machines can also create 3-D imaging. This technology is particularly helpful for identifying problems with heart valves, replacement heart valves, and the heart's lower left chamber (left ventricle).

In our study, **B-Mode/2D transthoracic echocardiogram** images will be used. In this type of image, the most common cross-sectional views are the parasternal long axis, the parasternal short axis, and the apical view (see Figure 3).

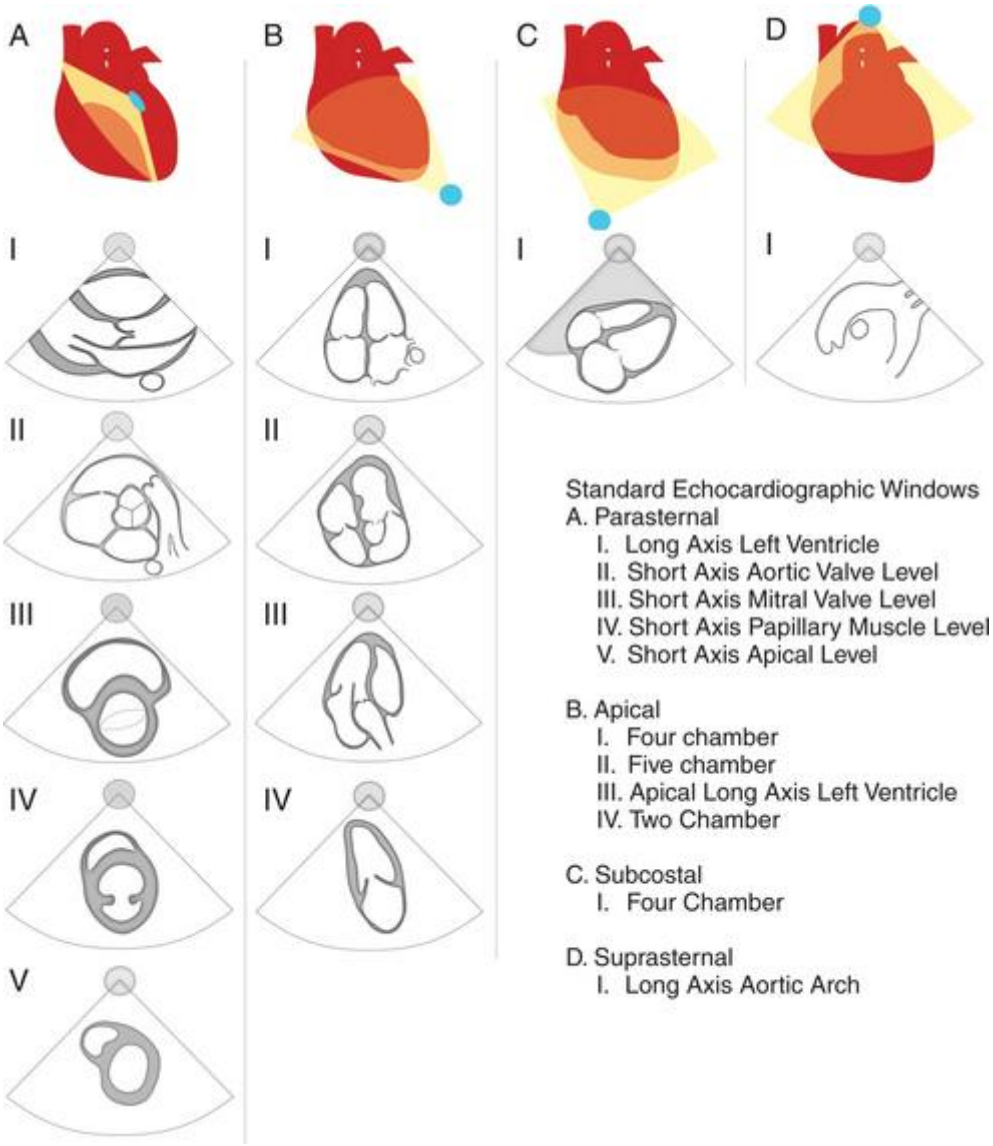


Figure 3. Standard Echocardiographic view types.
Source: https://radiologykey.com/echocardiography-2

In this study, we will focus on the **Four Chamber Apical View** (window B.I above), which gives general information about the four chambers of the heart. This is a good view to assess the chamber's size and function. It also can assess valve morphology (see Figure 4).

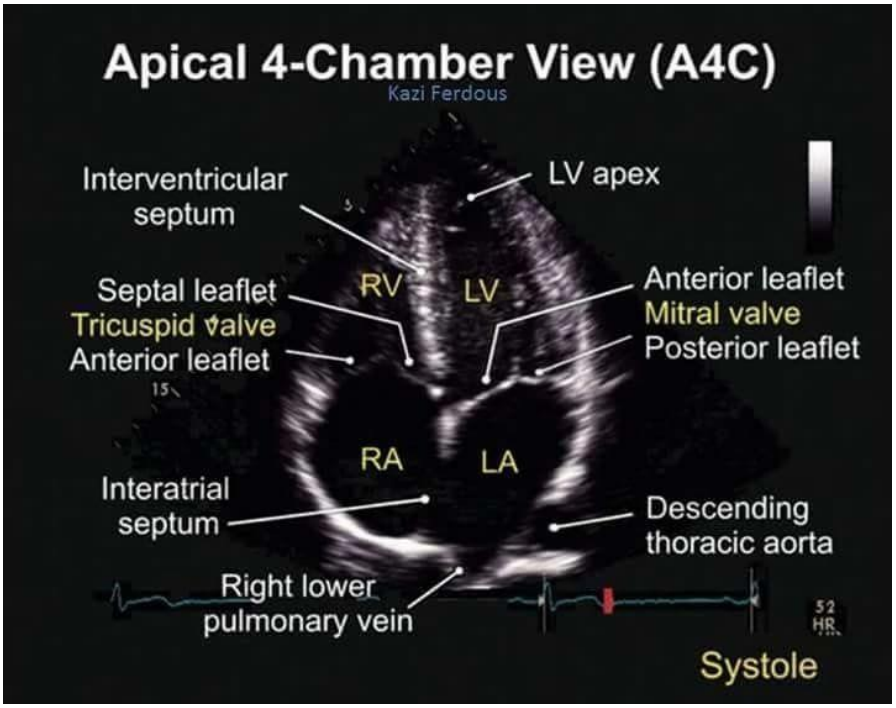


Figure 4. Typical image of an Apical 4-Chamber View with its parts.
Source: *https://www.pinterest.es/pin/130534089180243360/*

1.1.3. Heart Diseases

Echocardiography is an important tool in assessing abnormalities and reaching an early diagnosis of some **heart diseases**. Also, it is important in treatment and follow-up in patients with heart failure, by assessing ejection fraction for example [4]. Echocardiography can help detect cardiomyopathies, such as hypertrophic cardiomyopathy and many others. The approximate normal values for various cardiac structures are described in Table 1.

Normal ranges for measures of systolic and diastolic function		
Echocardiography		
Fractional shortening (%)	28–44	
Doppler		
Systolic velocity integral (cm)	15–35	
Mitral valve E (cm/s)	44–100	
Mitral valve A (cm/s)	20–60	
E:A ratio	0.7–3.1	
Tricuspid valve E (cm/s)	20–50	
Tricuspid valve A (cm/s)	12–36	
E:A ratio	0.8–2.9	
Time intervals		
Mitral E deceleration time (ms)	139–219	
Mitral A deceleration time (ms)	>70	
Isovolumic relaxation time (ms)	54–98	
Normal intracardiac dimensions (cm)		
	Men	**Women**
Left atrium	3.0–4.5	2.7–4.0
LV diastolic diameter	4.3–5.9	4.0–5.2
LV systolic diameter	2.6–4.0	2.3–3.5
IV septum (diastole)	0.6–1.3	0.5–1.2
Posterior wall (diastole)	0.6–1.2	0.5–1.1

Table 1. The approximate normal values for various cardiac structures.
IV: interventricular; LV: left ventricular.
Source: https://www.ncbi.nlm.nih.gov/books/NBK2215/

One of the most important parameters in echocardiography to assess the heart is the ejection fraction. The **ejection fraction** (EF) is a measurement, expressed as a percentage, of how much blood the left ventricle pumps out with each contraction. An ejection fraction of 60 percent means that 60 percent of the total amount of blood in the left ventricle is pushed out with each heartbeat. A normal heart's ejection fraction may be between 50 and 70 percent [5]

Abnormal parameters in an echocardiogram can be caused by a range of heart diseases. The most common and important ones are enumerated below (see Table 2). In addition, the following **features** that can be seen in an echocardiogram can give information for the doctor to determine which diseases are affecting the heart performance:

- **Systolic dysfunction**: LV systolic dysfunction is assessed using the ejection fraction (the percentage of the end-diastolic volume ejected during systole). In most cases, this is estimated by eye from all the available echo views. A normal ejection fraction is 50%–80%, but values as low as 5% are compatible with life (end-stage heart failure).

- **Diastolic dysfunction**: a normal LV ejection fraction in the presence of heart failure syndrome leads to a search for diastolic dysfunction. Typical echo findings in diastolic dysfunction are normal LV cavity size, thickened ventricle, and reversed E/A ratio.

- **Wall-motion abnormality**: When ischemia occurs, contractile abnormalities of segments of the myocardium can be detected by echo before the appearance of electrocardiogram (ECG) changes or symptoms. Therefore, echo can be a valuable tool in the diagnosis of acute myocardial infarction. In this situation, it offers some measure of the extent of the infarct and a screen for complications, such as ventricular septal defect (VSD).

- **Valve assessment**: Echo is the tool of choice for the assessment of valvular abnormalities. Different related types of diseases can be diagnosed.

- **Embolic sources**: The primary cardiac sources for embolism are: an akinetic ventricular segment, an LV aneurysm, and the atrial appendage. These are best visualized with Transesophageal Echocardiogram.

In this work, as mentioned before, we focus on the 2D apical four-chamber view. Common measurements in this type of view are the Left Ventricle ejection fraction and Left Atrial dimensions [6]. This is the reason why most databases and articles (explained in section 1.2.3) only focus on the segmentation of the Left Ventricle and Left Atrium.

However, doctors need to have a **view of all four chambers and two valves** instead of only the Left Ventricle and Atrium because sometimes the disease affects the other parts of the heart. Some examples are shown below (see Figure 5 and Figure 6). This is the reason why in the current work we will focus on obtaining labeled four-chamber datasets.

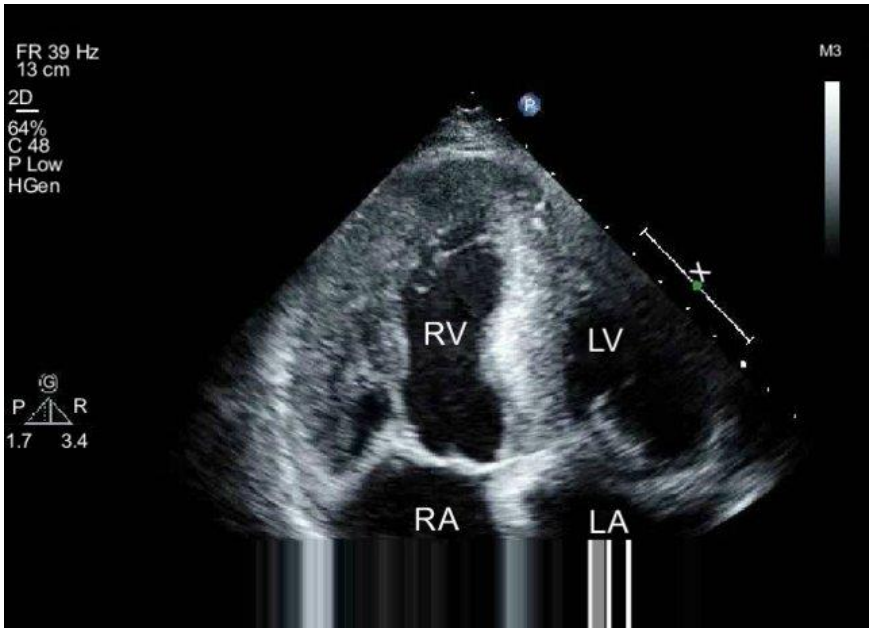


Figure 5. A four-chamber apical view echocardiogram showing biatrial dilatation, valve thickening, thick ventricular walls, and interventricular septum with a speckled appearance, which suggests amyloid infiltrate.
Source: Xu, Zhan-Wen & Li, Ya-Qin & Liu, Li-Xia & Zhou, Bing-Juan. (2015). Light-chain cardiac amyloidosis with neuropathy: A case report. Clinical interventions in aging. 10. 1219-22. 10.2147/CIA.S87540.

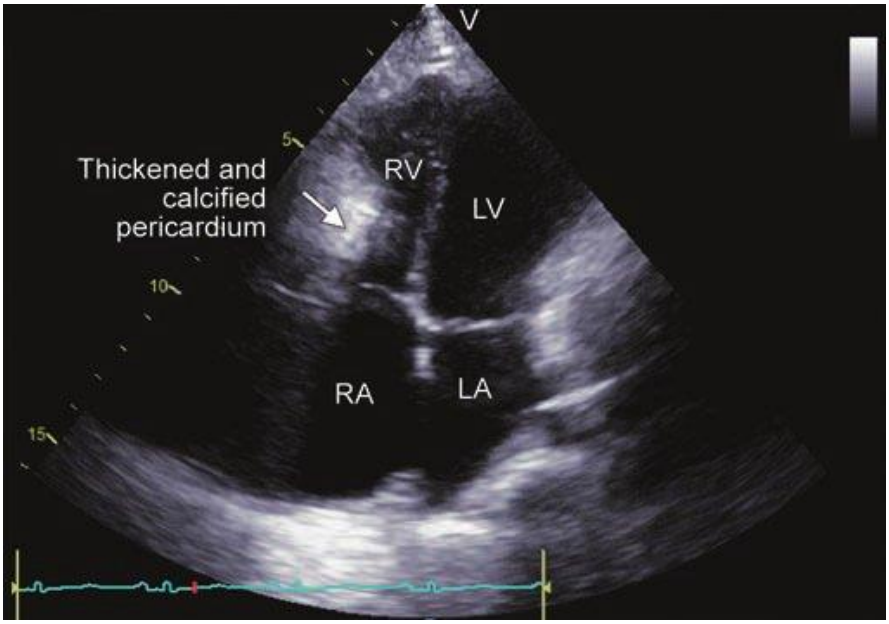


Figure 6. Transthoracic apical four-chamber view showing the thickened pericardium with calcium plaques obscuring the full view of the right atrium.
Source: Ganesan, Rajarajan & Kumar, Bhupesh & Munirathinam, Ganesh & Bhat, Imran & Mahajan, Sachin. (2017). Modification in Surgical Plan following Intraoperative Detection of Co-existent Right Atrial Thrombus by Transesophageal Echocardiography in Chronic Constrictive Pericarditis. Journal of Perioperative Echocardiography. 5. 34-37. 10.5005/jp-journals-10034-1068.

It is important to mention that diseases in the heart can affect **multiple parts** of it, as we have seen in the examples before. The heart can have multiple chambers or valves affected especially when the disease is in an advanced stage. For this reason, it is important to analyze as many parts of the heart as the ultrasound lets. This way, early diseases can be detected in the part of the heart where they originated, so the causes are easier to distinguish than in the case where all of the heart is already affected. Table 2 below shows **the most common heart diseases** that can be detected by the chosen echocardiography modality.

These diseases put at very risk the health of the patients. Therefore, heart diseases need to be detected **at a very early stage** to prevent them from evolving into a worse diagnostic. This is why echocardiograms have to be accurately segmented. As will be explained in section 1.2.3, **Deep Learning** methods applied to echocardiogram segmentation are used to help the experts to analyze heart diseases at a very early stage.

Condition/Dysfunction	Parts involved	Brief Explanation	Features					
			Abnormal Ventricles/Atriums Movement	Abnormal Ventricles/Atriums Area	Abnormal Valve Movement	Abnormal Valve Area	Abnormal Heart Walls Area and Movement	Abnormal blood flow
Valve Insufficiency and Regurgitation	Tricuspid valve Pulmonary valve Mitral valve Aortic valve	Is characterized by the **failure** of one or more of the heart valves **to close perfectly** resulting **blood flowing backwards** across the valve (valvular regurgitation or leaking).	x	x	x	x		x
Valve Stenosis	Tricuspid valve Pulmonary valve Mitral valve Aortic valve	Occurs when a heart's valve narrows. The valve **doesn't open fully**, which reduces or blocks blood flow from your heart into the blood cavity belonging.	x	x	x	x		
Heart failure	All	Is a condition that develops when the heart **doesn't pump enough blood** for the body's needs. This can happen if a heart can't fill up with enough blood or when **a heart is too weak** to pump properly.	x	x	x	x	x	x
Hypertrophic Cardiomyopathy	Myocardium Ventricular Septum	Is a disease in which **the heart muscle becomes abnormally thick** (hypertrophied). The thickened heart muscle can make it harder for the heart to pump blood.	x	x			x	
Dilated Cardiomyopathy	Myocardium Left Ventricle Right Ventricle and Atriums	Is a disease of the heart muscle that usually starts in the heart's main pumping chamber (left ventricle). **The ventricle's wall stretches and thins (dilates)** and can't pump blood as well as a healthy heart can. Over time, both ventricles may be affected.	x	x			x	
Restrictive Cardiomyopathy	Left Ventricle Ventricular Septum Atrial Septum Right Atria	Is a condition where **the chambers of the heart become stiff over time**. Though the heart is able to squeeze well, it's not able to relax between beats normally. This makes it harder for the heart to fill with blood. The blood backs up in the circulatory system.	x	x			x	
Ventricular/Atrial Fibrillation and Flutter	Both parts of Atriums and Ventricles	Is a type of abnormal heart rhythm (arrhythmia) where **disorganized heart signals** cause the lower heart chambers (ventricles) to **twitch (quiver)** uselessly. As a result, the heart doesn't pump blood to the rest of the body.	x	x				
Endocarditis	Endocardium and valves	Is a life-threatening **inflammation** of the inner lining of the heart's chambers and valves (endocardium), usually **caused by an infection**.	x	x		x	x	
Pericarditis	Pericardium	Is a **inflammation** of the pericardium, a sac-like structure with two thin layers of tissue that surround the heart to hold it in place and help it work. A small amount of fluid keeps the layers separate so there's less friction between them as the heart beats.	x	x			x	
Myocardial Ischemia	Myocardium Blood cavities	Myocardial ischemia occurs when blood flow to the heart muscle (myocardium) **is obstructed** by a partial or complete blockage of a coronary artery by a **buildup of plaques** (atherosclerosis).	x	x	x	x	x	x
Cardiac Embolisms/Atheroesclerosis	All the heart and blood cavities	Is an **obstruction** that travels from the heart to lodge in a blood vessel. An embolus can be made up of **fatty material**, or it can be a **blood clot**.	x	x				x
Inherit cardiac pathologies: - Tetralogy of Fallot - Ventricular septal defect	4 chambers and septums	A defect in the interventricular septum allows **communication between the systemic and pulmonary circulations**. As a result, flow moves from a region of high pressure to a region of low pressure—that is, from the LV to the RV (a left-to-right shunt).	x	x				x

Table 2. Common diseases and features in heart ultrasound investigation.

·Green: Features that can be seen in 4-chamber apical view images.
·Blue: Diseases related to the features.
·Yellow: Parts affected by the diseases related to the features.

