Final Degree Project

Biomedical Engineering Degree

Coronary artery segmentation using Transformer Neural Networks

Barcelona, 6th of June 2022

Author: Claudia Sanchez Gomez

Directors: Oscar Camara

Abdel Hakim Moustafa

César Acebes

Tutor: Roser Sala

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my director Oscar Camara for giving me the opportunity to carry out this fantastic project. I would also like to highlight his support and efficient help during the process.

My deep gratitude goes to my co-director Abdel Hakim Moustafa and my fellow César Acebes, for their endless support, patience, kindness, encouragement, insightful comments, and all the fun we have had in the last year. I am extremely grateful to them.

Also, this work would not have been possible without the Cardiac Imaging Department at Hospital de la Santa Creu I Sant Pau in Barcelona.

In addition, I would want to express my gratitude to my tutor Roser Sala, for her thoughtful comments, feedback and recommendations, and for accompanying me as a teacher during the degree.

Last but not least, on a more personal note, I would like to thank my family and friends for their unconditional support, with special mention to my grandmother and my mother.

ABSTRACT

Coronary Artery Disease (CAD) is the leading cause of death in developed countries. It is a multi-factorial disease consisting of a plaque accumulation in the coronary vessels, causing ischemia or myocardial infarction. An early diagnosis is important to avoid fatal consequences. Nowadays, Computed Tomography (CT) images are used as a diagnostic tool as well as a technique for selecting appropriate therapies for CAD patients. In order to compute disease-related metrics, it is necessary to process these images, including the coronary artery delineation. In the last years, the use of Artificial Intelligence (AI) has grown exponentially in clinical environments, especially in time-consuming tasks, such as image segmentation. There exist plenty of AI algorithms that have proven good performance in these tasks, including Transformers Neural Networks. Hence, the main aim of this project was to develop a coronary artery segmentation algorithm using this approach and study its performance to evaluate its potential in clinical practice. The results showed segmentations with the coronary artery shape well-defined but with several stops in the segmentation of the main branches and a huge presence of artefacts. These could be solved by computing a longer training using an extended dataset in the future, allowing their implementation in the clinical field. As healthcare professionals would only be responsible for the validation of the segmentation, they could devote more time studying markers to enhance the diagnosis of patients and provide a more personalised treatment. The created algorithm may work nowadays as a support tool in semi-automated segmentation.

Keywords: Artificial Intelligence; Deep Learning; Segmentation; Cardiovascular Disease; Cardiac Imaging.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AIMDD	Active Implantable Medical Device Directive.
AI	Artificial Intelligence.
ASOCA	Automatic Segmentation of the Coronary Arteries.
CAD	Coronary Artery Disease.
CCTA	Coronary Computed Tomography Angiography.
CFD	Computational Fluid Dynamics.
CNN	Convolutional Neural Networks.
CUDA	Compute Unified Device Architecture.
CT	Computed Tomography.
ECG	Electrocardiogram.
EU	European Union.
FCN	Fully Convolutional Networks.
FFR	Fractional Flow Reserve.
GSPRs	General Safety and Performance Requirements.
LDL	Low-Density Lipoprotein.
ML	Machine Learning.
MRI	Magnetic Resonance Imaging.
MDCG	Medical Device Coordination Group.
MDD	Medical Device Directive.
MDSW	Medical Device Software.
MDR	Medical Device Regulations.
MLP	Multilayer Perceptron
MSA	Multi-head Self-Attention.
PACS	Picture Archiving and Communication System.
SA	Self-Attention.
UNETR	U-Net Transformers.

1. INTRODUCTION

According to the World Health Organization [1], one-third of the deaths worldwide is caused by coronary artery disease (CAD) [2]. CAD is generated by the presence of plaques in the coronary artery that can restrict the blood flow to the heart muscle, producing the death of the cardiac tissue. It is necessary to diagnose the population in an early stage of the disease to prevent potential risks and the fatal consequences. Computed tomography (CT) imaging is a non-invasive CAD diagnosis method that is used nowadays as an alternative to the current invasive methods [3]. Although CT is the most used imaging technique in cardiology, there exist multiple types of imaging modalities that variably provide functional and anatomical delineation of coronary anatomy and help guide appropriate therapy. The acquired images have to be processed manually before the analysis and the further diagnosis by the physician.

In the last years, the use of Artificial Intelligence (AI) in clinical environments has grown exponentially due to its good performance in the automatization of a wide range of tasks such as accurate diagnosis, helping professionals in the choice of appropriate therapies, making prognostics of diseases as well as processing raw images [4]. AI-based algorithms are the most innovative tools in segmentation, which means delineating the anatomical structures from an image. Image segmentation is a potential tool for diagnosis since it allows providing quantitative information about the anatomical structures [5]. Currently, segmentation is done semi-manually by professionals. This task requires plenty of time and it is not only is prone to error but also could be replaced by a machine**.**

Before the implementation of AI-based algorithms, there existed several models that were able to segment the coronary artery automatically. These models were region-growing models [6], [7], deformable models, the ones based on statistical analysis [8], minimal pathfinding, vessel tracking [9], differential analysis [10], or mathematical morphology models [11]. They presented a good performance in segmentation, giving a clear structure of blood vessels with a very high image resolution. However, they needed much time to perform the segmentation task.