1.2. Deep Learning

Once the medical context is explained, the focus is now on the engineering area. **Deep learning** is part of a broader family of **machine learning** methods based on artificial neural networks with representation learning (see Figure 7).

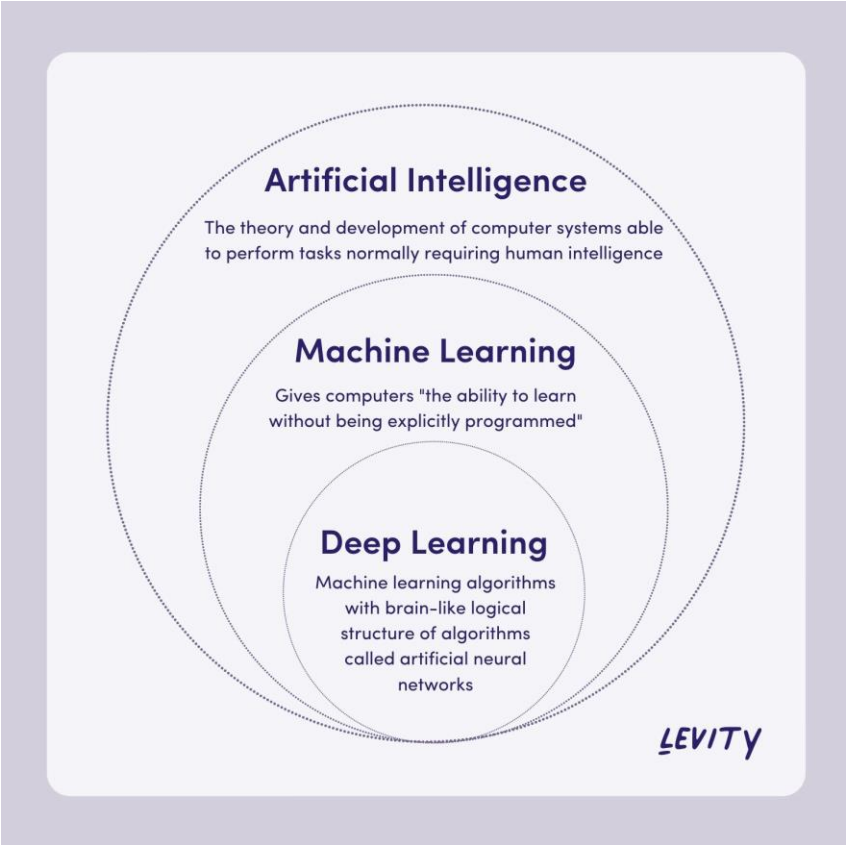


Figure 7. Definitions of AI, ML, and DL.
Source: https://levity.ai/blog/difference-machine-learning-deep-learning

Deep learning techniques use multiple layers to progressively extract higher-level **features** from the raw input. The major part of modern deep learning models is based on artificial neural networks [7].

1.2.1. Basic Concepts

Artificial Neural Networks are computing systems inspired by the brain's biological neural network [8] [9]. An ANN contains nodes called **neurons**, which are the simile to a biological neuron in the brain. Nodes are connected by **edges** just as neurons in the brain connect each other by synapses. Each node sends information through an edge that will be computed in the next node. One neuron can be connected to multiple neurons. The neurons in the architecture of an ANN are commonly grouped by **layers** (see Figure 8).

First is the **input layer**. This layer will accept the data and pass it to the rest of the network. The second type of layer is called the **hidden layer**. Hidden layers are either one or more in number for a neural network. Hidden layers are the ones that are responsible for the excellent performance and complexity of neural networks. They perform multiple functions at the same time, such as data transformation, automatic feature creation, etc. The last type of layer is the **output layer**, which holds the result or the output of the problem. Raw images get passed to the input layer and output is received in the output layer.

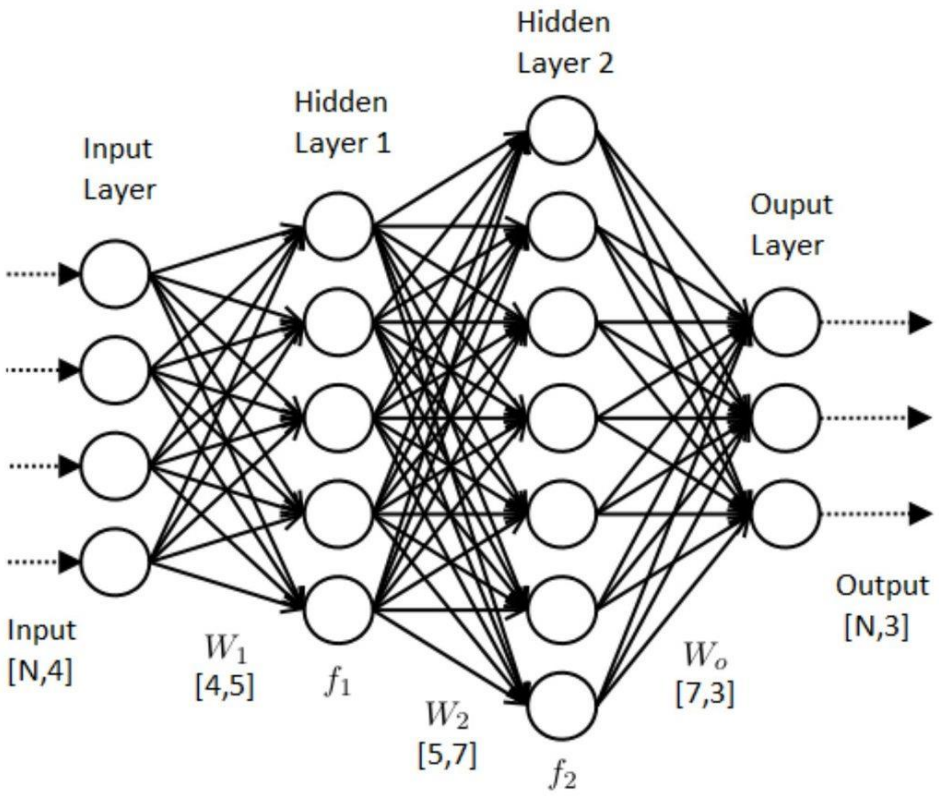


Figure 8. Common structure of an ANN, containing
input and output layers with multiple hidden layers between. Under, weights and activations functions.
Source: https://www.datasciencecentral.com/the-artificial-neural-networks-handbook-part-1/

An Artificial Neural Network is designed to learn to do a task just as the brain learns to do so. The ANN architecture has **weights** related to the edges, which are constantly being updated as the training of the network is performed. Weights convey the importance of each feature in predicting the output value. Features with weights that are close to zero are said to have less importance in the prediction process compared to the features with larger weights.

Another variable inside the network is **bias**. Bias allows the network to shift the activation function by adding a constant (the given bias) to the input. Bias in Neural Networks can be thought of as analogous to the role of a constant in a linear function, whereby the line is effectively transposed by the constant value [10].

On the other hand, the one in charge to define the output of a given neuron inside the network is the **activation function** [11]. An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simple mathematical operations (see Figure 9). The purpose of an activation function is to add non-linearity to the neural network.

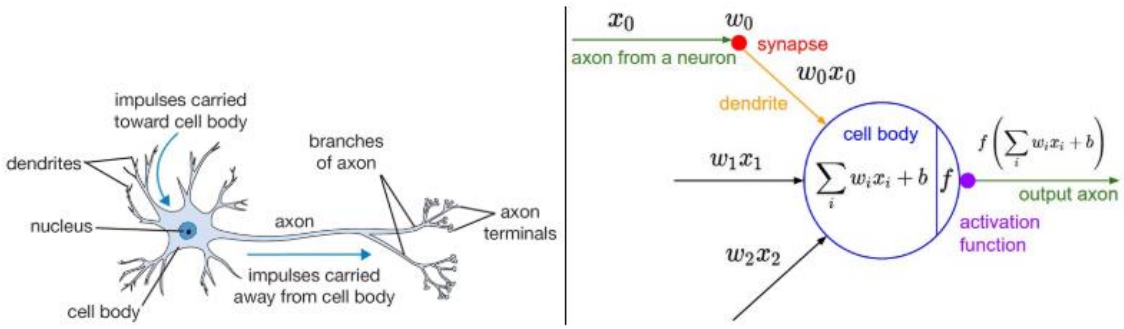


Figure 9. A comparison of a biological network with a neuron in an ANN. The output value of a neuron is the outcome of applying the activation function to the sum of the weighted inputs plus the biases.
Source: cs231n by Stanford

All the explained variables determine the structure or architecture of a Neural Network, but how do these systems learn to do a task? By an **optimizer** and a **loss function**.

An **optimizer** is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improving the accuracy. The problem of choosing the right weights for the model is a daunting task, as a

deep learning model generally consists of millions of parameters. It raises the need to choose a suitable optimization algorithm for each application.

On the other hand, the **loss function** in a neural network quantifies the difference between the expected outcome and the outcome produced by the machine learning model. From the loss function, we can derive the gradients which are used by the optimizer to update the weights. The average of all losses constitutes the cost.

There are many **types of neural networks** available. They can be classified depending on their: Structure, Data flow, Neurons used and their density, Layers and their depth activation filters, etc. [12]

Here is a list of different types of neural networks that exist: Perceptron, Feed Forward Neural Network, Multilayer Perceptron, Convolutional Neural Network, Radial Basis Functional Neural Network, Recurrent Neural Network, etc.

As will be seen in section 1.2.3, Convolutional Neural Networks are claimed to obtain good results in echocardiogram segmentation [24]. Therefore, in the current work, we will focus on a **Convolutional Neural Network**.

1.2.2. Convolutional Neural Networks

Once the basics of neural networks are explained, let's focus on Convolutional Neural Networks. In Deep Learning (DL), a **Convolutional Neural Network** (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image and assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. [13]

Convolutional neural networks are a specialized type of artificial neural network that uses a mathematical operation called **convolution** in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing.

But what exactly is a convolutional layer? In Figure 10 the operation of a convolution layer is represented:

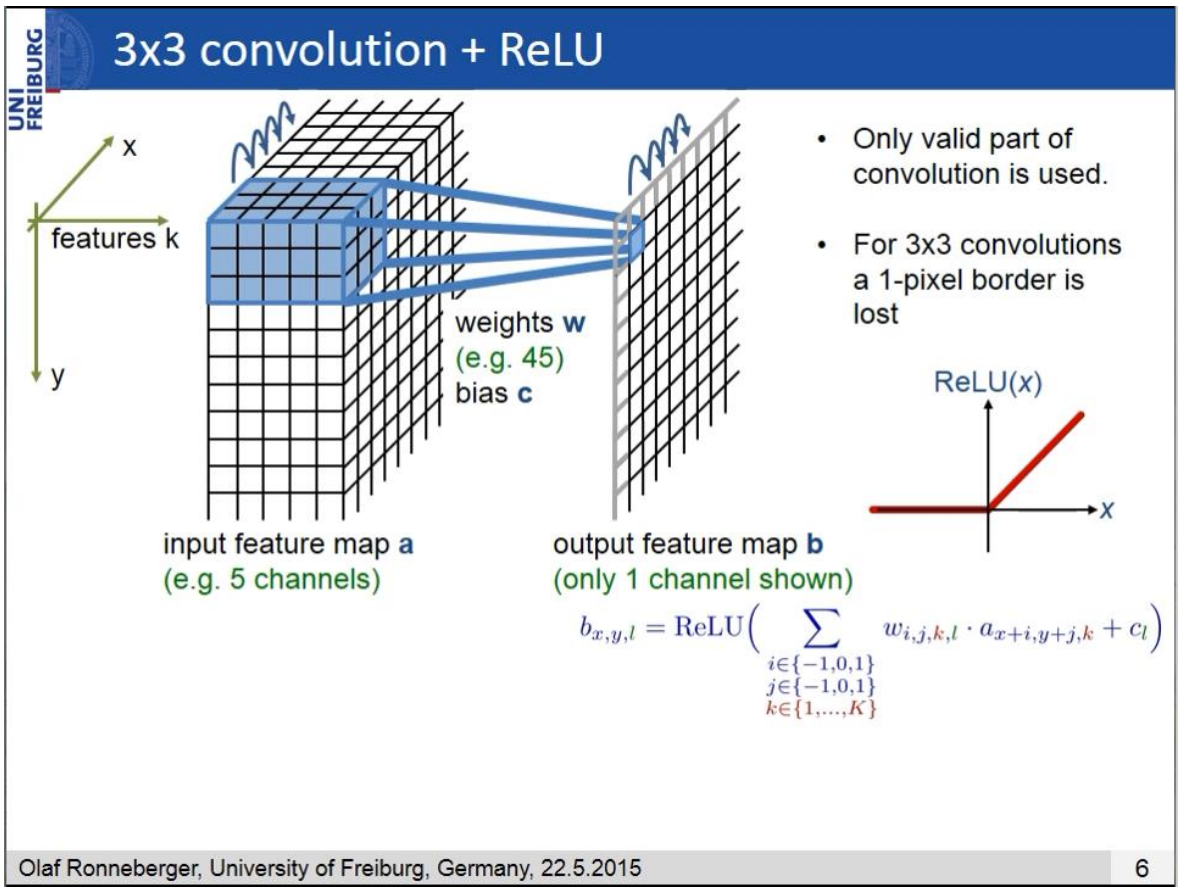


Figure 10. Example of a convolution layer in a CNN followed by a ReLU activation function.
Source: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. A **convolution** converts all the pixels in its receptive field into a single value (see Figure 11). For example, if you would apply convolution to an image, you will be decreasing the image size as well as bringing all the information in the field together into a single pixel. The final output of the convolutional layer is a matrix.

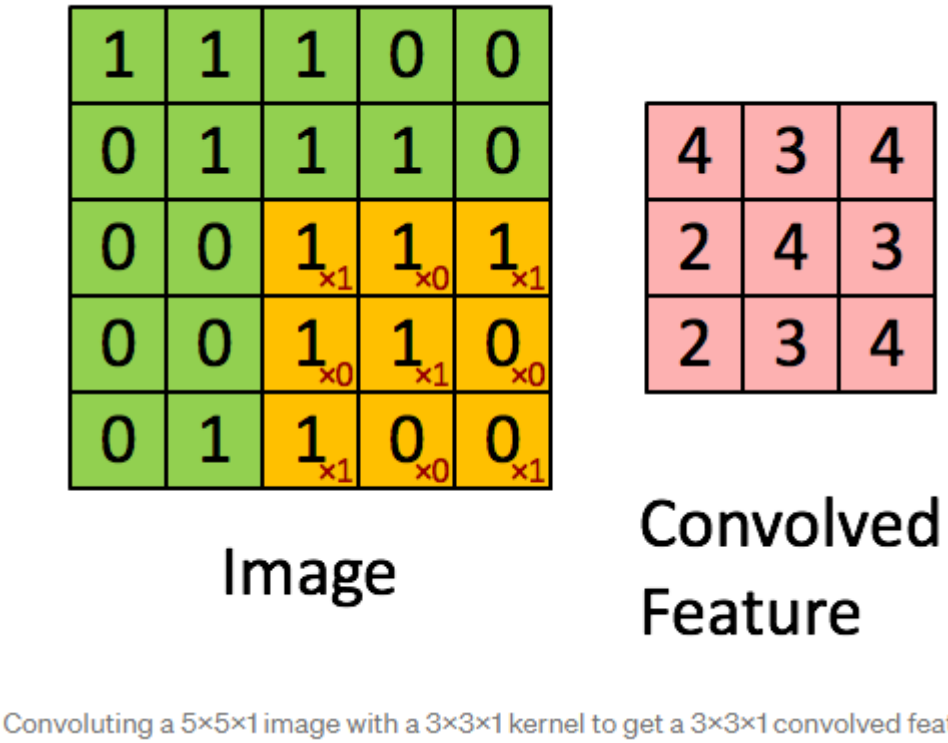


Figure 11. Each cell in the output matrix is the sum of the products of each cell in the orange matrix and their related weights (x1; x0; x1; etc., also called kernel) passing by all cells in the green matrix.
Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field.

The objective of the Convolution Operation is to extract the high-level features such as objects or complex patterns, from the input image. Conventionally, lower or earlier layers identify **Low-level features** like edges and colors, while higher layers identify **High-level features** such as concepts or complex patterns, i.e., digits, letters, or faces.

The other type of layer commonly used in CNN is the **Pooling layer**. Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting **dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training the model.

There are two types of Pooling: **Max Pooling** and **Average Pooling** (see Figure 12). Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better** than Average Pooling [13].

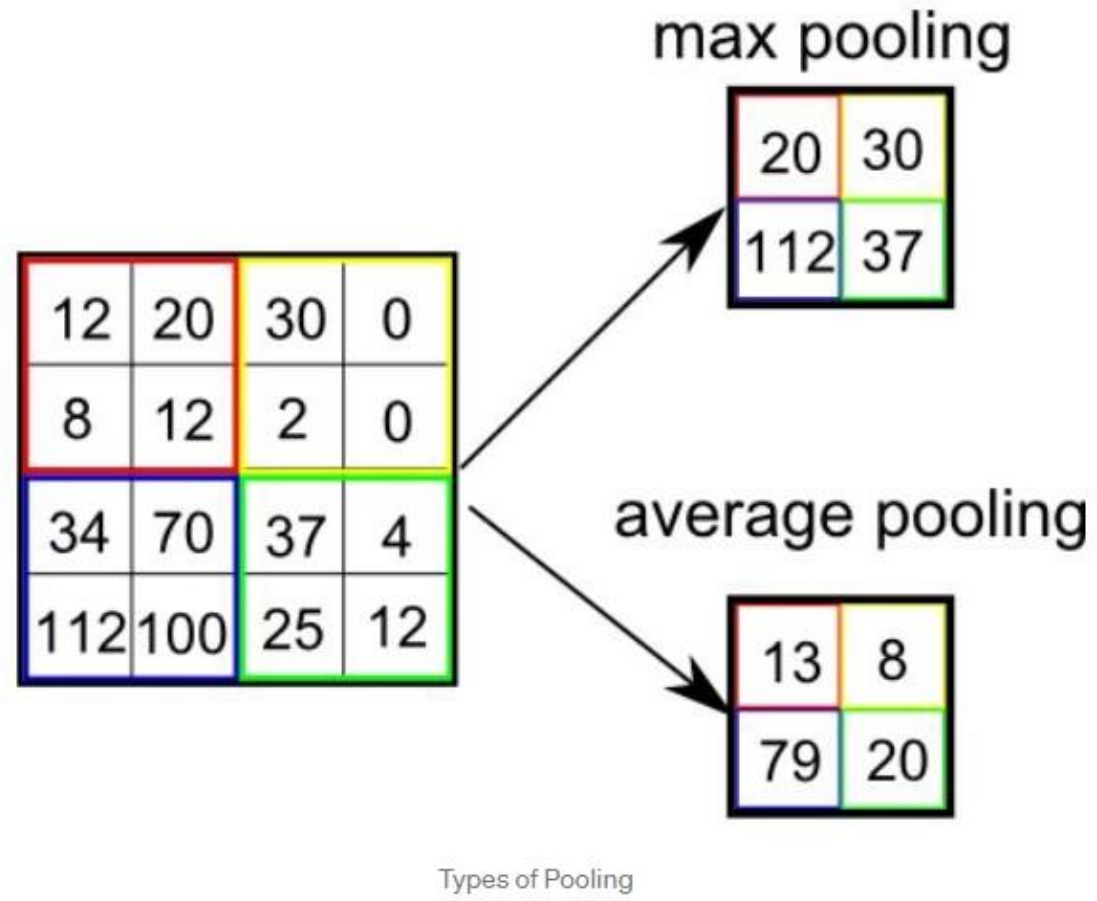


Figure 12. Each type of pooling.
Source: *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*

There are various architectures of CNNs available which have been key in building algorithms that power and shall power AI as a whole in the foreseeable future. Some of them have been listed: LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet, U-Net, etc.

The following section will present a general review of articles where CNNs have been used for the task of echocardiogram segmentation.

1.2.3. Convolutional Neural Networks in Echocardiogram Segmentation

Once the theoretical concepts of both medical and engineering areas have been presented, we have reached the topic of this work: **echocardiogram segmentation**. As explained in section 1.1.2, echocardiograms generally need to be segmented to obtain measurements and indexes of the shape and performance of the chambers, valves, and walls of the heart. Segmenting means creating a **ground-truth** or **mask** delineating the regions of interest (ROI), as can be seen in Figure 13 below:

Figure 13. Typical images extracted from a segmented echocardiogram labeled dataset. At left is the original ultrasound image and at right is its related ground-truth image.
Source: [18]

Cardiologists have traditionally performed this task, but several studies showed that this manual segmentation suffered from **subjective interpretation** and inter- and intra-cardiologist **variability** [14][15]. The inherent causes of this variability are well documented [16]:

- Poor contrast between the myocardium and the blood pool.
- Brightness inhomogeneities.
- Variation in the speckle pattern along the myocardium.
- Presence of papillary muscles with intensities similar to the myocardium.
- Significant tissue echogenicity variability within the population.
- Shape, intensity, and motion variability across patients and pathologies.

All these difficulties led echocardiography to search for new techniques for accurately segmenting the ultrasound images and sequences. This is when **deep learning techniques** take action. Figure 14 shows several studies related to Deep Learning techniques in echocardiogram segmentation that have been published until 2019. We also have listed the most recent papers on CNN's focused on echocardiogram segmentation from 2019 to 2022 (see Table 3).

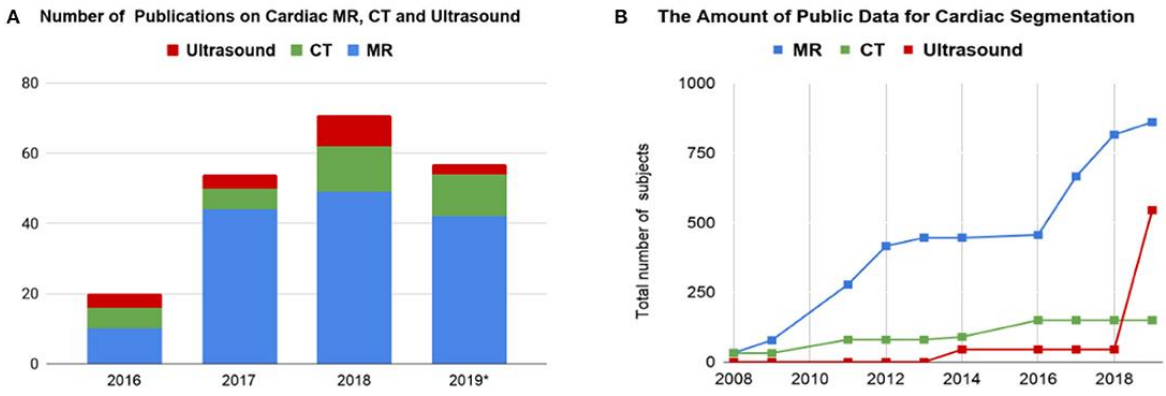


Figure 14. Overview of numbers of papers in the last decade regarding deep learning-based methods for cardiac image segmentation. CT, computed tomography; MR, magnetic resonance.
Source: https://www.frontiersin.org/articles/10.3389/fcvm.2020.00025/full

As can be seen in Figure 14, the number of published articles on cardiac segmentation has been increasing over the past years. The amount of ultrasound public data for cardiac segmentation has been the lowest over the last years until the CAMUS dataset arrived in 2019, as will be seen on the next page.

In Table 3 below, the most recent articles on echocardiogram segmentation are listed. The datasets used are CAMUS and EchoNet public datasets and other private datasets. The Region of Interest is mainly focused on the Left Ventricle, and the architectures used are mainly CNN-based architectures.

Reference	Year	Method	Region of Interest (ROI)	Data
[16]	2019	U-Net 1, U-Net 2, ACNN, SHG, U-Net ++	Left Ventricle	CAMUS dataset
[17]	2018	U-Net	Left Atrium, Right Atrium, Left Ventricle, Right Ventricle	Private dataset of Radiological Society of North America Clinical Trials
[18]	2020	VGG-based FCN	Left Atrium, Right Atrium, Left Ventricle, Right Ventricle	Dataset from the Loma Linda University Medical Center
[25]	2021	U-Net	Left Ventricle, Left Atrium, and LV pericardium	CAMUS, EchoNet and private Hospital Sant Pau dataset
[35]	2021	MTC-Net	Left Ventricle myocardium	Private dataset from Xijing Hypertrophic Cardiomyopathy Center
[36]	2020	DW-Net	Left Atrium, Right Atrium, Left Ventricle, Right Ventricle, Epicardium, descending aorta, and the thorax.	A private dataset of the echocardiography department in Anzhen Hospital.
[37]	2019	MFP-Unet	Left Ventricle	CAMUS and private dataset from Rajaie Cardiovascular Medical and Research Center
[38]	2021	PLANet	Left Ventricle myocardium	CAMUS and EchoNet datasets

Table 3. Recent articles on Convolutional Neural Networks applied to echocardiogram segmentation.

However, if we look at all Deep Learning-based recent articles for cardiac segmentation, the majority only focus on the Left Ventricle area [24]. There are only 3 studies that pursue four-chamber segmentation [17][18][36]. In section 1.1.3 the importance of not forgetting the other chambers than the LV has been established, so in our work, we will focus on the **segmentation of the four chambers** of the heart.

One of the major difficulties in automatic echocardiography segmentation is obtaining enough labeled data to train, validate and test the neural network. As we will see in section 2.1, there are **very few public datasets** available for training and validating deep learning architectures due to patients' privacy and lack of law agreement on this subject [16]. In this project, we will focus on solving this issue by searching for very recent public data.

Then, the next difficulty when working in echocardiogram segmentation is choosing which deep learning technique to work with. In our work, as explained in section 1.2.2 we will work with CNN as this type of neural network is claimed to achieve good results in echocardiogram segmentation [24], but the specific **architecture** still has to be chosen.

In Table 3**,** the architectures used in recent articles are shown. Most of them are modifications of **U-net**, which is a popular state-of-the-art encoder-decoder convolutional neural network claimed to be pretty accurate in segmenting the heart chambers [19]. These U-Net modifications are based on augmenting or combining U-Net architecture to achieve better accuracies at automatic segmentations.

However, the training of neural networks is time-consuming and architecture-dependent [39], so in this project, we will try to **optimize** the architecture and its training hyperparameters to reduce the computation time.

1.3. Objectives and Planning

Once all the introductory part has been explained, the purpose of this work and the objectives are specified in this section. Given the explained variability in echocardiogram segmentation and the necessity to assess all parts of the heart, the purpose of this work is:

- **Purpose**: Provide an optimized and accurate Deep Learning architecture to automatically segment the four chambers of the heart.

To achieve this main purpose, the following steps or **objectives** have to be achieved:

1. **Obtain** echocardiographic labeled data of the four chambers of the heart.
2. **Choose** an accurate Convolutional Neural Network to train with the obtained data.
3. **Optimize** the chosen CNN architecture and parameters and analyze the results.

This project takes place within the Bachelor's degree in Industrial Technology Engineering and weights 12 ECTS credits. The duration of this work is from February 2022 until June 2022, i.e., **5 months**.

The planning of this project, according to the thesis directors, is set in **weekly online meetings** via Google Meet where weekly objectives for the next week are established and the previous week's objectives are revised. This type of planning allows the project development to adapt to the variability of the data acquisition and deep learning techniques implementation. Weekly objectives are registered in a Thesis Tracking document.

The resources planned for this project are only obtaining a **capable computer** environment to implement Neural Network techniques with Matlab. No more economic resources are planned to be necessary.

2. Methodology

In this section, the technical methods used to achieve the purpose are explained. The data acquisition and the deep learning architecture will be presented.

2.1. Data Acquisition

One of the biggest issues in echocardiogram segmentation is **the lack of available public data** to test the automatic segmentation methods. If the data used to train the architecture is not enough, the architecture won't reach proper accuracy values. In addition, another problem presented is over-fitting, explained in section 2.1.4. To prevent training under-performance and over-fitting (i.e., having a trained network that only performs well in the training dataset but not in other datasets) a big heterogeneous dataset would be the solution.

2.1.1. Publicly Available datasets

After looking for open-access 2D heart ultrasound images, the following datasets were found (see Figure 15):

- **CAMUS dataset** [16]: Dataset composed of **450 patients**' echocardiograms acquired at the University Hospital of St Etienne (France). For each patient, four echocardiogram images (2-Chamber End Diastole and End Systole and 4-Chamber ED and ES) with their respective **ground-truth data** from **the Left Ventricle, Left Atrium,** and **Left Ventricle's Myocardium** are given, and two sequences of a cardiac cycle of the heart for each patient. In addition, a text file describing sex, age, image quality, and left ventricle volumes and ejection fraction for each patient is given.

- **EchoNet-Dynamic dataset** [20]: Dataset composed of **10,030** apical-4-chamber echocardiography **videos** from individuals who underwent imaging between 2016 and 2018 at Stanford University Hospital. In addition to the video itself, each study is linked to clinical measurements and calculations (ejection fraction and LV volumes) done by experts. **No masks/ground-truth data** are given in this dataset.

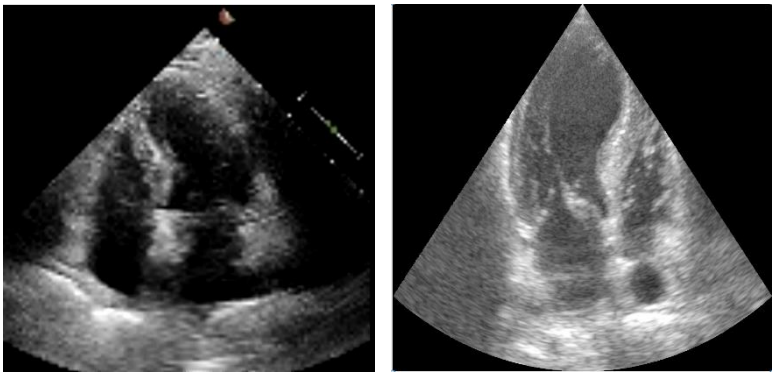


Figure 15. Left, EchoNet-Dynamic dataset image. Right, CAMUS dataset image.
Source: EchoNet-Dynamic; CAMUS

- **Echogan Dataset** [18]: Anonymized Multi-chamber Echocardiograms Database (AMED) is an anonymized dataset of **1395** annotated images from 100 patients from Loma Linda University Medical Centre. The study population included 73 female and 27 male subjects, with a mean age of 36.5± 13.5, who underwent clinically indicated standard transthoracic echocardiography from 2014 to 2015. To address the existing challenges for the analysis of low-quality images, no clinically interpretable study was excluded. The study protocol was approved by the institutional review board of Loma Linda University. The studied images were apical four-chamber views without any contrast agent. Manual segmentations were performed by a cardiologist for one in every five frames of the cardiac cycle, in addition to the end-diastolic (ED) and end-systolic (ES) images for each subject.

2.1.2. Image Processing Software

In the first dataset search performed, only CAMUS and EchoNet datasets were found, as they were published before the Echogan dataset and more used in all the reference articles found (see Table 3). However, none of them had ground-truth data for all the four chambers of the heart. Only CAMUS had Left Ventricle masks. Given the necessity to acquire **four-chamber ground-truth segmentations**, different medical segmentation programs were evaluated to obtain ground-truth data from EchoNet and CAMUS datasets. Three programmes were used: **Ilsatik** [21], **Seg3D** [22] and **3D Slicer** [23].

· Ilastik

The first program used was Ilastik. To create ground-truth masks, different **filters** were applied to the imported data, and then labeling was performed for automatic segmentation. **The batch processing** option was used to apply determined filters to a full batch of images for automatically segmenting a big number of images.

Filtering the image acts equally to all parts of it, so an option for locating those filters to the chambers had to be found for segmenting only the chambers. This option was the **labeling** stage, where the relevant parts are labeled to indicate to the program where the chambers are expected to be. Despite this, the results showed that the program needed a full-labeling process, i.e., labeling the whole image to acquire good segmentations:

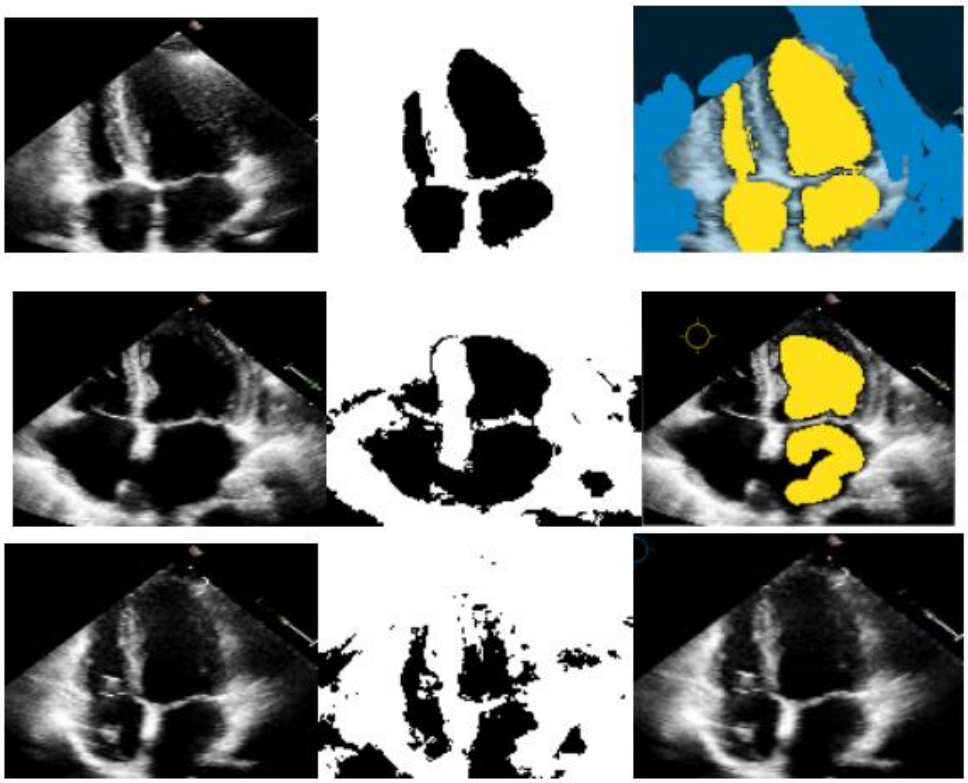


Figure 16. Original images, results, and their respective manual labels.

1st row: fully-manually-labeled image

2nd row: partially-manually-labeled image

3rd row: non-labeled image

Source: Ilastik

In Figure 16 above, the first row corresponds to an almost fully-manually-labeled image, where the results are visually pretty well-segmented (center image). In the second row, only the left ventricle and atriums are labeled, and thus the program segments also regions of the outer heart that are similar to the interior of the chambers.

The last row corresponds to a non-labeled image, where only filtering and automatic segmentation of the program are applied. The resulting ground-truth of this non-labeled image is the 3rd-row center image, where the chambers are visually poorly segmented and the outer area of the heart is also segmented as if it was a chamber. Therefore, it can be seen that the program **didn't achieve good** results when automatically segmenting the four chambers without manually labeling.

Neither did when the number of manually labeled images increased. The results above were obtained by labeling 5 images in a group of 20 images (see Figure 17).

Figure 17. Automatic segmentations in a batch with 20 manual labeled images.

Source: File explorer.

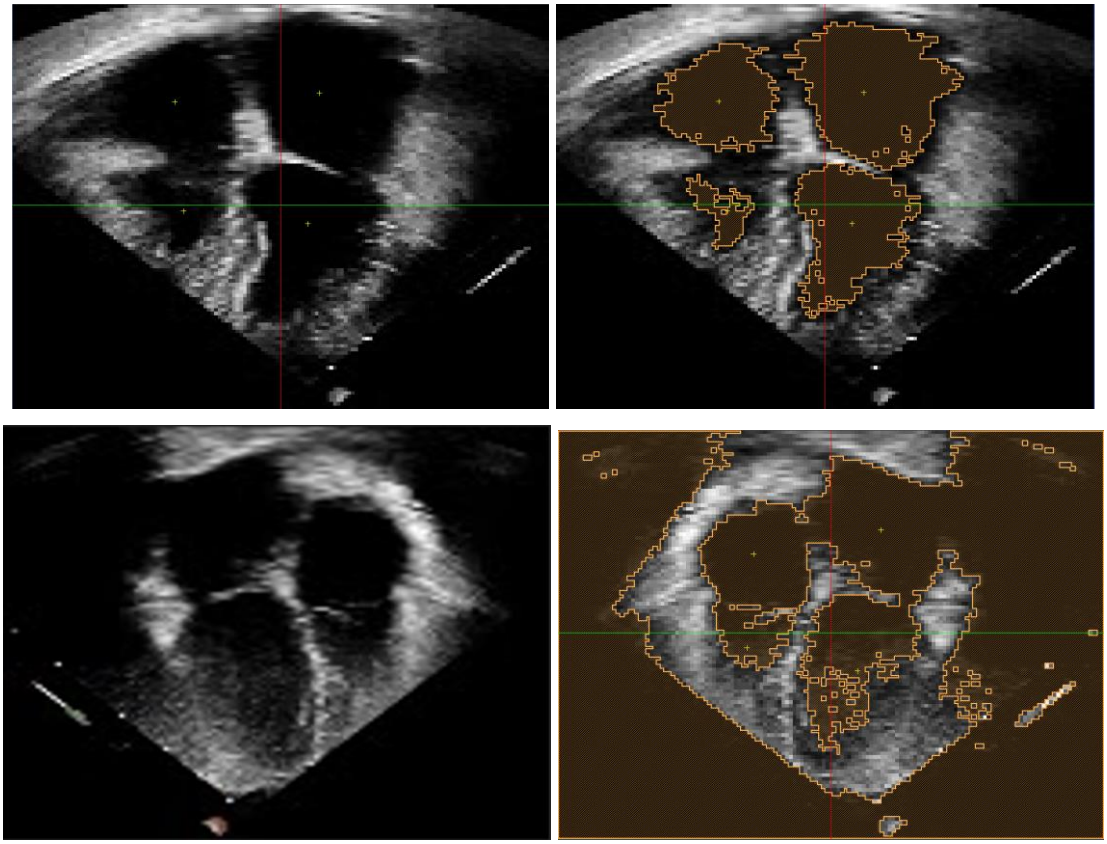
When the number of labeled images was increased up to 200 images, the automatic results remained poor. Therefore, increasing the number of labeled images was not the solution.

.