Nowadays, AI-based algorithms in different imaging modalities have a great performance in cardiac segmentation. Depending on the main objective, several architectures can be used, as it will be described in the following sections. In the coronary artery, there exist algorithms for the segmentation of the lumen and the atherosclerotic plaque [4]. The most recent AI-based architectures that are used to perform the segmentation of the coronary artery are based on convolutional neural networks (CNN) or similar architectures [12]–[14]. The majority have a good performance and are accurate, segmenting the coronary vessels with high resolution and without errors. However, most of them require large quantities of data. Over the last year, Transformer Neural Networks, which are a novel AI-based architecture commonly used in language processing, have gained popularity in a wide variety of environments. They are designed to handle input data in order to perform the following transduction, transforming any input sequence into an output, focusing in the most important part of the information they are given. UNETR (U-Net Transformers) was recently introduced as a Transformer-based segmentation tool [15]. In medical imaging, it is able to learn, by paying attention, the contextual information of input volume and predict new inputs Its main advantages are the low training time and the good performance in segmentation tasks in different multi-modality images. For example, in brain segmentation, UNETR outperforms a closest baseline by 1.5% on average regarding other architectures [15]. This project consists in the development of a UNETR-based algorithm and the evaluation of its performance in the segmentation of the coronary artery from cardiac CT images.

1.1. Motivation

Coronary heart disease is currently the leading cause of death in developed countries. AI-based segmentation algorithms of the coronary arteries can help professionals not only for an easy and practical diagnosis of CAD but also in therapy guidance.

The factor that motivates me the most to carry out this project was the fact that I was able to use a novel segmentation approach such as UNETR, improving the actual segmentation techniques and enhancing coronary artery disease diagnosis. Apart from the clinical necessity, this project is also driven by personal motivation and keenness toward the biomedical imaging and the AI fields. For me, it has been a unique opportunity to take part in this challenging project.

Last but not least, working hand in hand with the Cardiac Imaging Department in Hospital de la Santa Creu i Sant Pau and Professor Oscar Camara's research group (Physense: Sensing in Physiology and Biomedicine) at Universitat Pompeu Fabra was also an encouragement. Not only for the reputation that these institutions entail but also because of the feasibility that they have added to the project.

1.2. Objectives

Segmentation is one of the most important bottlenecks in medical imaging. Although it is a useful tool to obtain diagnostic insights, the current methods tend to be manual and labour-intensive. Consequently, professionals spend a lot of time performing repetitive tasks that can be learned by machines. The development of automatic algorithms able to perform segmentation tasks in cardiology would represent a significant advance for the hospitals, not only for the reduction of time, but also as an early diagnosis tool. The research in using AI tools in segmentation has grown exponentially in the last years due to their good performance in medical imaging [4], especially Transformer Neural Networks, a novel segmentation approach that has shown better clinical outcomes than the most used architectures. Thus, the main aims of this project can be summarized as:

i. To develop a Transformer-based algorithm for the segmentation of the coronary arteries from cardiac computed tomography images and to study its performance in these images.
ii. To perform a quantitative and qualitative analysis of the algorithm using new patient images.
iii. To study the viability of implementing the created algorithm into a clinical workflow.
iv. To compare the trained algorithm with a different AI-based segmentation approach previously trained using the same dataset but outside the Hospital's facilities.

1.3. Methodology

In order to reach the main objective of the project, the steps presented in this section were carried out. They were performed under the supervision of César Acebes and the direction of Abdel Hakim Moustafa and Oscar Camara, each of them in their field of expertise.

First and foremost, an intense bibliographic search about coronary artery disease was done in order to understand the clinical problem to figure out. After understanding the basis of the different medical image modalities in cardiology and the importance of the AI in clinical workflows, especially in segmentation tasks, different cardiac image segmentation papers using deep learning in the most commonly used image modalities were studied in order to know the stage of art of our project.

Moreover, my personal experience in the hospital allowed me to get familiar with the clinical workflow in the hospital, as well as how professionals work with images.

Then, some tutorials were done in order to understand deeply the mathematical fundamentals of neural networks and UNETR [16], and their implementation in a segmentation workflow. It was important to focus on the importance of modifying the parameters in order to obtain better outputs.

A reduced cardiac CT images dataset, labelled by César Acebes in January 2021 for his Master's Thesis [17] was used. An additional extended dataset was going to use for training, comprised of 200 more labelled images. However, problems related to data format occurred and the extended dataset was not taken into consideration in the final training.

Once the operating system Ubuntu version 20.04 [18], and the required libraries and toolboxes were installed in the hospital's computer, as well as the needed virtual environment, the selection of the best hyperparameters to achieve the best segmentation possible was the next step. It was done by an ablation study; the network was trained with different combinations of hyperparameters in order to study how much they affecteded the model's accuracy.

Subsequently, we trained the model with the parameters that yielded the most optimal results in the ablation study to archive the best segmentation accuracy in a suitable time for the clinical practice. The UNETR architecture [16] was compared with standard U-Net architectures under the nnU-Net framework [19], the most used deep learning-based architecture nowadays, and we came up with conclusions about the implementation of the novel transformer-based architecture in hospitals.

Finally, the present Final Degree Project was written and the oral presentation was prepared. Regarding the thesis, it is organized in the following way:

- Section 2. Background of deep learning principles and the clinical importance of coronary artery disease. Review of the state or art in coronary artery segmentation deep learning-based algorithms for CT images.
- Section 3. Market analysis of deep learning-based algorithms able to segment coronary artery.
- Section 4. Description of actual and alternative alternatives to the project. Explanation of UNETR.
- Section 5. Description of the development of the UNETR-based algorithm for coronary artery segmentation.
- Section 6, 7, 8 and 9. Execution of the chronogram, financial and technical viability, and description of legal aspects.
- Section 10. Conclusion of the whole project

1.4. Limitations and scope

The achievement of the segmentation algorithm using Transformer Neural Networks in order to segment the coronary arteries is the key for the whole global project. In this section the scope of the project will be explained in detail in accordance with the accomplishment of its objectives, but also with the existent limitations such as time and logistical constraints.

The project includes the segmentation algorithm using Transformers Neural Networks trained using 2500 epochs with a 32 images dataset. The hyperparameters of the network were chosen by performing an ablation study of determined hyperparameters. The volumetric meshes and the inferences of the new patients images were obtained at the end of the project. Moreover, the limitations of the algorithm as well the potential value, for patients and professionals, which could bring to incorporate the algorithm into the clinical workflow were also studied. The comparison of the algorithm trained with the same dataset but outside the Hospital’s facilities with other segmentation model was also performed.

The project does not include a training with a huge dataset obtained from CT images from a unique CT scan, as two scans were used for the used dataset. The dataset labels included the coronary arteries as well as the aorta. Moreover, it does not include an end-to-end interface incorporating the segmentation and the post-processing tools. However, it is expected to be done in the project future lines.

1.5. Location of the project

The project has been developed in collaboration with the Cardiac Imaging Department at Hospital de la Santa Creu I Sant Pau, in Barcelona. It has been performed in the Hospital’s facilities under the daily supervision of César Acebes, a biomedical engineering and AI researcher, and under the direction of Abdel Hakim Moustafa, a cardiologist with high expertise in cardiac clinical imaging. Oscar Camara, the director of the Information and Communication Technologies Department (DTIC) at Universitat Pompeu Fabra, directed the project. Roser Sala Llonch, Associate Professor at the University of Barcelona's Department of Biomedicine, tutored and supervised this thesis.

2. BACKGROUND

2.1. Anatomy of the coronary artery

The heart is the main organ in the cardiovascular system. It is primarily responsible for pumping blood around the body, delivering oxygen and nutrients to cells, and removing waste products. In order to achieve its function efficiently and subsequently the homeostasis of the body, the cardiac muscle is blood-supplied by the coronary tree, which is a network of blood vessels that wrap the heart, as it is seen in *Figure 1.*

There are two main coronary arteries: the right coronary artery and the left one. Both originated from the root of the aorta. The right coronary artery, which is divided into smaller branches called the right marginal artery and the posterior descending artery, supplies blood primarily to the right atrium and to the right ventricle, and to the atrioventricular and sinoatrial nodes. On the other hand, the left coronary artery is divided into two main branches called the left anterior descending artery and the circumflex artery [20]. Both are responsible for perfusing blood mainly to the left ventricle.

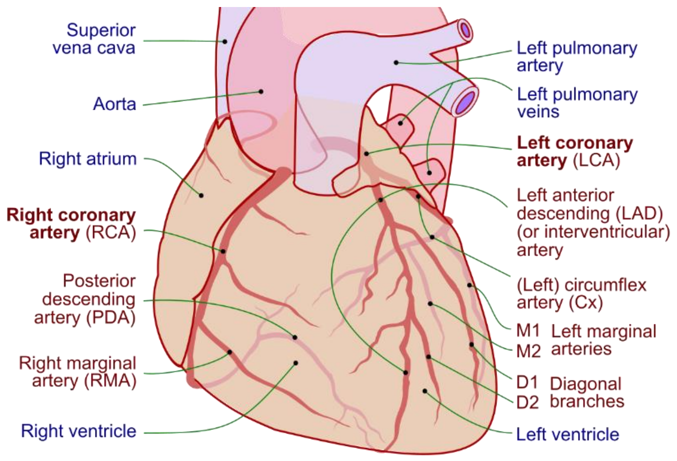

Superior vena cava
Aorta
Right atrium
Right coronary artery (RCA)
Posterior descending artery (PDA)
Right marginal artery (RMA)
Right ventricle
Left pulmonary artery
Left pulmonary veins
Left coronary artery (LCA)
Left anterior descending (LAD) (or interventricular) artery
(Left) circumflex artery (Cx)
M1 M2 Left marginal arteries
D1 D2 Diagonal branches
Left ventricle

Figure 1. Anatomy of the coronary tree, with coronary arteries labelled in red text and other landmarks in blue text. From Lynch, P. J. [21]

As said before, the coronary arteries are organized and subdivided into small branches that provide blood to the heart. Thus, an obstruction in these vessels can lead into an ischemia and consequently into a tissue deterioration and a sudden death since there is a reduction of the blood flow. A complete blockage of the blood vessel can produce a heart attack. Obstructions can occur due to many different factors but they are usually related to cholesterol and fibrous tissue blockages. This event requires immediate treatment and an early diagnose can prevent fatal consequences such as death.

2.2. Coronary artery disease

Coronary artery disease (CAD) is a multi-factorial disease caused by an increase of apolipoprotein B-containing lipoproteins concentration in blood, of which low-density lipoprotein (LDL), colloquially known as "cholesterol", usually is the most prevalent form [22]. More often, however, the disease is developed at lower levels of LDL in combination with other risk factors such as glucose intolerance, isolated systolic hypertension, genetics, or smoking, among others [23]. The process of accumulating cholesterol in the walls of the coronary artery tree is called atherosclerosis.

As a result, a lipoprotein plaque is built up due to intimal inflammation, necrosis, fibrosis and after some decades, calcification processes. It is important for diagnosing CAD if the coronary plaque is calcified or non-calcified. *Figure 2* shows the main problems associated with atherosclerosis. Plaque accumulation can lead to stenosis, which means the deformation of the blood vessel, or to a plaque rupture, which is the plaque disruption and fissuring. Stenosis causes a luminal narrowing and, consequently, the blood flow throughout the blood vessel is reduced, causing a possible ischemia or, if it is totally obstructed, a deterioration of the cardiac tissue that is being irrigated by the vessel. On the other hand, a plaque rupture will cause a thrombosis since the coagulation cascade is activated after the disruption. Therefore, this thrombus will block completely the blood vessel and there will be a lack of blood flow in the surrounding cardiac muscle tissue. This process is called myocardial infarction, colloquially known as "heart attack", and it requires immediate treatment to survive.