· 3D Slicer and Seg 3D

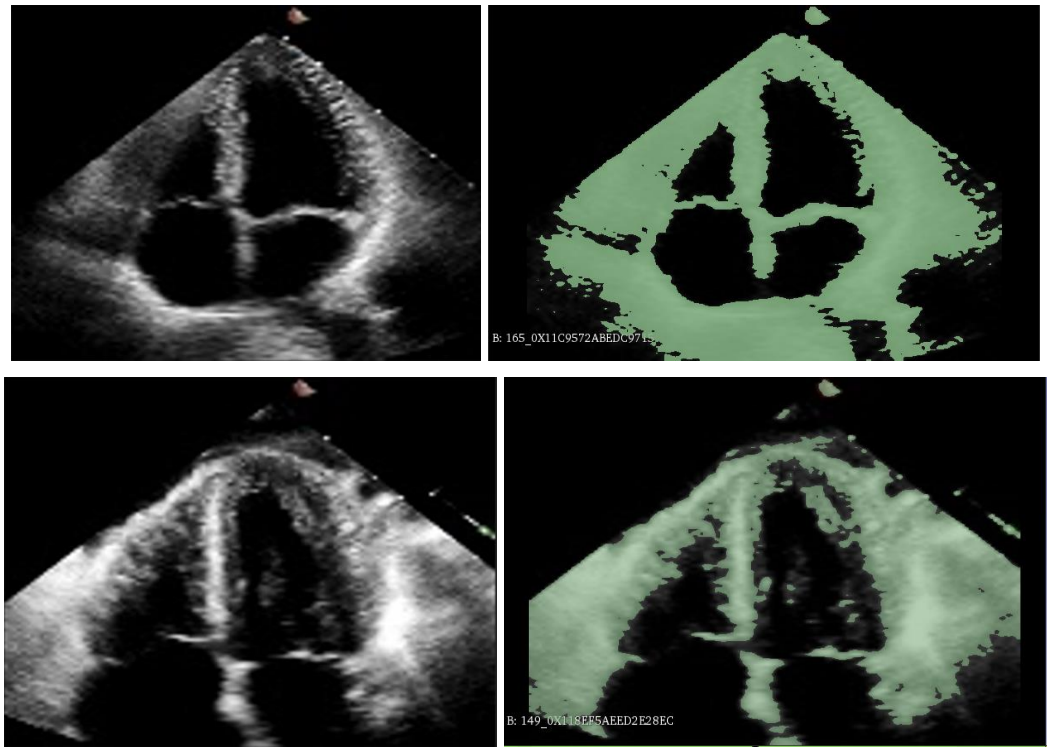
To ensure that the problem found in Ilastik wasn't only related to this particular software, two other image processing environments were evaluated. 3D Slicer and Seg3D worked pretty similar to Ilastik: **applying filters** to segment the images.

One useful filter found in seg3D was **Confidence Connected**, where **seeds** were manually located inside the chambers to tell the program where these chambers are (see Figures 18). This procedure was much easier and faster than labeling the whole image as in Ilastik, but results weren't much better in general, only when four chambers were totally "closed":



Figures 18. Proper segmentations and poor segmentations with Confidence Connected filtering.
Source: Seg 3D

In 3D Slicer, **thresholding** was also found to be useful to solve the poor-quality issue that happened with Ilastik, but the problem of having to manually label the chambers remained (see Figures 19).

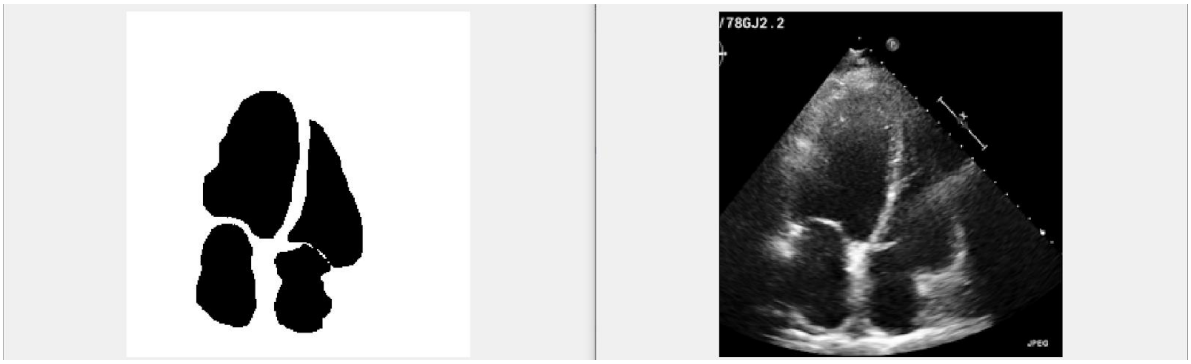


Figures 19. Results with 3D Slicer.

Source: 3D Slicer

2.1.3. Echogan dataset

After testing the medical image processing software and obtaining inaccurate results for our particular case (see Figure 16. Figures 18, and Figures 19), another research on public datasets was carried out. We found the article [18] where a brand-new dataset from the Loma Linda University Medical Center described in section 2.1.1 was proposed and available to download:



Figures 20. Echogan dataset.
Source: https://bitbucket.org/Aarafati/echogan/src/master/

This dataset had all four chambers with their ground-truth masks attached (see Figures 20) and had considerably better quality than the other datasets.

2.1.4. Data Augmentation and Preparation

As most articles state [24], preventing and reducing over-fitting is one of the biggest challenges in neural network training. **Over-fitting** happens when the trained neural network only achieves good results in the training dataset but not in other datasets and environments. To try to generalize the neural network learning to multiple and different environments, experts use different techniques.

Data augmentation is a training strategy that artificially generates more training samples to increase the diversity of the training data. This can be done via applying affine transformations (e.g., rotation, scaling), flipping, or cropping to original labeled samples.

In this study, we applied the following data augmentation variations to our dataset: horizontal and vertical flips, random rotation with an angle α between $\alpha \subseteq [-20^{o}; 20^{o}]$, random zooming

with a factor x between x ⊆ [x1; x1,25], adding Gaussian noise and random brightness augmentation. This way, we obtained an augmented dataset of 9765 images, i.e., the original dataset multiplied by a factor of 7 (see Figure 21).

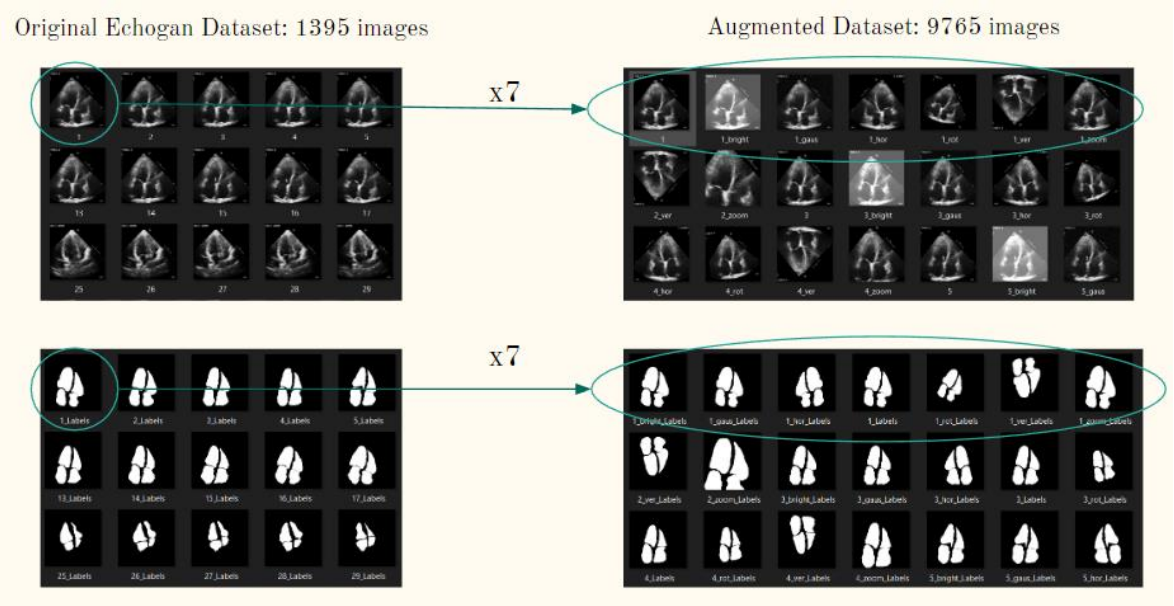


Figure 21. Data augmentation. Left, original Echogan dataset; right, augmented dataset.

Source: Powerpoint

All these techniques were executed using MATLAB R2022a. The augmented dataset was split into three sub-sets: 6251 images for **Training** (64%), 1561 for **Validation** (16%), and 1953 for **Testing** (20%). Thus, the 1st objective of the work (obtaining labeled data of the four chambers) was achieved.

2.2. Deep Learning Architecture

As explained in section 1.2.3, Convolutional Neural Networks are recently shown to be accurate in the task of echocardiogram segmentation [16] [25]. In this section, we will present the chosen architecture for this work and the optimizations that will be applied to it.

2.2.1. U-Net

Given the limited duration of the project, the architecture had to be relatively suitable to be understood and optimized. In Table 3, we saw that the major part of recent articles worked with U-Net-based architectures. Therefore, we chose to work with **U-Net**, proposed by Ronneberger et. al. [19], which is popularly used for echocardiogram segmentation.

Its architecture, shown in Figure 22 below, is symmetrical. This makes optimization an easier task as the two symmetrical parts are complementary to each other and modifications have to be applied to both parts at the same time.

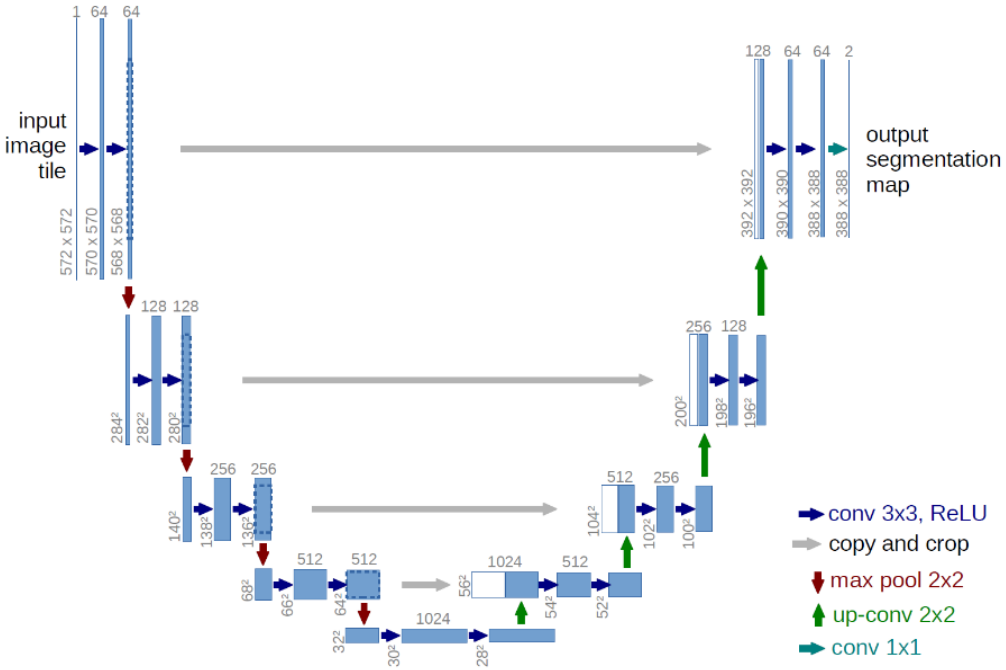


Figure 22. U-net architecture. Each blue box corresponds to a multi-channel feature map with dimensions and channels annotated. The arrows denote the different operations.

Source: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

This network contains multiple convolutions and max-pooling operations, explained in section

1.2.2. The activation function of this network is **Rectifier Linear Unit** (ReLU) (see Figure 23):

$$ReLU(x) = \max(0, x)$$

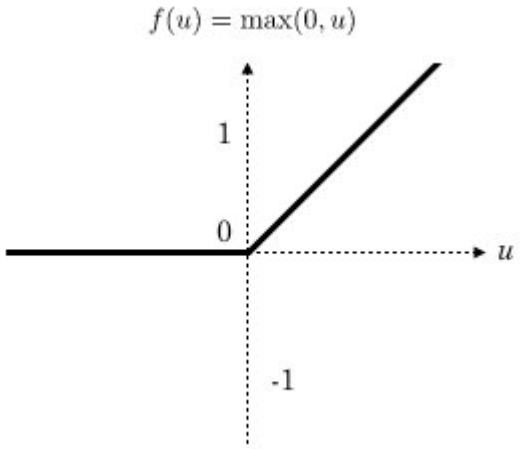


Figure 23. ReLU function.

Source: Pauly, Leo & Peel, Harriet & Luo, Shan & Hogg, David & Fuentes, Raul. (2017). Deeper Networks for Pavement Crack Detection. 10.22260/ISARC2017/0066.

On the other hand, the loss function for our U-Net implementation is **cross-entropy**. To understand this loss function's nature, let's explain the output layers: the softmax layer and the classification layer (loss function) [33].

A **softmax layer** applies a softmax function to the input. A classification layer computes the cross-entropy loss for classification and weighted classification tasks with mutually exclusive classes. For classification problems, a softmax layer and then a classification layer usually follow the final fully connected layer.

The output unit activation function is the **softmax function**:

$$y_r(x) = \frac{\exp\left(a_r(x)\right)}{\sum_{j=i}^{k} \exp\left(a_j\left(x\right)\right)},$$

Where $0 \leq y_r \leq 1$ and $\sum_{j=1}^{k} y_j = 1$

In the **classification layer**, the trainNetwork function takes the values from the softmax function and assigns each input to one of the K mutually exclusive classes using the **cross-entropy function** for a 1-of-K coding scheme:

$$loss = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} w_i t_{ni} \ln\, y_{ni}$$

where N is the number of samples, K is the number of classes, w_i is the weight for class i, t_{ni} is the indicator that the n^{th} sample belongs to the i^{th} class, and y_{ni} is the output for sample n for class i, which in this case, is the value from the softmax function. In other words, y_{ni} is the probability that the network associates the nth input with class i.

In this U-Net implementation, weights are initialized with the **He weight initialization** method [34]. He Initialization is an initialization method for neural networks that takes into account the non-linearity of activation functions, such as ReLU activations. It consists of initializing weights with random values coming from a zero-mean Gaussian distribution whose standard deviation is $\sqrt{2/n_l}$ This is our way of initialization. Biases are initialized at 0.

The network architecture consists of a **contracting path** (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions, each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step, we double the number of feature channels (number above blue boxes).

Every step in the **expansive path** consists of an upsampling of the feature map followed by a 2x2 convolution (“up-convolution”) that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23

convolutional layers. This way, the 2nd objective of the work was achieved.

The implementation of U-Net in this work was carried out in **MATLAB R2022a** for Windows 10 with Computer Vision, Deep Learning, and Parallel Computing toolboxes.

U-Net layers were created by function **unetLayers** [26]. In our case, the image size was 256x256 and there were 5 classes with respective values 0, 1, 2, 3, and 4: background, Left Ventricle, Right Ventricle, Left Atrium, and Right Atrium. The neural network training was performed using the function **trainNetwork** [27]. Within these options, multiple training hyperparameters in section 2.2.2 can be determined.

The environment used to carry out all the experiments was Intel(R) Core (TM) i7-6800K CPU @ 3.40GHz with 32 GB RAM and NVIDIA TITAN Xp GPU with Computing Capability 6.1

2.2.2. Hyperparameter Optimization

Once the U-Net architecture was implemented, the following training hyperparameters were tested to see how the neural network could be optimized These parameters were chosen given their importance on the network training's performance [29].

- **Optimizer** (explained in section 1.2.1): In training options, the available optimizers were stochastic gradient descent with momentum (sgdm), the rate of the squared gradient moving average (rmsprop), and Adam optimizer:

    - The **stochastic gradient descent** algorithm can oscillate along the path of steepest descent towards the optimum. Adding a momentum term to the parameter update is one way to reduce this oscillation. The stochastic gradient descent with momentum (SGDM) update is

    $$\theta_{\varphi+1} = \theta_{\varphi} - \alpha\nabla E(\theta_{\varphi}) + \gamma(\theta_{\varphi} - \theta_{\varphi-1})$$

    where γ determines the contribution of the previous gradient step to the current iteration.

    - The **RMSProp** (root mean square propagation) seeks to improve network training by using learning rates that differ by parameter and can automatically

adapt to the loss function being optimized. It keeps a moving average of the element-wise squares of the parameter gradients,

$$v_\ell = \beta_2 v_{\ell\_1} + (1-\beta_2)[\nabla E(\theta_\ell)]^2$$

- Finally, **Adam** (derived from adaptive moment estimation) uses a parameter update that is similar to RMSProp, but with an added momentum term. It keeps an element-wise moving average of both the parameter gradients and their squared values,

$$m_\ell = \beta_1 m_{\ell\_1} + (1-\beta_1)\nabla E(\theta_\ell)$$

$$v_\ell = \beta_2 v_{\ell\_1} + (1-\beta_2)[\nabla E(\theta_\ell)]^2$$

- The next hyperparameter to be optimized is the **Initial learning rate** used for training, specified as a positive scalar. The default value is 0.01 for the 'sgdm' solver and 0.001 for the 'rmsprop' and 'adam' solvers. If the learning rate is too low, then training can take a long time. If the learning rate is too high, then training might reach a suboptimal result or diverge.

- Finally, two values for **Mini-Batch Size** were tested. This is the size of the mini-batch to use for each training iteration, specified as a positive integer. A mini-batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights.

One training on the full augmented dataset was carried out for each combination of the hyperparameters mentioned above. This was done by taking advantage of Experiment Manager in MATLAB [28].

The **other training parameters** were fixed at the following values [27]:

- Shuffle = “every-epoch” Shuffle the training data before each training epoch, and shuffle the validation data before each network validation.

- MaxEpochs = 10 Maximum number of epochs to use for training, specified as a positive integer. An iteration is one step taken in the gradient descent algorithm towards minimizing the loss function using a mini-batch. An epoch is the full pass of the training algorithm over the entire training set.

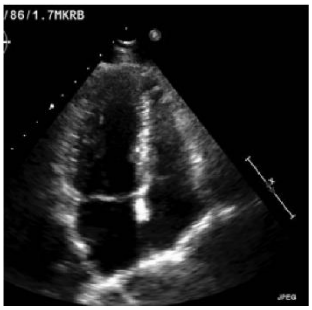
- Validationdata = “dsval” Data to use for validation during training
- ValidationFrequency = 10 Frequency of network validation in number of iterations, specified as a positive integer.
- Plots = “training-progress” The plot shows mini-batch loss and accuracy, validation loss and accuracy, and additional information on the training progress.
- OutputNetwork = “best-validation loss” Return the network corresponding to the training iteration with the lowest validation loss.

The results of this experiment are shown in **Table 4**.