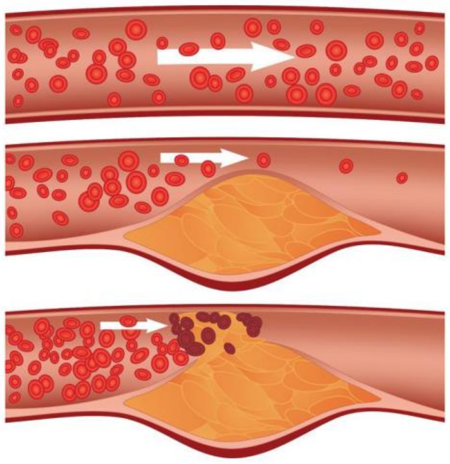


Figure 2. Possible associated problems with atherosclerosis. It is shown, in order from up to down, a healthy coronary artery, a stenosis and a plaque rupture. From University of Rochester Medical Centre [24].

Myocardial infarction and ischemia are two of the most potentially dangerous consequences of coronary artery disease. They are the leading cause of morbidity and mortality in the developing world [25]. Bad habits and lifestyle in modern societies such as bad diets and smoking are the main risks factors that can cause of the disease. Prevention and, especially, an early and easy diagnose is important to avoid fatal consequences.

2.3. Cardiac medical imaging

Coronary angiography is the gold standard test for the diagnosis of CAD. It is an invasive procedure that uses X-rays to obtain images of the coronary arteries. It is done by introducing a catheter into the patient's arteries until it reaches the aorta. Then, a small amount of iodine is released as a contrast for the X-ray imaging, and coronary arteries are observed. Like any other invasive technique, it has associated certain risks such as bleeding or radiation. Nevertheless, these risks have decreased significantly in the last few years due to the advanced equipment design and the increase of experience by professionals [26].

Imaging is fundamental in cardiology since the heart is an internal structure and there is a huge need to evaluate it in a non-invasive way. Nowadays, in order to avoid invasive techniques such as coronary angiography, imaging is used not only as a diagnostic tool but also as a technique for selecting appropriate therapies for CAD patients. For example, the functional significance of a coronary artery stenosis needs to be determined to guide treatment. This is typically established through fractional flow reserve (FFR) measurement, performed during coronary angiography [27]. This measurement calculates the difference in pressure across the coronary arteries. However, imaging can be used to determine the FFR avoiding catheterism interventions. Thus, the professionals can treat the patient in a more accurate way.

With ongoing technological advancements, functional and anatomical information of the heart are provided by the current imaging modalities. Although there exists a lot of imaging modalities divided in functional, such as nuclear medicine; and anatomical, such as cardiac Magnetic Resonance Imaging (MRI) or echocardiography, Computed Tomography (CT) is the most used technique in the clinical environment in cardiology. A CT scan is an imaging technique used in radiology which its main goal in to obtain accurate images of the patients' body. It has a X-ray rotation tube and a row of detectors, located in the opposite side of the X-ray source, that measure X-ray attenuations by different tissues inside the body. As it rotates, X-ray images from different angles are obtained and then, they are combined using tomographic reconstructions techniques into 3D volumes.

The first line to diagnose CAD using imaging is Coronary Computed Tomography Angiography (CCTA). This technique is derived from CT and it uses an injection of iodine contrast material and CT scanning in order to evaluate if the coronary arteries have been narrowed. *Figure 3* presents how different stages of atherosclerosis are seen in CCTA images.