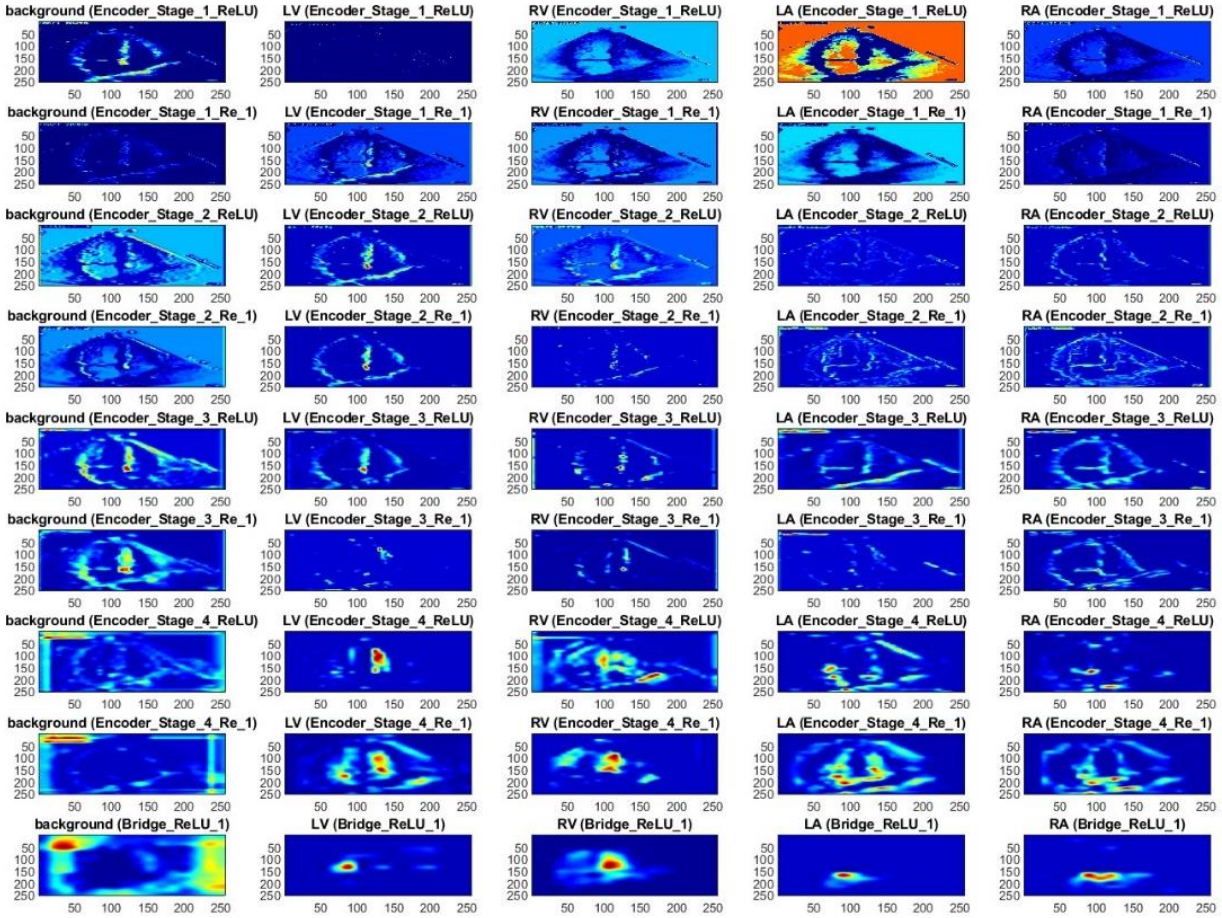
2.2.3. Architecture Optimization

Once the best hyperparameter combination was obtained, the following optimization was performed on the **U-Net architecture** itself. As seen in the first results in section 3.1, the training of the architecture took a lot of computation time and effort. Therefore, optimizing the architecture to reduce this computational time and still being able to achieve good results would be a major advance.

For this purpose, **gradient-weighted class activation mapping** (Grad-CAM) [32] methods were used to analyze the architecture and decide which layers could be removed or replaced. This technique was invented to understand why a deep learning network makes its classification decisions. Grad-CAM uses the gradient of the classification score related to the convolutional features determined by the network to understand which parts of the image are more important for classification (see Figures 24, 25, and 26 below). Taking advantage of Matlab's in-built function **gradCam** all activation maps were visualized for each ReLU layer (activation function) to see which part of the images the architecture was focusing on:

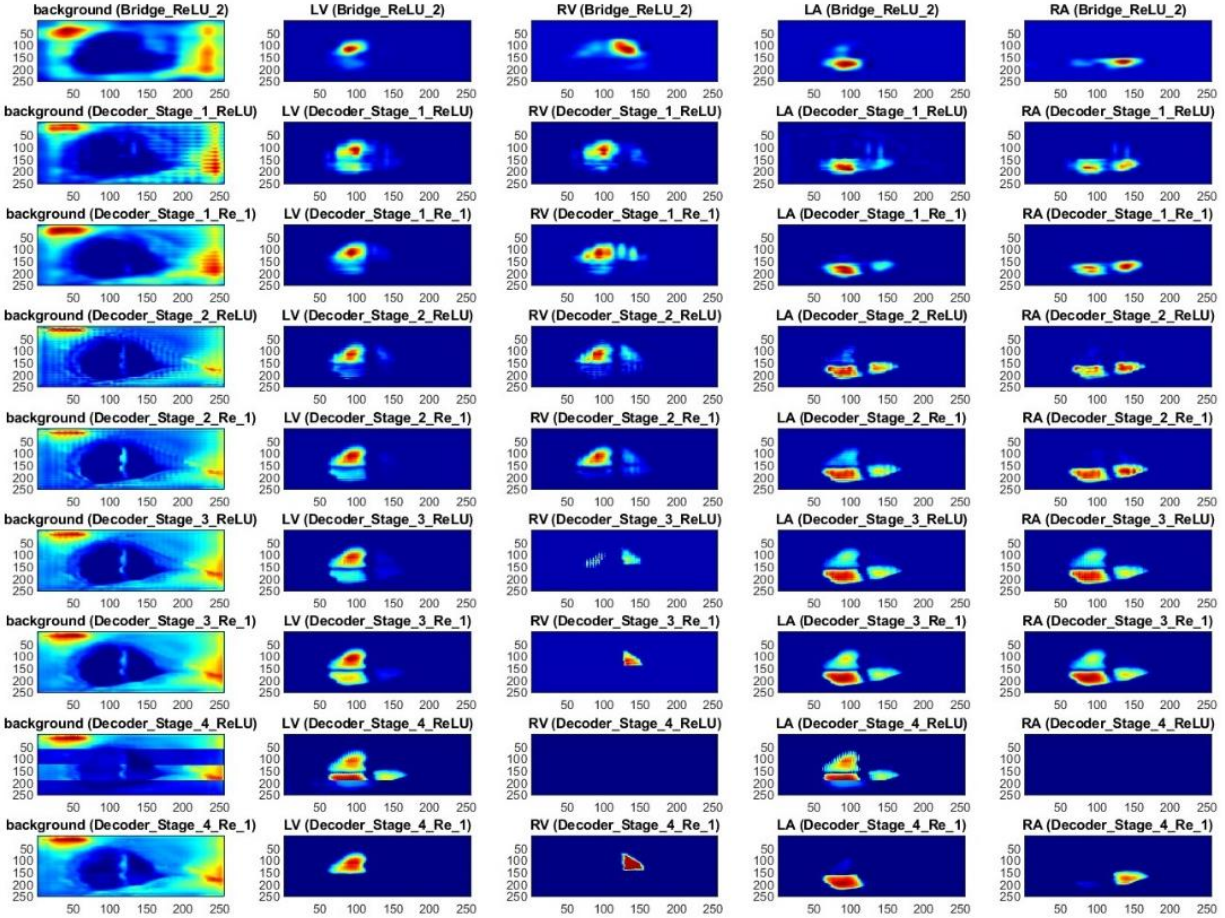


Figures 24. Test image above
and activation CAM heat maps for each layer (rows) and class in the next two pages
(columns: background, Left Ventricle [LV], Right Ventricle [RV, Left Atrium [LA], Right Atrium [RA]).
Source: Matlab



Figures 25. Activation CAM heat maps in the Encoder for each layer (rows) and for each class (columns: background, Left Ventricle [LV], Right Ventricle [RV, Left Atrium [LA], Right Atrium [RA]).

Source: Matlab



Figures 26. Activation CAM heat maps in the Decoder for each layer (rows) and for each class (columns: background, Left Ventricle [LV], Right Ventricle [RV, Left Atrium [LA], Right Atrium [RA]).

Source: Matlab

As can be seen in Figures 25 and Figures 26, **later layers** (the ones in the decoder) within the architecture produced maps very similar to the classified outcome, i.e., the predicted segmentation. However, **earlier layers** (encoder) in the network produced more abstract results that are typically more related to **lower-level features** like edges, with less awareness of semantic classes.

In maps at encoder layers, background and chambers classes were almost no different from each other apart from the contours of the heart. This fact suggested that earlier layers focused on **common basic features** of all five classes like edges and colors. The network started classifying the background and the chambers at bridge layers. Until the last layer of the architecture, it didn't completely differentiate all the chambers.

After analyzing all layers within the architecture and seeing that there was a significant difference between early and later layers in the network, the architecture modification applied was modifying the **Encoder-Decoder depth**. As explained in section 2.2.1, U-Net is composed of an **encoder** subnetwork and a corresponding **decoder** subnetwork (see Figure 22). The depth of these subnetworks determines the number of times the input image is downsampled or upsampled during processing. The encoder network downsamples the input image by a factor of 2, where D is the value of EncoderDepth. The decoder network upsamples the encoder network output by a factor of 2D (see Figure 27).

Therefore, the **Encoder-Decoder depth** is the number of bridge layers between the Encoder and the Decoder:

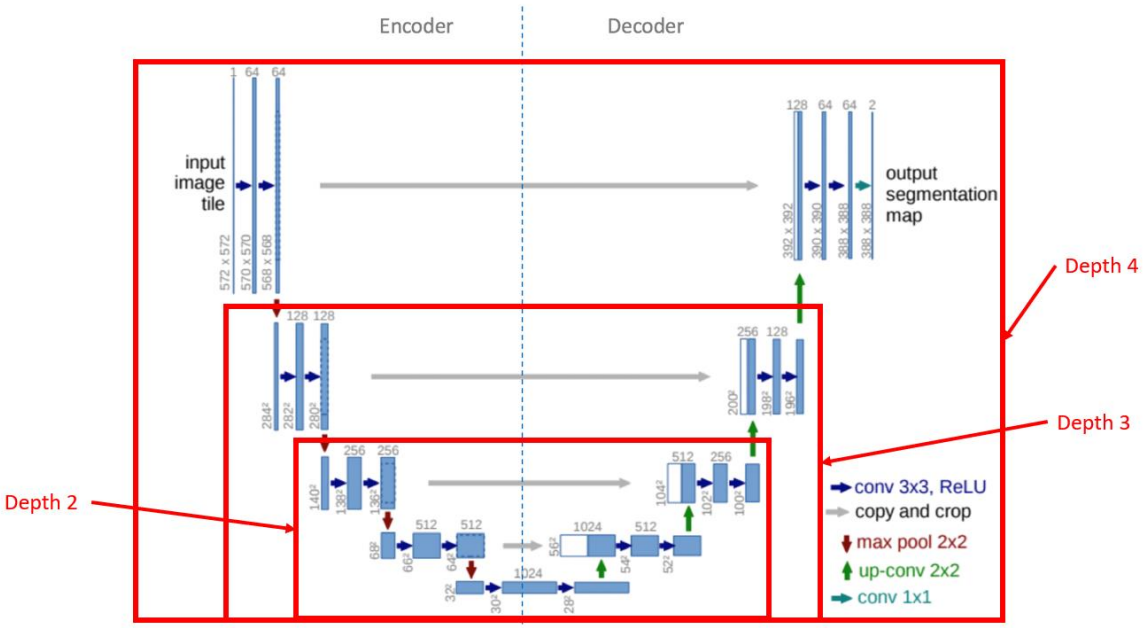


Figure 27. U-Net networks with different Encoder-Decoder depth.

Source: Powerpoint

U-Net default Encoder-Decoder value is 4. To analyze how optimal this value is, U-Nets with Encoder-Decoder depths of **2, 3, 4, and 5** were tested with the best hyperparameter combination obtained in the previous optimization. The Initial Learning Rate was set at 0,0001 after showing less computation time than with 0,001 (see section 4.2). The computation time and test accuracy were the responses for this experiment. Results are shown in section 3.3.

3. Results

In this section the results of each optimization explained in the methodology section will be presented. Results will be grouped by tables to summarize the experiments done, and training accuracies will be represented in plots.

3.1. Hyperparameter Optimization

As explained in section 2.2.2, training trials were executed for each **hyperparameter combination**. For each trial, we calculated the training and validation accuracy and loss, respectively. Results from the first hyperparameter optimization are shown in Table 4:

TRIAL	ELAPSED TIME (h)	INITIAL LEARN RATE	SOLVER	MINI BATCH SIZE	TRAINING ACCURACY (%)	TRAINING LOSS	VALIDATION ACCURACY (%)	VALIDATION LOSS
1	13:02:53	0,01	adam	8	80,55	0,48	79,16	0,50
2	08:28:26	0,01	adam	32	82,31	0,47	79,16	0,51
3*	00:11:10	0,01	sgdm	8	79,17	2,97	20,84	12,62
4	08:23:32	0,01	sgdm	32	79,10	0,51	79,16	0,51
5	16:15:41	0,01	rmsprop	8	19,90	12,77	20,84	12,62
6	08:20:24	0,01	rmsprop	32	17,92	13,09	20,88	12,61
7	16:20:31	0,001	adam	8	96,74	0,08	96,25	0,09
8	09:52:35	0,001	adam	32	94,87	0,13	95,30	0,12
9	13:09:06	0,001	sgdm	8	90,41	0,21	87,49	0,28
10	08:34:00	0,001	sgdm	32	89,28	0,25	86,37	0,32
11	13:26:03	0,001	rmsprop	8	95,07	0,12	95,45	0,11
12	08:36:27	0,001	rmsprop	32	94,20	0,15	93,32	0,19

Table 4. Results from hyperparameter optimization.

Source: Matlab

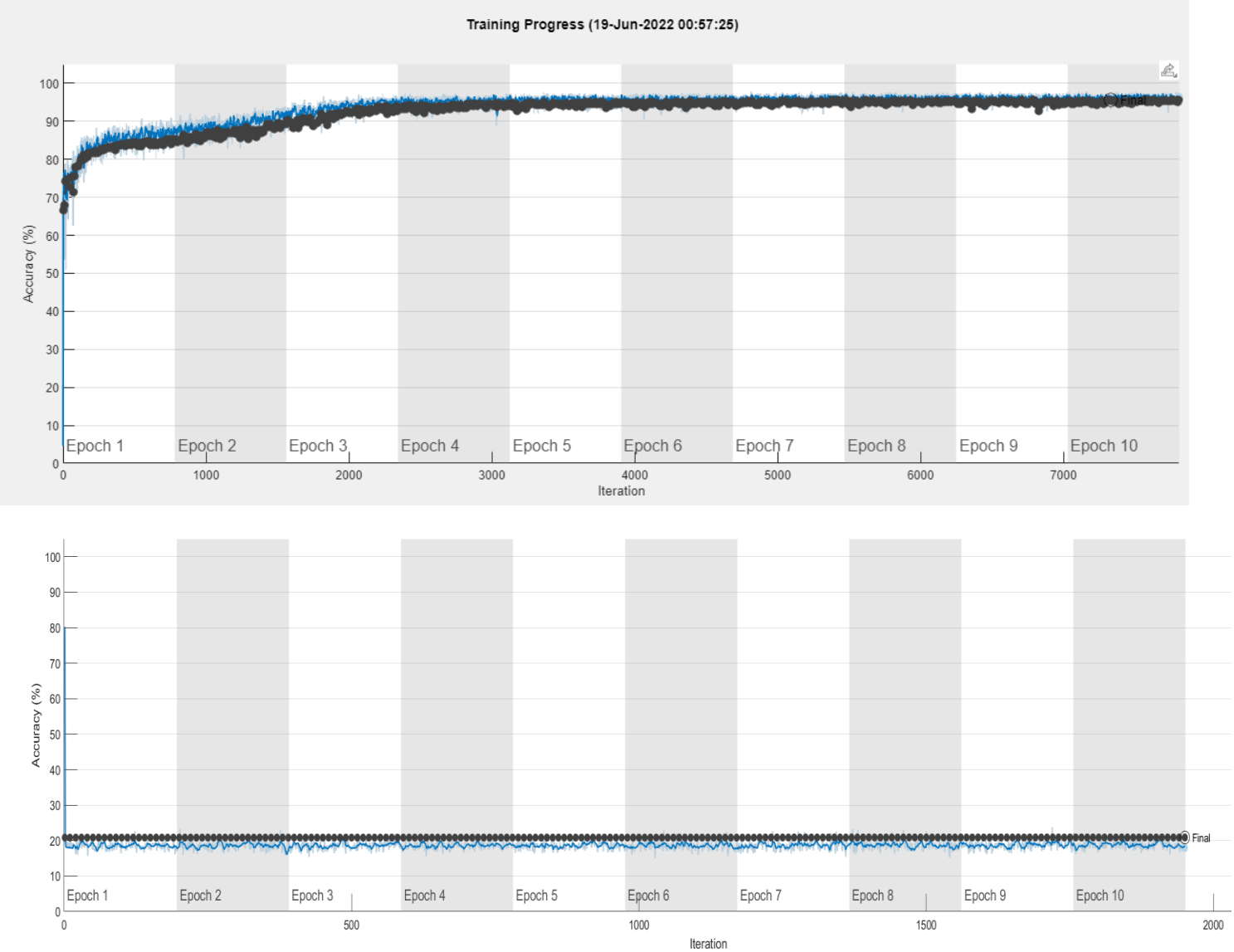
Where training and validation **accuracy** is measured by first predicting the mini-batch images with and the validation set images respectively, and then calculating the logical comparative mean of these predicted masks with their respective ground-truth:

$$Accuracy = \frac{\sum_{i=1}^{n}(\overline{Mask_{predicted\_i} == Mask_{ground-truth\_i}})}{n}$$

Training and validation **losses** are the loss function (explained in section 2.2.1) mean values on each mini-batch size and validation set.

As can be seen in the table above, the best hyperparameter combination for training the neural network was **Mini Batch Size of 8, Adam solver, and 0,001 of Initial Learn Rate**. The worst trial with a lower Initial Learn Rate value, the root mean square optimizer, and high Mini Batch Size. To see which range of accuracies was covered between these trials, the training plot of each case was obtained.

This training plot was automatically generated by the Experiment Manager (see Figures 28). The plot shows the **training accuracy** (blue line) calculated with the mini-batch, and **validation accuracy** (black points) calculated with the validation set at each validation period (see validation frequency in section 2.2.2):



Figures 28. Training (blue) and validation (black points) accuracies over the iterations for the best and worst trials.