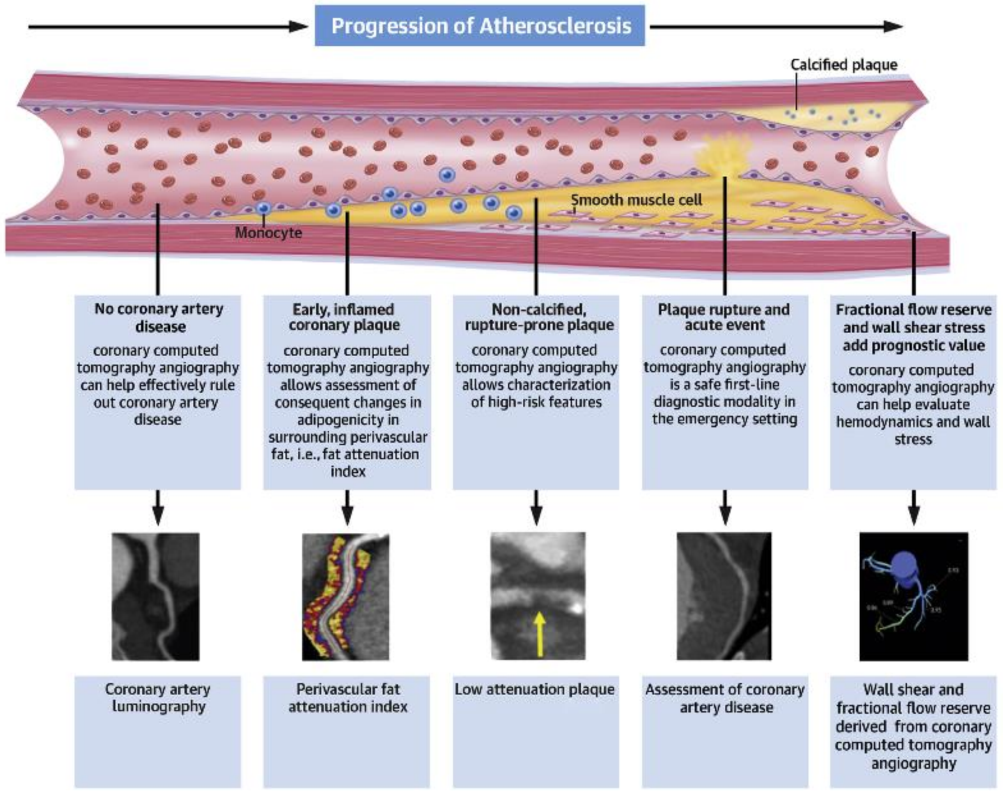

Progression of Atherosclerosis
Calcified plaque
Smooth muscle cell
Monocyte
No coronary artery disease
coronary computed tomography angiography can help effectively rule out coronary artery disease
Early, inflamed coronary plaque
coronary computed tomography angiography allows assessment of consequent changes in adipogenicity in surrounding perivascular fat, i.e., fat attenuation index
Non-calcified, rupture-prone plaque
coronary computed tomography angiography allows characterization of high-risk features
Plaque rupture and acute event
coronary computed tomography angiography is a safe first-line diagnostic modality in the emergency setting
Fractional flow reserve and wall shear stress add prognostic value
coronary computed tomography angiography can help evaluate hemodynamics and wall stress
Coronary artery luminography
Perivascular fat attenuation index
Low attenuation plaque
Assessment of coronary artery disease
Wall shear and fractional flow reserve derived from coronary computed tomography angiography

Figure 3. Utility of cardiac computed tomography in coronary artery disease. From Abdel Rahman, K.M. et al. [28].

Despite CT and CCTA scanning have some limitations such as radiation, its main advantage is that detailed images are created because of its huge sensibility and specificity [3]. These techniques are useful not only for the coronary anatomical information that they provide but also for the precise detection of the atherosclerosis stage, helping professionals in quantifying and characterizing the morphology and composition of the plaque and in making decisions for therapy [29]. New advances in CT are solving current limitations. For example, in the last years, scanning time and radiation exposure have been reduced [28].

2.4. Artificial Intelligence and deep learning

Artificial Intelligence (AI) is a branch of computer science that aims to simulate human intelligence in machines. It is able to learn from previous experiences, acquiring knowledge autonomously to work on problem-solving tasks. Machine Learning (ML) is a subset of AI that uses statistical learning algorithms to build models that can automatically learn and improve their performances from experience without being explicitly programmed [30]. ML algorithms can be classified into two main categories, supervised learning, in which the algorithm uses a known dataset to learn, and unsupervised learning, in which the model work on its own to discover information, without known data. Deep learning is a class of ML that uses multiple hidden layers to combine features progressively and automatically obtain predictions from a raw input data, which is usually a large dataset.

2.4.1. Deep learning configuration

In medical imaging, there exist several efficient deep learning-based algorithms that are able to learn features from input images. In supervised leaning, once we have collected our raw input images or data, it has to be labelled in order to extract features. This process is done manually and it is used afterwards as the ground truth by the computer to learn the features and modify its internal patterns. After that, the chosen deep learning architecture is trained. It means to feed our algorithm with data in order to improve its performance. The number of times that we pass all the data throughout the model is called epoch. Moreover, it is possible that all the dataset does not pass directly throughout the model. We can split the data into training groups to train the model by interactions. The number of samples in each iteration is called batch. After every epoch, the algorithm will adjust its parameters progressively according to the desired output, which is the data that has been labelled in the first steps of the workflow. For example, in a training with 6 batches and a dataset with 42 images the model will be trained in 7 iterations, taking the best parameters of every iterations. Once the algorithm is trained, it is tested with a different dataset to see if its performance is up to scratch for other situations and the error is computed. By adjusting the parameters, we can improve the performance of the model.

2.4.2. Data for deep learning

Data has an important role in deep learning-based algorithms. The goal of a model is to learn features from known examples, the training dataset, in order to generalize the same procedure on data the model has never seen, the test dataset. However, if a model does not have enough data, it is possible that it will fail learning deeply the features and the performance in the test dataset will be low. This event is called underfitting. On contrary, it is possible that a model perfectly learns the problem in the training dataset and it is not able to predict new outcomes, which is called overfitting. In order to avoid overfitting, the dataset is split into 3 parts: the training dataset, which is the one that the model uses to learn the problem and change weights inside the algorithm; the validation dataset, it is used to evaluate the model during the training phase and it modifies the weights of the network, so that hyperparameters can be tuned and the best performing model can be selected; and the test dataset, used to assess the performance of the model in a future and where the performance of the model can be observed.

2.4.3. Convolutional Neural Networks

As it has been said before, deep learning uses multiple layers to extract features progressively and automatically from a raw input. In image processing, lower layers may identify edges in images, while higher layers may identify the abstract concepts, in the case of this project, anatomical structures.

Despite the different deep learning-based architectures, Convolutional Neural Networks (CNNs) are the most used structure in medical images. There exist some structures based on CNN that are also used in medical imaging such Fully Convolutional Networks (FCN) [31], U-Nets [32], nnU-Nets [19], among others. The applications of these CNN-based architectures, especially in segmentation tasks, will be discussed in the following sections.

The architecture of a CNN was inspired by the neural connectivity pattern of the human brain [33]. It consists in several feedforward layers, in which output information in the one layer is the input of the following one, just as brain neurons usually work. The main aim of CNNs is taking images as inputs, assigning importance to various objects or aspects in the images in other to differentiate from others, and sending the results as output to another layer. As long as we go deeply into the architecture, more complex structures are detected. As *Figure 4* shows, there exist different types of CNNs layers, such as convolutional layers, the most typical ones, pooling layers, and fully connected layers.