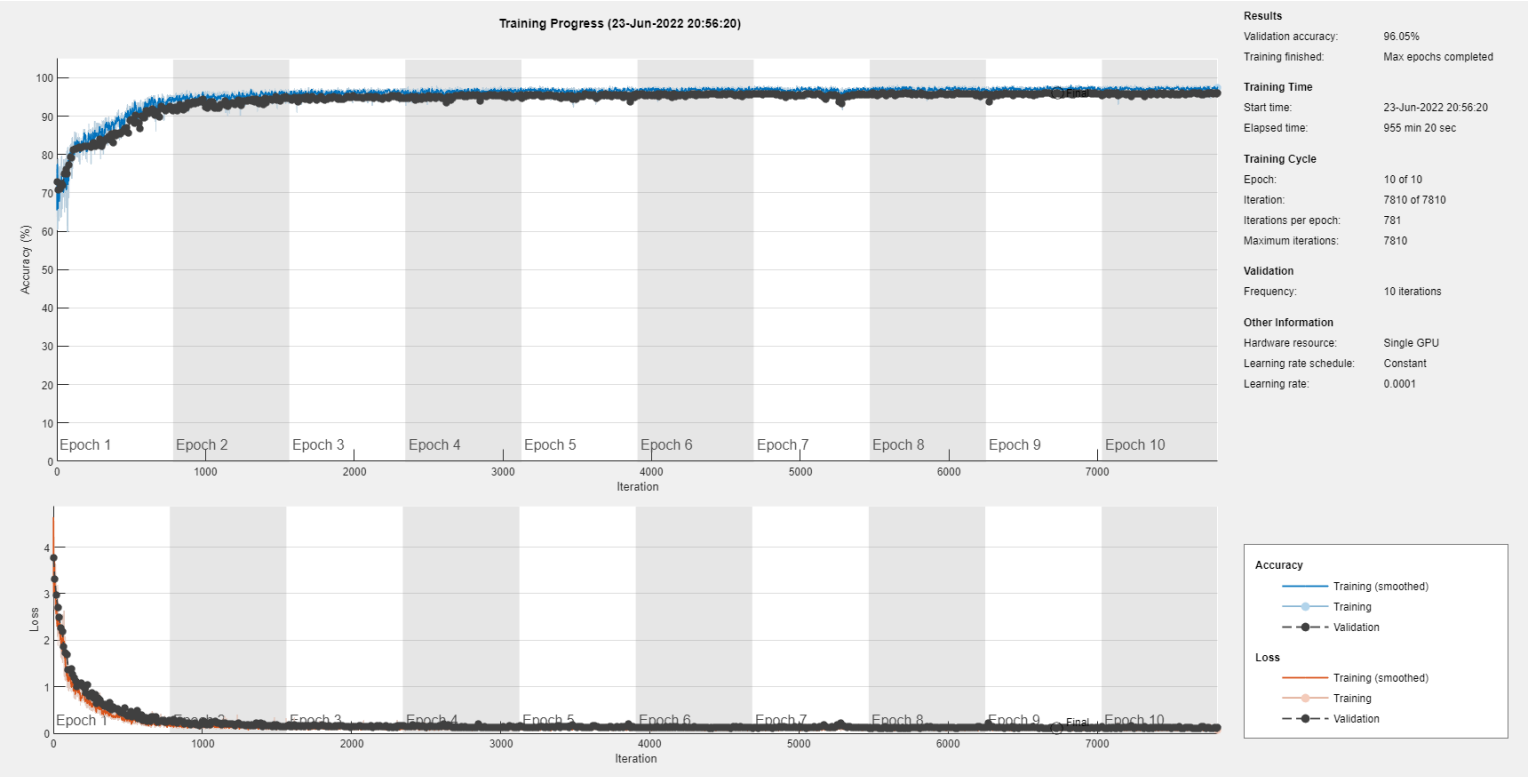
Source: Matlab

Finally, trial number 3 shown in Table 4 resulted in an **error**. The cause was that Initial Learning Rate was too low for this optimizer and the Mini Batch Size of this combination [29]. Therefore, this trial was determined to be removed from the discussion and analysis. Discussion of these results is presented in section 4.1.

3.2. Further Hyperparameter Testing

Given the importance of the Initial Learning Rate parameter explained in the previous results' discussion in section 4.1, an extra experiment was performed. This extra experiment, consisted in training the network with an **Initial Learning Rate of 0,0001** to see if a reduction in the Initial Learning Rate value would increment the computation time and training accuracy. The same hyperparameters obtained before were used but with the Initial Learning Rate set to 0,0001.

To see the performance of the training, we obtained the training plot automatically generated by the Experiment Manager on Matlab (see Figures 29) as we did before. Training accuracy (blue line) and loss (orange) and validation accuracy and loss (black points above and below, respectively) were plotted during the training:

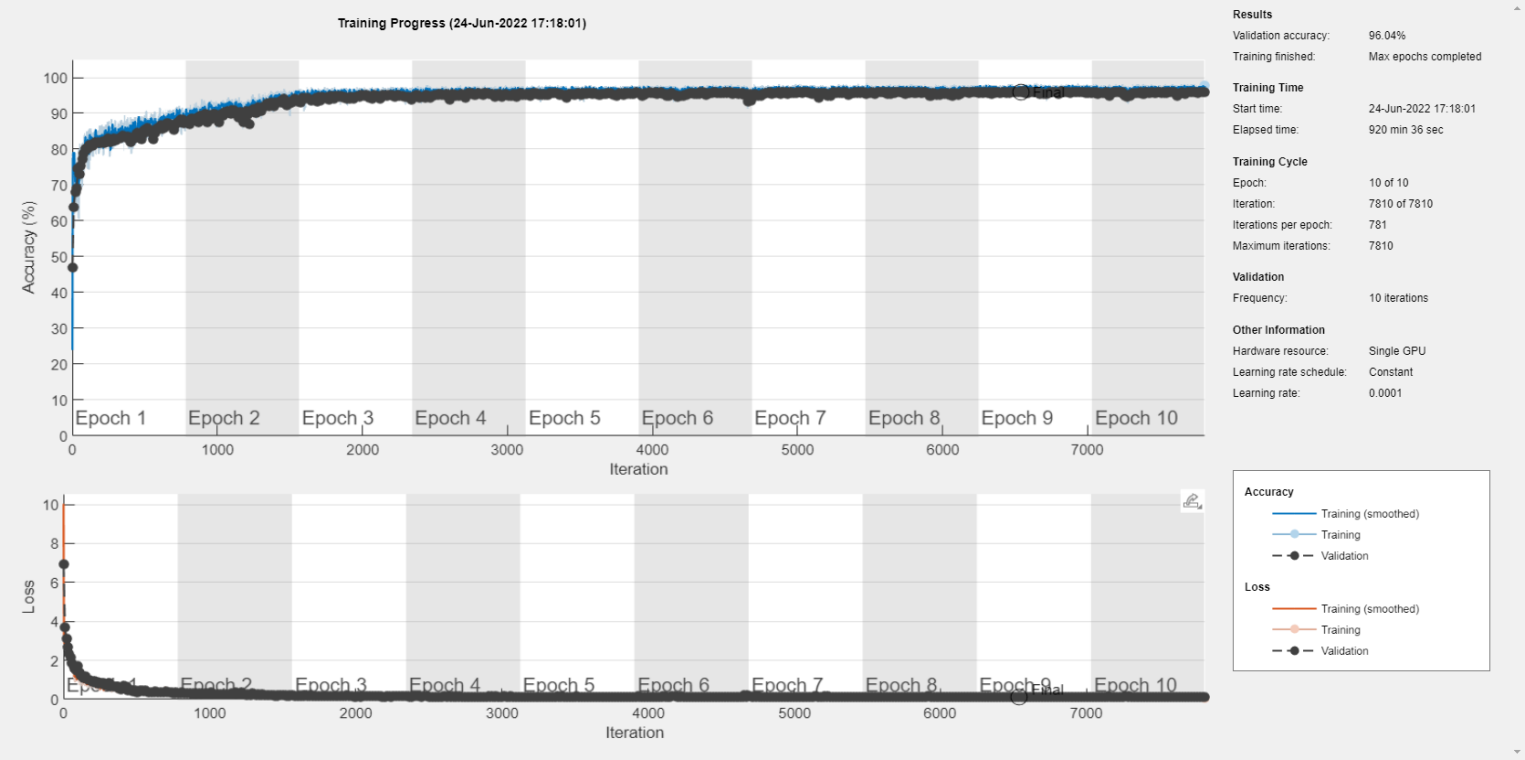


Figures 29. Training and validation accuracies and losses for ILR=0,0001.

Source: Matlab

Results showed that **neither the accuracy nor computation time seemed to vary** from the best trial of the first results. With an Initial Learning Rate of 0,001, it took 16 hours and 20

minutes, and the validation accuracy was 96,25%. Now, with an Initial Learning Rate of 0,0001, it took 16 hours and validation accuracy was 96,05%. The **same test** was repeated to see if the initialization's variability affects the results. The training plot is shown in Figures 30:



Figures 30. Training and validation accuracies and losses of repeated trials with ILR=0,0001.

Source: Matlab

Surprisingly, the training accuracy was almost the same as before but the computation time was 35 min less this time. Discussion on these results is presented in section 4.2.

3.3. Architecture Optimization and Testing

After optimizing the hyperparameters, in this section, the focus will be on the architecture. For this purpose, five training trials of U-Net on the full training subset were carried out with **Encoder-Decoder depth** values of 2, 3, 4, and 5 as explained in section 3.3. For each trial, we calculated the final training and validation accuracies and losses, just as in hyperparameter optimization. In addition, we looked at the computation time, the number of layers, and the parameters of each modified architecture. Results are summarized in Table 5:

Encoder-Decoder depth	***Elapsed Time (min)***	***Layers***	***Parameters (millions)***	***Training Accuracy (%)***	***Training Loss***	***Validation Accuracy (%)***	***Validation Loss***
2	*459*	*34*	*1,8*	*93,43*	*0,18*	*90,67*	*0,27*
3	*726*	*46*	*7,6*	*96,73*	*0,08*	*95,36*	*0,13*
4	*855*	*58*	*31*	*96,99*	*0,07*	*95,72*	*0,11*
5	*1114*	*70*	*124,3*	*95,73*	*0,10*	*96,10*	*0,10*

Table 5. Training results of the architecture modifications

Source: Matlab

No higher Encoder-Decoder depths were tested given the purpose of reducing the computation time, i.e., reducing the number of layers and therefore the depth of the architecture. Trial of depth 5 only was performed to see how the results changed when augmenting the architecture vs when reducing it. The discussion on these results is presented in section 4.3.

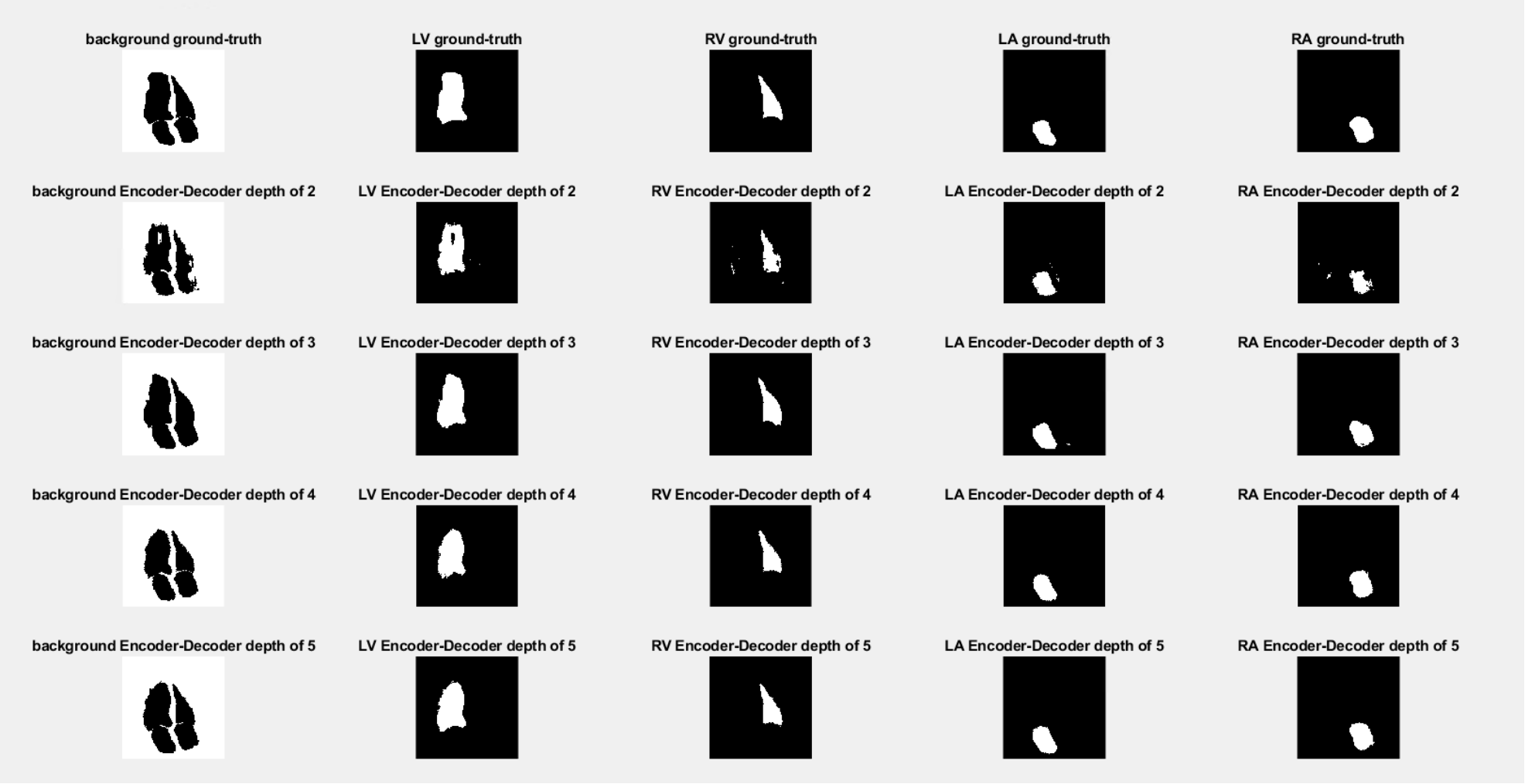
After obtaining the training results, the accuracy of the trained networks had to be analyzed. For this reason, we used the 4 architectures to make predictions at the **testing subset** defined in section 2.1.4. Testing accuracy results are shown in Table 6. The accuracy mean value was calculated for each class, comparing the networks' predictions and the ground-truth data.

Encoder-Decoder depth	***Background***	***LV***	***RV***	***LA***	***RA***	***Average***
2	*92,92%*	*96,00%*	*96,48%*	*97,61%*	*97,17%*	*96,04%*
3	*95,73%*	*98,20%*	*98,08%*	*98,72%*	*98,89%*	*97,92%*
4	*96,06%*	*98,34%*	*98,54%*	*99,13%*	*99,20%*	*98,25%*
5	*96,21%*	*98,40%*	*98,61%*	*99,05%*	*99,21%*	*98,30%*

Table 6. Testing accuracies for the modified architectures for each class (Left Ventricle [LV], Right Ventricle [RV, Left Atrium [LA], Right Atrium [RA]).
Source: Matlab

The discussion on these results is presented in section 4.3. To represent more visually the results shown in the table above, a random image from the testing subset was selected (see Figure 31) to show the **networks' predictions** depending on the Encoder-Decoder depth and the different classes. Results are shown in Figures 32.

Figure 31. Image from patient 1288 from the testing dataset.
Source: augmented Echogan dataset.



Figures 32. Predictions for each class (columns) and depth (rows) compared to the ground-truth (1st row above) (Left Ventricle [LV], Right Ventricle [RV, Left Atrium [LA], Right Atrium [RA]).

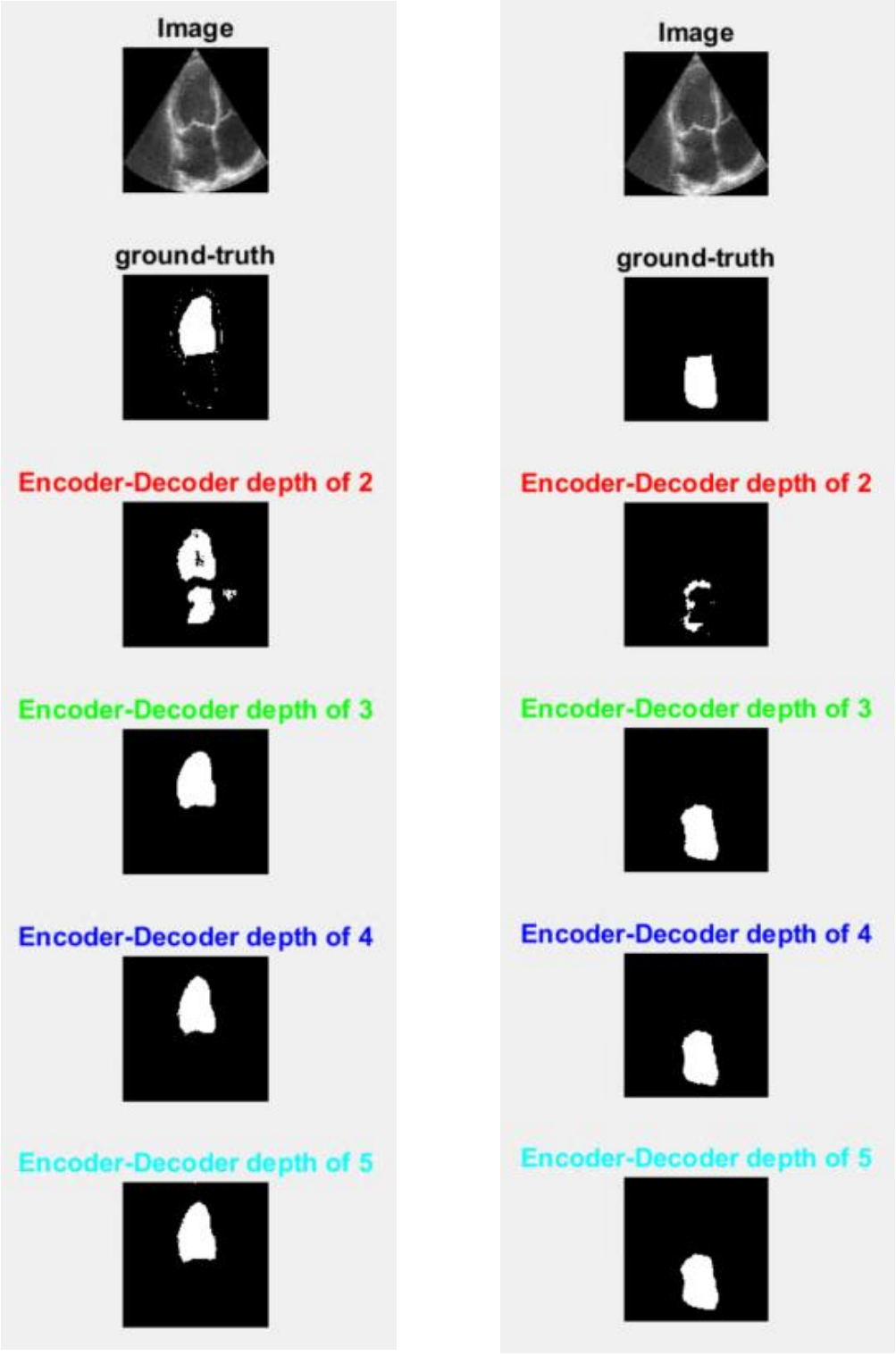
Source: Matlab

Furthermore, to see whether the obtained architectures were robust, all four modified U-Nets were tested on the **annotated CAMUS dataset** (see section 2.1.1) and the mean accuracy was computed for each case. As the CAMUS dataset only has annotations **for Left Ventricle and Left Atrium**, only these two classes were tested for End-Diastole and End-Systole moments, which are moments of relaxation and contraction of the Left Ventricle, respectively. Results are shown in Table 7:

E-D depth	LV ES	LV ED	LA ES	LA ED
2	94,37 %	95,15 %	95,07 %	96,69 %
3	96,79 %	96,94 %	97,05 %	97,63 %
4	97,07%	96,82 %	98,00 %	98,07 %
5	97,20 %	96,92 %	98,27 %	98,27 %

Table 7. Average accuracies of the architectures' predictions on CAMUS dataset.
LV = Left Venticle, LA = Left Atrium, ES = End-Systole, ED = End-Diastole, E-D = Encoder-Decoder.
Source: Matlab

Finally, to further visualize the results in Table 7, in Figures 33 below we showed the Left Ventricle and Left Atrium predictions of one selected patient for each U-Net architectures compared to the ground-truth reference. The discussion on these results is presented in section 4.3.



Figures 33. Predictions of each architecture for Left Ventricle (left) and Atrium (right) of patient 0037 at End-Systole moment from CAMUS dataset.
Source: Matlab

4. Discussion

Once the results have been presented, this section will consist of a detailed analysis of these results. Statistical plots will help to visualize the obtained information and extract conclusions from it. Each subsection consists of a discussion on each of the optimizations performed.

4.1. Hyperparameter Optimization

As can be seen in Table 4, the experiment carried out can be considered a **factorial experiment** with 3 variables and 4 responses. Each variable (Initial Learn rate, solver, and Mini Batch Size) has 2, 3, and 2 values respectively. First of all, the responses are going to be analyzed to see whether each one of them gives the same information or not, i.e., they are dependent on each other.

The scattered plot of the accuracies and losses (Figure 34) can provide this information by calculating the regression coefficient, which gives information about the **regression line's accuracy** of the analyzed variables. First, the training and validation accuracies are plotted (training accuracy on the x-axis and validation accuracy on the y-axis):

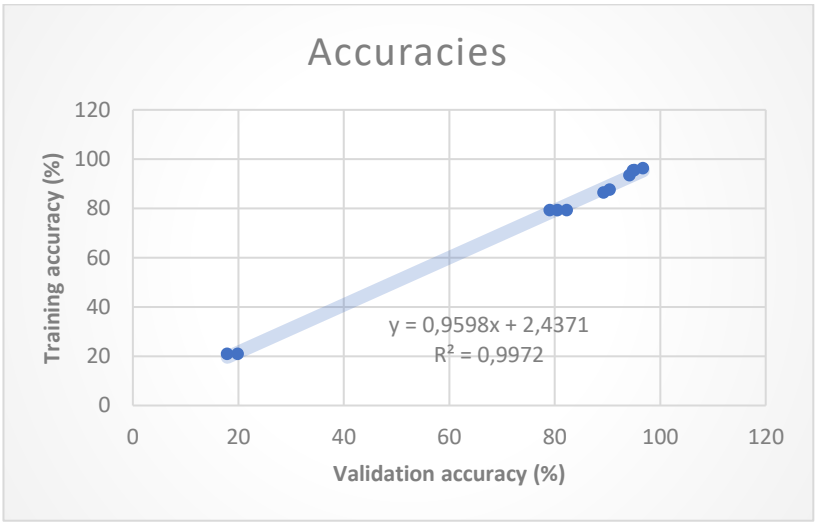


Figure 34. Training vs validation accuracies. Regression coefficient of 0,9972.

Source: Excel

Then, we can plot training loss and accuracy to see if they are dependent (see Figure 35). Training accuracy is plotted on the x-axis and training loss is plotted on the y-axis. This time the regression is exponential as the loss function is logarithmic:

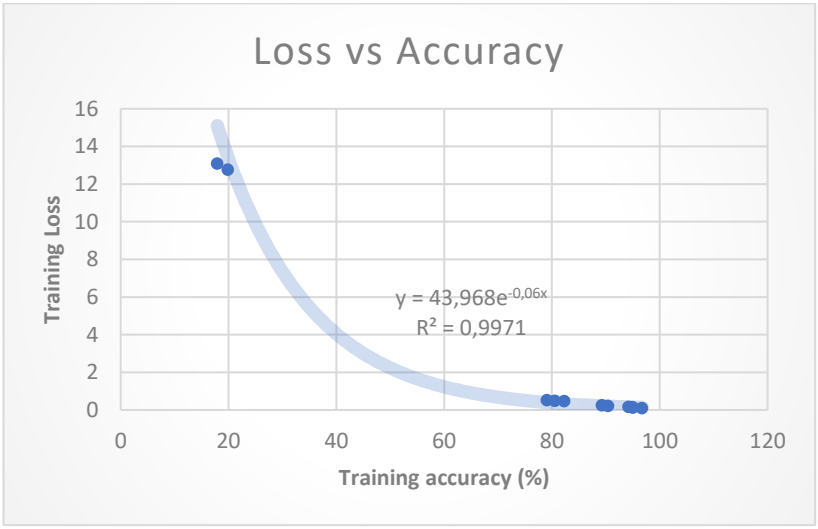
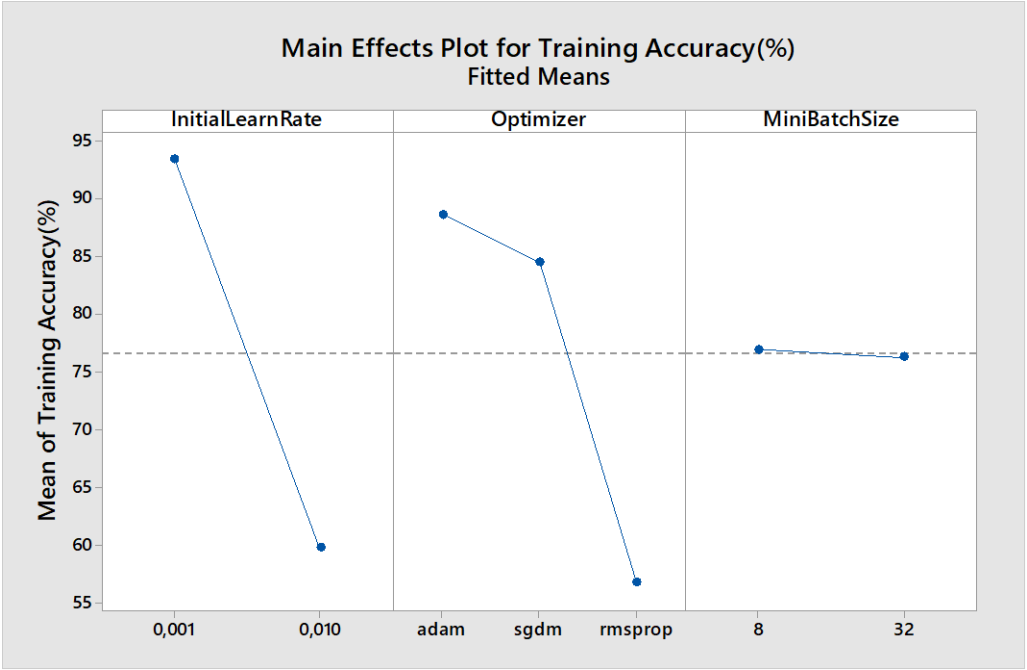


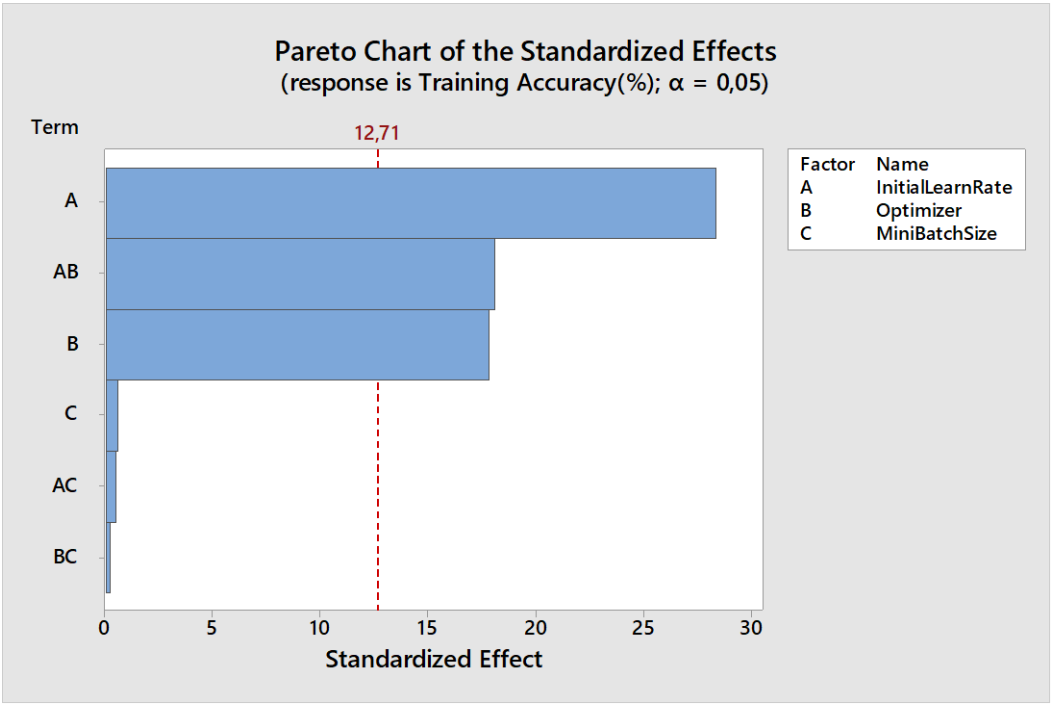
Figure 35. Training loss vs accuracy. Regression coefficient of 0,9971.
Source: Excel

The **dependency between responses** is very clear (regression coefficients of almost 1) as thought, therefore it can be established that all responses give the same information about the variables.

To measure the dependency and variability of the response (training accuracy) with each one of the variables, the **main effects and *Pareto* plots** can be shown (see Figures *36*).

The Main effect plot represents the mean value of the response for each variable value. This way the variables that influence the response the most can be identified:





Figures 36. Main effects and interaction plots of the hyperparameters and the training accuracy as the response.

Source: Minitab

As seen in the Pareto chart in Figures 36, the hyperparameter that affects the response the most is the **Initial Learn Rate** followed by the **Optimizer**. Mini Batch Size almost doesn't affect the responses, probably because the difference between the 2 chosen values is not very considerable. The election of Mini-Batch Size was conditioned by the fact that increasing its size affects the **computation capability**, and trials with values higher than 32 didn't run due to lack of memory.

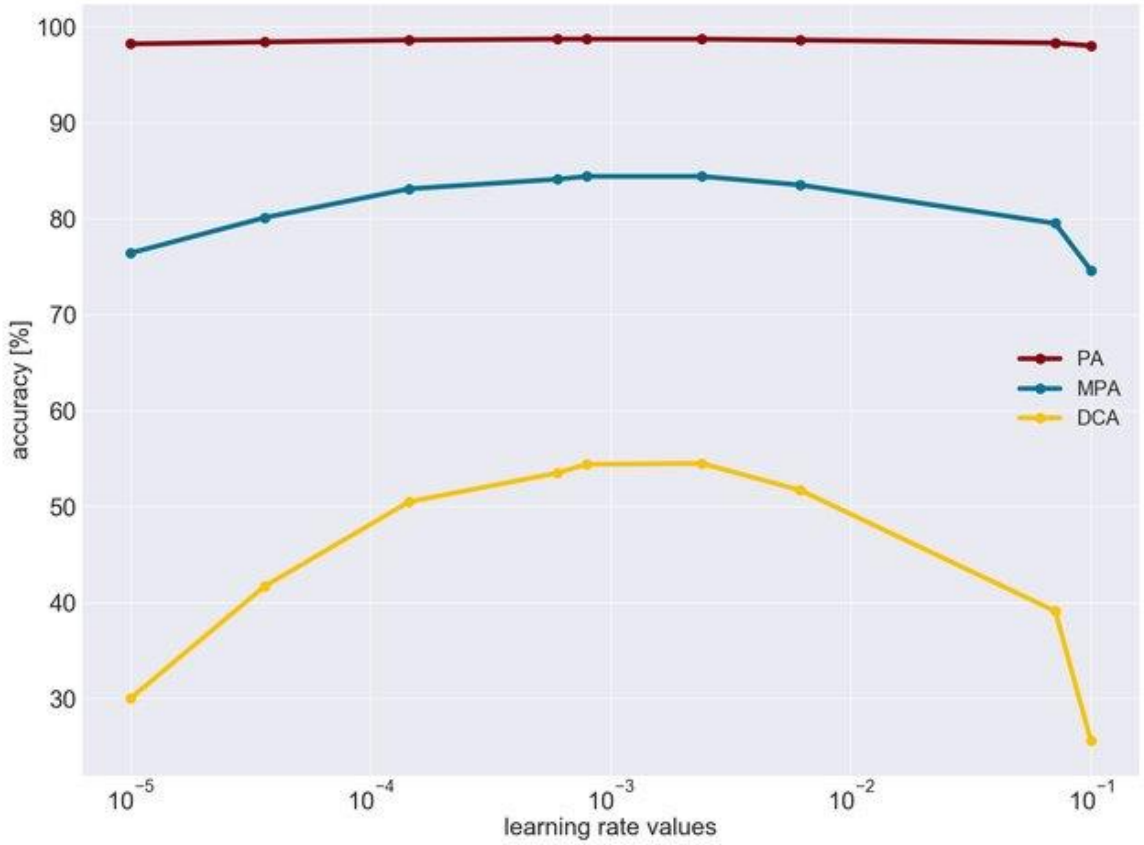
The reasoning behind the other hyperparameter influence over the responses is that **Initial Learn Rate** is known to be related to accuracy [30]. If the value is too high, the architecture ends up obtaining a sub-optimal result.

However, reducing Initial Learning Rate also increments the **computation time** and it doesn't ensure that the result would be better, so an optimal Initial Learning Rate Value has to be figured out for every case. A default value of Initial Learning Rate of 0.01 typically works for standard multi-layer neural networks but it doesn't mean that it is always necessary to rely exclusively on this default value. Matlab's website suggests using an Initial Learning Rate value of 0,01 for the sgdm solver and 0,001 for rmsprop and adam solvers [27]. To **validate** this hypothesis, an Initial Learning Rate of 0,0001 will be tested in section 3.2.

On the other hand, the **Adam** optimizer was supposed to be the best for this experiment. As said in its original article [31], this solver is aimed toward machine learning problems with large datasets and/or high-dimensional parameter spaces. It combines the advantages of two recently popular optimization methods: the ability of AdaGrad to deal with sparse gradients, and the ability of RMSProp to deal with non-stationary objectives. For this reason, adam optimizer works well in classification deep learning problems.

4.2. Further Hyperparameter Testing

The extra test, presented in section 3.2, showed that there is not a significant difference between the Initial Learning Rates of 0,001 and 0,0001. In general, this hyperparameter is known to be related to training computation time and accuracy:



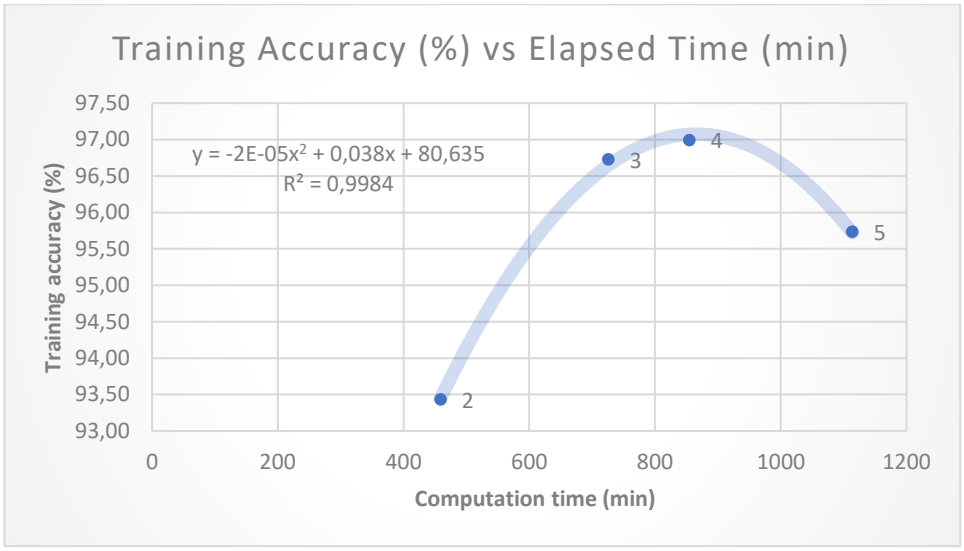
Figures 37. The influence of different initial learning-rate values on network performance, namely pixel accuracy (PA), mean pixel accuracy (MPA), and defect-class accuracy (DCA).
Source: Stern, Maike. (2020). Development of a Fully-Convolutional-Network Architecture for the Detection of Defective LED Chips in Photoluminescence Images.

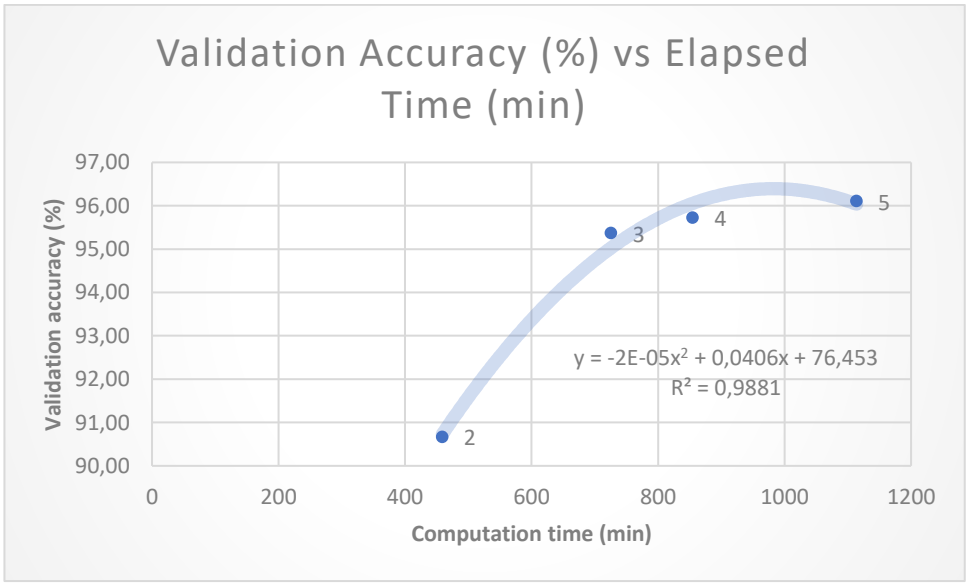
In Figures 37 above we see that the best accuracies are obtained for values between 0,01 and 0,0001. These values strongly depend on the network and the problem to be solved.

Given the results in section 3.2, we can conclude that there's **no significant difference** in accuracy between choosing 0,001 and 0,0001 as Initial Learning Rates with this combination of hyperparameters specified. However, in the architecture optimization, an Initial Learn Value of 0,0001 will be used given the reduction in the computation time.

4.3. Architecture Optimization and Testing

After concluding the hyperparameter optimization, the focus is now on the architecture and the results presented in section 3.3. As can be seen in Table 5, the trial with the **highest training accuracy** was the one with depth 4. However, to analyze the dependency of accuracy vs computation time, we can focus on the scattered plot as we did in the previous hyperparameter optimization to analyze the dependency on the response and variables:





Figures 38. Scattered plot of Training and Validation Accuracies (%) on the y-axis and Elapsed Time in minutes on the x-axis. Regression equation and coefficient are shown.