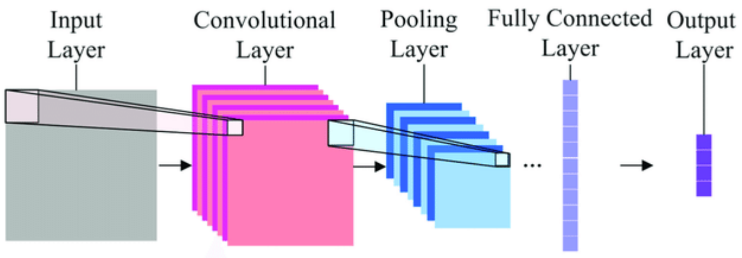


Figure 4. Example of a typical structure of Convolutional Neural Network. From Peng et al. [34]

Mathematically, convolutional layers have a matrix shape called kernel. Each matrix element, called weights, has a big contribution to model's predictive power. They are the ones that change in time in order to learn the desired output during training. The output of each layer is the result of the convolution operation between the kernel and the input image. *Figure 5* shows how the convolutional operation is done.

Apart from weights, convolutional layers have other important parameters such as the output bias. It is used to adjust the output along with the weighted sum of the inputs to the neuron and it is obtained after in every layer as an output and the activation function, which is a function that controls the amplitude of the output of the neuron, usually ranging between 0 and 1. *Equation 1* shows mathematically the output of a convolution layer, which is the result of applying the activation function (f) to the weights (w) and the sum of the inputs (x) plus the bias (b). The most used activation function is ReLU (Rectified Linear Unit).

$$z = f\ (b + \textstyle\sum_{i=1}^{n} x_i w_i) \tag{1}$$

During training, backpropagation is used to optimize the weights in the convolutional layers. It is a computational algorithm that is able to perform a backward pass while adjusting the weights and biases from the convolutional layers. The backpropagation algorithm uses the gradient descent function in order to calculate the gradients, or derivatives, of the loss function, which is the function responsible for measuring the error between the real and the expected outputs. Then, the existing parameters are updated in response to these gradients. All these functions and algorithms are defined mathematically in [35].

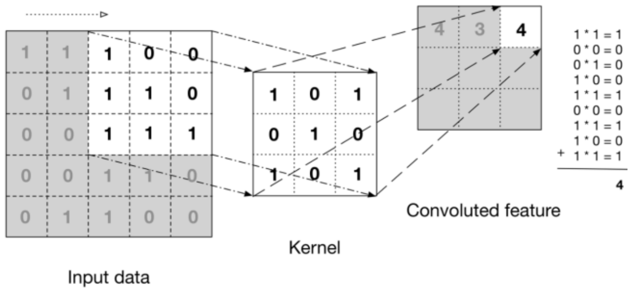


Figure 5. Simple representation of how each kernel in convolutional layers works. From Manav Mandal [36].

After training, there exist several functions in order to evaluate our model's performance: the loss function and the similarity metric such as the Dice score. While the information given by the first one is about the error between the desired and the obtained output, the second one compares the similarity between the ground truth and the obtained image, giving us information about how similar they are.

2.5. Cardiac imaging segmentation using deep learning

The actual procedure for CAD diagnosis from cardiac imaging is done at the radiology department in a hospital, the area that has all the equipment necessary to provide radiological images of the human body. The obtained images are used to make the predictions, diagnoses, and follow-ups of diseases that are otherwise difficult to assess. The usual pipeline for CAD diagnosis in a patient is the following:

i. First, the patient goes to the hospital, and the physician recommends him or her a particular radiological study, selecting the most appropriate imaging modality;
ii. After that, the study is performed, and the images are stored in the Picture Archiving and Communication System (PACS) of the hospital, a system for storing, distributing, and viewing medical images obtained from multiple modalities.
iii. Then, once the images are obtained, they are post-processed and analysed by the corresponding expert to make a diagnosis.

One of the clinical areas in which radiologic diagnostic imaging is fundamental is cardiology since the heart is an internal structure of the body, and it is necessary to evaluate it in a non-invasive way. Consequently, many hospitals have a unit dedicated exclusively to cardiac imaging. Depending on the imaging modality and the heart structure, the processing is carried out differently, tending to be manual and labour-intensive. Segmentation is one of the things that professionals can do in the post-processing part. It consists in the partition of the image into a set of relevant regions. An example of segmentation could be the division of the heart into its four chambers.

Cardiac image segmentation has numerous useful applications [1]. However, a manual segmentation of the heart structures from CCTA images, as in our case, is a huge complication not only due to the fact that many images is obtained in CCTA scan but also due to the limited time dedicated by professionals to a single patient. Moreover, manual segmentation processes are subjective and can present some errors that can be avoided.

To overcome these limitations, deep learning-based algorithms have become widely used for medical image segmentation in recent years. They have made a significant contribution to the automation of segmentation tasks. The deep learning algorithms are based on learning and knowing the location and limits of the different structures of the heart and other parts of the body. Recently, deep learning-based applications have provided efficient and effective ways to segment particular heart structures such as the four heart chambers and the coronary arteries for the three main imaging modalities in cardiology: MRI, CCTA, and ultrasounds facilitating a follow-up quantitative analysis of the cardiovascular structure and function [37].

For CAD diagnosis, deep learning-based segmentation algorithms from CCTA images have a wide range of applications. There exist two types of deep learning segmentation algorithms in the coronary arteries. The first one is based on the delimitation of the coronary arterial lumen in order to know if there are possible stenoses [4], which is basic for an early detection and treatment of coronary heart disease. On the other hand, the other algorithms segment and quantify non-calcified and calcified coronary plaques, useful to know the stage of the disease [38]. Other further practical applications of the algorithms are computational fluid dynamics (CFD) studies using numeric meshes or 3D anatomical models, among other applications.

Although the use of deep learning-based algorithms has grown exponentially and they are really practical in the clinical environment, there are several challenges that deep learning has to overcome if we want its implementation in the clinical routine in a proper way. For example, a successful validation of the models does not necessarily guarantee clinical implementation. Having not enough samples, a high training time or low image resolution can be a limitation. Moreover, labelling errors before training our model can affect the final behaviour of our models. Finally, the human factor in labelling can be misleading if the information provided by the model is not used in the right way by clinicians, with harmful consequences for the patient.