Source: Excel

As shown in Figures 38 above, architectures with Encoder-Decoder depths of **3 and 4 outperformed** the ones with depths of 2 and 5 in terms of training accuracy. It is interesting to notice that the trial with an Encoder-Decoder depth of 5 was less accurate in training than the one with a depth of 4. This fact is opposite to what other articles suggest [25], stating that CNNs with few layers perform better at segmenting small objects, and CNNs with a high number of layers are more effective at segmenting large objects. However, the difference between accuracies was so low that these two trials can be considered almost equally accurate.

Despite having few trials, the accuracies seem to be related to a **polynomial equation** with a regression coefficient of 0,998. Architectures with a higher number of Encoder-Decoder depths must be evaluated to see if the training accuracy keeps decreasing as the depth increases.

For the two best trials (the architectures with 3 and 4 Encoder-Decoder depths) we see that the time difference in the training duration is considerable: 2 hours and 9 minutes which is a 15,1 % reduction from depth 4 (default value) trial to depth 3 trial. The accuracy only decrements by a factor of 0,28%. Therefore, here can be seen that choosing an **Encoder-Decoder depth of 3** can be a good option for still acquiring high training accuracies and considerably less computation time, especially when an important amount of data is available.

In Table 6, the accuracies of **each class** for **each depth** on the testing subset were calculated. We can represent these results in line charts to further analyze the dependency between classes and the accuracy of these networks:

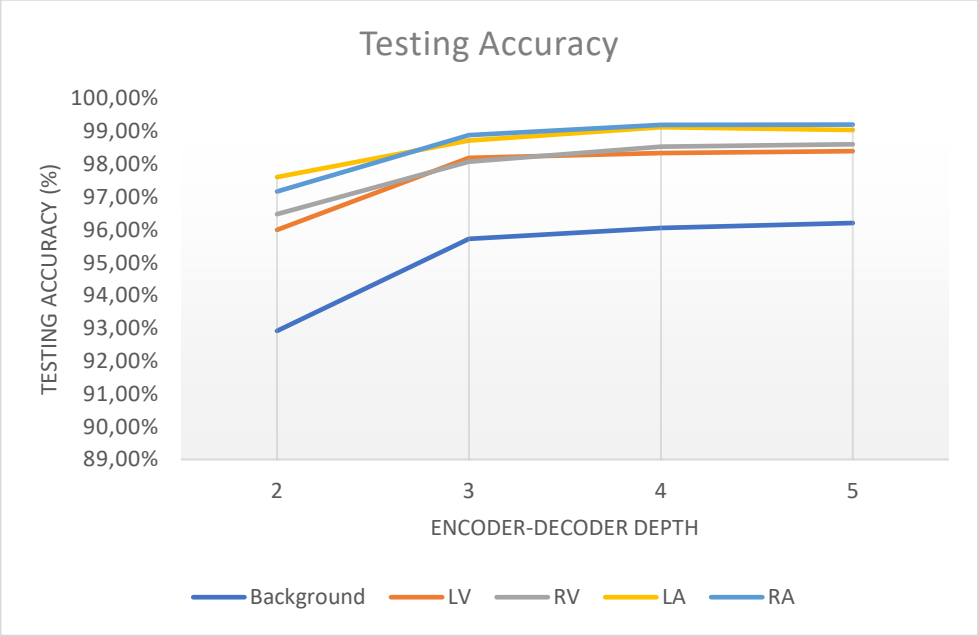


Figure 39. Accuracies (y-axis) on testing for each architecture's depth (x-axis) and class. (Left Ventricle [LV], Right Ventricle [RV, Left Atrium [LA], Right Atrium [RA]).
Source: Excel

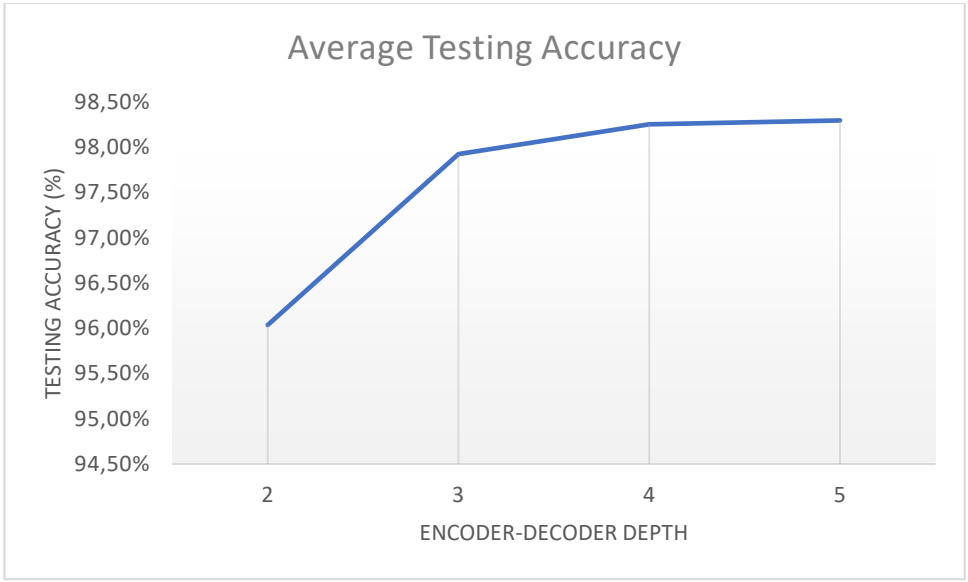


Figure 40. Average accuracies (y-axis) on testing for each architecture's depth (x-axis).
Source: Excel

As can be seen in Figure 39 and Figure 40, the **background accuracy** is significantly lower than the rest of the classes' accuracies. This is explained by the fact that the background contains the weighted product of inaccuracies of the rest of the classes, as it is the negative image of the four chambers silhouette (see Figures 32).

The rest of the classes' accuracies go according to the **dimension** of their shape. For example, the Left Ventricle's predictions are less accurate than the Right Ventricle's predictions because the Left Ventricle is bigger in the image and therefore contains more inaccuracies.

However, if we analyze the testing accuracy differences between 3, 4, and 5 depths, we can see that are relatively low. Taking depth 4 (default U-Net's value) as a **reference**, changing the depth to 5 increments accuracy by a factor of 0,04% and changing to 3 decrements accuracy by a 0,34%. Changing to depth 2 decrements by a factor of 2,26%.

Although this might not seem very much, the accuracy (defined in section 3.1) computes the **mean difference between predictions and ground-truth masks** of all parts of the image. The inaccuracies in the predictions, as seen in Figures 32, are located in the chamber's boundaries, so the accuracy might seem very high because the chamber is correctly predicted, but the edges of the chamber might not be that accurate. This inaccuracy is very visible for depth 2, but predictions at depths 3, 4, and 5 have very similar aspects.

To further explore the influence of the Encoder-Decoder depth, and also to test the robustness of the modified architectures, we tested them with the **CAMUS annotated dataset**, as explained in section 3.3. Results of this testing are presented in Table 7. In Figure 41, we can visualize these results more clearly with a line chart to see the dependence on the Encoder-Decoder depth.

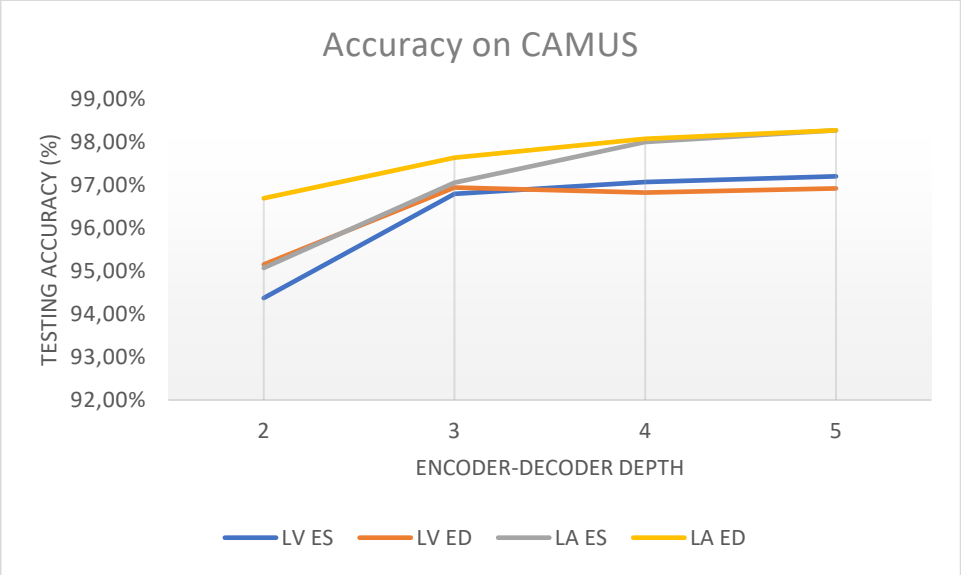


Figure 41. Accuracy (y-axis) on CAMUS dataset for each architecture's depth (x-axis) and part and moment. LV = Left Venticle, LA = Left Atrium, ES = End-Systole, ED = End-Diastole.
Source: Excel

The graph is pretty similar to the accuracies on the testing subset shown in Figure 39. Accuracies of architectures with Encoder-Decoder depths of 4 and 5 are **almost equal** and only slightly better than the architecture with a depth of 3. Visual results in Figures 33 show pretty similar results on the predictions with architecture depths of 3, 4, and 5.

Therefore, given the significant computation time reduction and the non-significant accuracy reduction in U-Net architecture with an Encoder-decoder depth of 3 compared to the one with a depth of 4 (default value), we can conclude that U-Net with an **Encoder-Decoder depth of 3 can be chosen as more optimal** in terms of accuracy vs computation time.

5. Conclusions

After presenting and analyzing the results of both optimizations, in this section the main achievements of this work are summarized and the work for future papers is established.

5.1. Thesis summary

In this study, we have obtained, implemented, and optimized a Convolutional Neural Network, U-Net, for automatically segmenting the four chambers of the Heart in B-Mode Four Chamber Apical View echocardiography images.

The **first objective**, explained in section 1.3, was obtaining echocardiographic labeled data (with the ground-truth attached) of the four chambers of the Heart. In section 2.1.3 we achieved this objective by obtaining and augmenting a four-chamber labeled dataset.

The **second objective** was choosing an accurate Deep Learning architecture. In section 2.2.1 we proposed U-Net architecture, known to achieve good results for semantic segmentation purposes in recent articles.

The **third objective** was to optimize the architecture to achieve good performance and less computation time. We explored two optimization paths in sections 2.2.2 and 2.2.3 respectively, the first one optimizing the training hyperparameters and the second one modifying the Encoder-Decoder depth of the network. Results of these optimizations were presented in section 3.

Therefore, the purpose of providing an optimized and accurate Deep Learning architecture to automatically segment the four chambers of the heart has been achieved.

5.2. Future Work

As explained in section 2.1, the **lack of labeled data** is one of the biggest challenges in this area. Re-training the optimized networks with new datasets from different patients and ultrasound equipment would increase the accuracy of automatic segmentations. Also, we used data from the 4 inner chambers of the heart, but creating new classes for the heart pericardium, valves, etc. and training the network with labeled data would improve the network value for

doctors.

For the proposed optimizations, future work on hyperparameters and the architecture has to be made. This work had a limited duration and we had to limit the number of experiments, but hyperparameters like loss function and max epochs have to be further optimized to achieve better accuracies while maintaining a low training computation time. Mini-batch values can be further explored with a more capable computing system.

In addition, further architecture modifications like removing and adding new types of layers have to be made to achieve more optimal results. Encoder-Decoder depths of 6 or higher values have to be evaluated.

Finally, in this work, we only focused on the specific task of semantic segmentation with Deep Learning. Image accuracies were computed to compare the results, but other indicators like Ejection Fraction or other indexes extractions have to be explored, as same as other medical tasks like pose classification for other types of echocardiography view modalities.

6. References

1 K. L. Moore, A. F. Dalley y A. M. R. Agur, Clinically Oriented Anatomy, Wolters Kluwel Health/Lippincott Williams & Wilkins, 2009.

2 M. J. Curtis, The Heart and Cardiovascular System, Cardiovascular Research., 1992.

3 S. H. Care, «What Is An Echocardiogram?» [En línea]. Available: https://stanfordhealthcare.org/medical-tests/e/echocardiogram/types.html.

4 J. K. Oh, Echocardiography in heart failure: Beyond diagnosis, European Journal of Echocardiography, 2007.

5 A. H. Association, «heart.org» American Heart Association editorial staff, 31 May 2017. [En línea]. Available: https://www.heart.org/en/health-topics/heart-failure/diagnosing-heart-failure/ejection-fraction-heart-failure-measurement.

6 S. Moses, «Apical Four Chamber Echocardiogram View» de Cardiovascular Medicine Book, 2020.

7 L. Deng y D. Yu, "Deep Learning: Methods and Applications", Retrieved, 2014.

8 J. Mahanta, «Introduction to Neural Networks, Advantages and Applications» Towards Data Science, 10 July 2017.

9 H. Singh, «Deep Learning 101: Beginners Guide to Neural Network» Anaytics Vidhya, 1 March 2021.

10 P. Knowledgebase, «The role of bias in Neural Networks» [En línea]. Available: https://www.pico.net/kb/the-role-of-bias-in-neural-networks/.

11 P. Baheti, «Activation Functions in Neural Networks [12 Types & Use Cases]» [En línea]. Available: https://www.v7labs.com/blog/neural-networks-activation-functions. [Último acceso: 10 June 2022].

12 G. L. Team, «Types of Neural Networks and Definition of Neural Network» Great Learning, 25 September 2021. [En línea]. Available: https://www.mygreatlearning.com/blog/types-of-neural-networks/.

13 S. Saha, «A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way» Towards Data Science, 15 December 2018. [En línea]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

14 M. Pinedo, E. Villacorta, C. Tapia, R. Arnold, J. López, A. Revilla, I. Gómez, E. Fulquet y J. San Román, «Inter- and intra-observer variability in the echocardiographic evaluation of right ventricular function,» Revista Española de Cardiología (English Edition), pp. 802-809, 2010.

15 C. Armstrong, «Quality control and reproducibility in M-mode, two-dimensional, and speckle tracking

echocardiography acquisition and analysis: The CARDIA study, year 25 examination experience» de Echocardiography, 2015, pp. 1233-1240.

16 S. Leclerc, E. Smistad, J. Pedrosa, A. Ostvik, F. Cervenansky, F. Espinosa, T. Espeland, E. Berg, P. M. Jodoin, T. Grenier, C. Lartizien, J. Dhooge, L. Lovstakken y O. & Bernard, «Deep Learning for Segmentation Using an Open Large-Scale Dataset in 2D Echocardiography» IEEE transactions on medical imaging, pp. 2198-2210, 2019.

17 J. Zhang y e. al., «Fully automated echocardiogram interpretation in clinical practice» Circulation, pp. 1623-1625, 2018.

18 A. Arghavan, M. Daisuke, A. M. R., A. M. Reza, A. R. A y J. H. a. K. Arash, «Generalizable fully automated multi-label segmentation of four-chamber view echocardiograms based on deep convolutional adversarial networks» Journal of the Royal Society Interface, 2020.

19 O. Ronneberger, P. Fischer y a. T. Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation» LNCS 9351, pp. 234-241, 2015.

20 D. Ouyang, B. He y A. Ghorbani, «Video-based AI for beat-to-beat assessment of cardiac function» Nature, pp. 252-256, 2020.

21 Ilastik, «ilastik the interactive learning and segmentation toolkit» Ilastik, 2010. [En línea]. Available: https://www.ilastik.org/index.html.

22 «Seg3D: Segmentation Image Processing» The NIH/NIGMS Center for Integrative Biomedical Computing , 1998. [En línea]. Available: https://www.sci.utah.edu/cibc-software/seg3d.html.

23 «3D Slicer image computing platform» Slicer, [En línea]. Available: https://www.slicer.org/.

24 Q. C. Chen Chen, Q. Huaqi, T. Giacomo, D. Jinming, B. Wenjia y R. Daniel, «Deep Learning for Cardiac Image Segmentation: A Review» Frontiers in Cardiovascular Medicine, vol. 7, 2020.

25 B. Calm Salvans, «Development of fully automated echocardiographic data interpretation technique» 2021. [En línea]. Available: http://hdl.handle.net/10230/48192.

26 MATLAB, «unetLayers,» MATLAB, 2022. [En línea]. Available: https://es.mathworks.com/help/vision/ref/unetlayers.html.

27 MATLAB, «trainingOptions,» MATLAB, 2022. [En línea]. Available: https://es.mathworks.com/help/deeplearning/ref/trainingoptions.html.

28 MATLAB, «Experiment Manager,» MATLAB R2022a, [En línea]. Available: https://es.mathworks.com/help/deeplearning/ref/experimentmanager-app.html.

29 M. Answers, «How to avoid NaN in the Mini-batch-loss from traning convolutional neural network?» MATLAB, 27 April 2017. [En línea]. Available: https://es.mathworks.com/matlabcentral/answers/337587-how-to-avoid-nan-in-the-mini-batch-loss-from-traning-convolutional-neural-network.

30 Y. Bengio, «Practical Recommendations for Gradient-Based Training of Deep Architectures» Cornell University, 2012.

31 D. P. Kingma y J. Ba, «Adam: A Method for Stochastic Optimization» 2014. [En línea]. Available: https://arxiv.org/abs/1412.6980.

32 MATLAB, «Explore Semantic Segmentation Network Using Grad-CAM» MATLAB, [En línea]. Available: https://es.mathworks.com/help/deeplearning/ug/explore-semantic-segmentation-network-using-gradcam.html.

33 MATLAB, «Specify Layers of Convolutional Neural Network» MATLAB, [En línea]. Available: https://es.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html#mw_ad6f0a9d-9cc7-4e57-9102-0204f1f13e99.

34 K. He, S. R. X. Zhang y J. Sun, «Delving Deep Into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification» Proceedings of the IEEE International Conference on Computer Vision, pp. 1026-1034, 2015.

35 S. Ren, Y. Wang, R. Hu, L. Zuo, L. Liu y a. H. Zhao, Automated segmentation of left ventricular myocardium using cascading convolutional neural networks based on echocardiography, Department of Ultrasound of Xijing Hospital, Xijing Hypertrophic Cardiomyopathy Center, Fourth Military Medical University, Xi'an, Shaanxi 710032, China, 2021.

36 L. Xu, M. Liu, Z. Shen, H. Wang, X. Liu, X. Wang, S. Wang, T. Li, S. Yu, M. Hou, J. Guo, J. Zhang y Y. & He, «DW-Net: A cascaded convolutional neural network for apical four-chamber view segmentation in fetal echocardiography.,» Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society.

37 S. Moradi, M. G. Oghli, A. Alizadehasl, I. Shiri, N. Oveisi, M. Oveisi, M. Maleki y J. Dhooge, «MFP-Unet: A novel deep learning based approach for left ventricle segmentation in echocardiography» de Physica Medica, 2019, pp. 58-69.

38 F. Liu, K. Wang, D. Liu, X. Yang y J. Tian, «Deep pyramid local attention neural network for cardiac structure segmentation in two-dimensional echocardiography» de Medical Image Analysi, 2021.

39 Hang, &. Zhang, &. Chen y G. &. Wang., «Classification of Plant Leaf Diseases Based on Improved Convolutional Neural Network» de Sensors, 2019.