2.6. State of art. Applications of deep learning in coronary artery segmentation.

Over the past decades, the use of deep learning-based algorithms in medical image has had a significant contribution in the automation of segmentation tasks [4]. Moreover, the availability of public datasets and code repositories has encouraged research, improving the segmentation algorithms by the computing time, and improving their performance in order to help professional making decisions.

As it has been said before, quantitative analysis of coronary arteries is an important step for the diagnosis of cardiovascular diseases, stenosis grading, blood flow simulation and surgical planning [39]. Although this topic has been studied for years, only a small number of works investigate the use of deep learning in this context since the coronary artery are thin and sometimes the image resolution in these structures is low, so it can lead into some difficulties when the segmentation is done. In the following section, some of the most commonly used deep learning-based methods using CCTA images for coronary artery segmentation are reviewed.

A summary of these approaches is presented in *Table 1.* Methods relating to coronary artery segmentation will be mainly divided into two categories: coronary artery segmentation and coronary artery plaque segmentation.

APPLICATION	SELECTED WORKS	APPROACH	IMAGING MODALITY	STRUCTURE
CORONARY ARTERY SEGMENTATION	***Pre- and post-processing***			
	Gülsün et al. (2016) [12]	CNN	CCTA	Centreline
	Guo et al. (2016) [40]	FCN	CCTA	Centreline
	End-to-End Segmentation			
	Merkow et al. (2016) [41]	3D-CNN	CCTA	Vessels
	Moeskops et al. (2017) [13]	Multi-imaging CNN	CCTA	Vessels
	Shen et al. (2019) [42]	3D-FCN	CCTA	Vessels
	Li-Syuan et al. [43]	U-net	CCTA	Vessels
	Lee et al. (2019) [44]	Template Transformers Networks	CCTA	Vessels
CORONARY ARTERY CALCIUM AND PLAQUE SEGMENTATION	***Two steps segmentation***			
	Wolterink et al. (2016) [14]	CNN pairs	CCTA	Plaque
	Lessmann et al. (2018) [45]	Two consecutive CNNs	CT	Plaque
	Direct segmentation			
	Ronneberger et al. (2015) [32]	U-Net	CT	Plaque
	Huang et al. (2018) [46]	U-DenseNet	CT	Plaque
	Vos et al. (2019) [47]	ConvNet	CT	Plaque
	Cano-Espinosa et al. (2018) [48]	CNN	CT	Plaque

Table 1. Review of deep learning-based approaches for coronary artery segmentation using CT and CCTA images.

Regarding coronary artery segmentation, there exists a wide range of approaches to segment the lumen. Most of them use end-to-end CNN to obtain the blood vessel. Merkow et al. [41], presented a novel 3D-CNN called I2I-3D that only takes one minute to predict the boundary locations in a volumetric image. It presented significant results even in fine structures such as small blood vessels. CNN algorithms can be used in other types of imaging modalities. Moeskops et al. [13], proposed a multi-modal imaging segmentation framework where a single CNN can be trained to perform different segmentation tasks, including CCTA images. The CNN was able to identify the imaging modality and visualized anatomical structures and the tissue classes in different types of images. This approach is a useful tool for the future of medical imaging. Apart from the techniques mentioned above, there are other deep learning architectures based on CNN which are able to segment the coronary arteries such as the 3D Fully Convolutional Network (3D-FCN) which can learn from the 3D shape of the coronary arteries as prior knowledge [42] or CNN U-Nets which are a U-shaped deep learning architecture based on CNN which they are currently one of the main topics in deep learning research. CNN U-Nets have become quite popular due to their good performance in segmentation. A huge number of scientists such as Pan et al. [43], among others, are proposing U-Net-based network architectures to perform fully automatic segmentation of the coronary artery from CCTA images. Moreover, Lee et al. [44] presented template transformer networks, where a shape template is deformed to match the underlying structure of interest precisely and without artefacts.

Coronary extraction is a challenging task due to the presence of nearby cardiac structures and coronary veins as well as motion artefacts in CCTA [49]. Several deep learning-based algorithms use CNN as pre- and post-processing tools instead of using traditional methods [12][40], which were able to detect the centre line of the blood vessel, avoiding noise from the wall and other parts of the coronary tree.

Although their main function is not the segmentation of the coronary arteries, there exist some deep learning-based algorithms that have been important approaches in the diagnosis of CAD using CCTA images. For example, Candemir et al. [50], proposed a classification 3D-CNN whose main function was to detect discriminatory features between vessels with and without atherosclerosis. Its detection was quite accurate so it is a potentially useful application for helping professionals in making decisions. A similar discriminatory procedure was used by Du et al. [51]. Using a CNN, they proposed a method for the automatic detection of a lesion in coronary arteries. The model was able to distinguish automatically the difference between a lesion area and a normal vessel area [4]. The algorithm was able to detect stenosis lesions in 88% of the cases. Also, Zreik et al. [27] presented a method for automatic and non-invasive detection of patients with stenosis that require surgical treatment, employing a deep unsupervised analysis of complete coronary arteries in CCTA images by measuring the FFR measurement. On the other part, regarding the atherosclerotic plaque, some methods allow automatic classification of images according to the presence of calcified and non-calcified plaque, and in some cases, mixed plaque. Among these methods, some of which can be found in the review by Hampe et al. [52] or by Karim et al. [53]. In other works, Zhao et al. [54], and Zuluaga et al. [55] presented different support vector machine algorithms aimed to classify images depending on the plaque composition.

Concerning atherosclerotic plaque segmentation algorithms, Wolterink et al. [14] proposed an automatic coronary calcium scoring method in CCTA which was able to identify calcified voxels using paired CNN. The first CNN identified voxels likely to be calcified and the second CNN further distinguished between calcified and non-calcified voxels more accurately. Similar to this, a two-stage scheme, Lessmann et al. [45] proposed a method for automatic detection of coronary artery in low-dose chest CT using two consecutive convolutional neural networks. The first network identified and labelled potential calcifications according to their anatomical location and the second network identified and quantified true calcifications among the detected candidates. More recently, several approaches have been proposed for a direct segmentation of calcium plaques from non-contrast cardiac CT or chest CT using only a single CNN, which less training time is needed. The majority of them employed combinations of U-net and DenseNet [46] for the precise quantification of calcium plaques. These approaches follow the same workflow as the paired ones where calcium plaque is first identified and then quantified. An alternative, effective and promising approach is an intermediate segmentation and a direct quantification, such as Vos et al. [47] and Cano-Espinosa et al. [48] proposed in their works.

Nowadays, the most emergent and promising deep-learning architecture for medical image segmentation are UNETRs (U-Net Transformers). They are possible candidates to overcome other architectures' limitations currently used in medical imaging.

This type of architecture was recently used in Natural Language Processing because it is able to learn, by paying attention, the contextual information of the input volume and predict new inputs. It utilizes a transformer as the encoder to learn sequence representations of the input volume and capture the global information and a CNN-based decoder where image resolution is increased in order to compute the final segmentation output.

Hatamizadeh et al. [15] showed that UNETRs have a good performance in segmentation and they can be useful not only for their low training time and its good capacity for they learning long-range dependencies, but also because they can work in multimodal images. In our case, they can provide an automated segmentation of the coronary arteries for CAD diagnosis in CT images. UNETRs will be discussed in the following sections.

3. MARKET ANALYSIS

3.1. Addressed sector

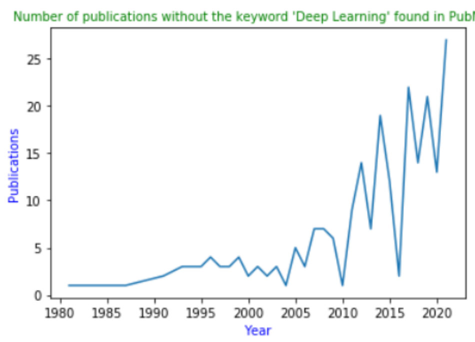
Coronary heart disease is a serious disease that endangers human health and life. It is the leading cause of death worldwide among men and women over 20 years old, causing one-third of deaths worldwide each year [56]. An estimated 3.8 million men and 3.4 million women die each year from CAD, which is almost the 20% of the total deaths in men and 1 in 6 women. In recent years, the incidence and mortality of CAD have increased rapidly due to current lifestyle in modern societies. It is estimated that 80-90% of people dying from CAD have one or more major risk factors that are influenced by lifestyle, such as diet, smoking, hypertension, diabetes mellitus, among others [57]. An early diagnosis is important to reduce mortality.

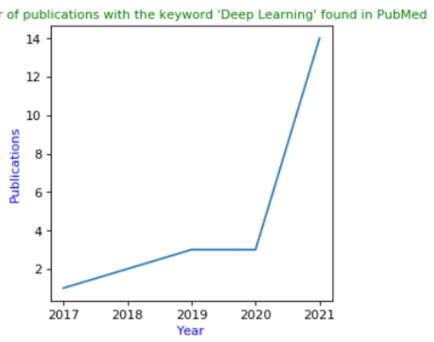
The current algorithms used in cardiac imaging avoid current invasive techniques such as cardiac interventional catheterism as well as manual and time-consuming tasks in the diagnosis of CAD such as image segmentation. On the other hand, the algorithms used in current AI-based software used for coronary artery segmentation are black boxes so novel developed algorithms and other functionalities cannot be integrated in the same software if they are not made by the software provider.

The Transformer Neural Network-based algorithm's for coronary artery segmentation main target sector is 18.2 million adults aged 20 and older that have CAD, which means about 6.7% of the worldwide population [56]. Consequently, the physicians that will use Transformer Neural Network-based algorithms to diagnose these patients and the hospitals and health systems as well as other medical companies will be also part of the main target.

3.2. Historical evolution of the market

As it was mentioned in the previous section, there exist a vast need to develop an automatic segmentation tool to diagnose CAD. In the last few years, several algorithms have been used to automatically segment the coronary vessels. First, algorithms based in statistical methods were used in different cardiac imaging modalities to divide the heart into its main anatomical structures semi or fully automatically [6]–[11], [39]. Then, the application of AI-based algorithms in cardiac imaging aroused ample interest due to their good performance and accuracy in large datasets. They were could be used in the daily physicians' routine, enabling a quick and non-invasive diagnose in CAD patients.





Figures 6 and 7. Number of publications found in PubMed in recent years with the keywords "segmentation" AND "coronary arteries" with and without the keyword "deep learning".

As it is shown in *Figure 6,* the number of publications about the segmentation of the coronary arteries in PubMed [58] has increased exponentially in the last few years due to its clinical importance and the amount of research done in AI. In addition, *Figure 7* shows the number of articles about deep learning-based algorithms used for CAD assessment. Even though the low number of algorithms, the large annual number of deaths shows a real need to incorporate into the market in order to provide an easy and fast diagnosis of CAD patients. However, these algorithms require abundant data and its corresponding ground truth data, which is a segmentation done manually by physicians to predict the final segmentation, as there are not any public available datasets to use for AI coronary arteries algorithms research. In the recent years, some advances have been done such as the Automatic Segmentation of the Coronary Arteries (ASOCA) Challenge [59] in 2020, in which the main aim of the challenge was to automatically segment the coronary artery lumen, not including calcified regions or other diseases, in order to provide a public dataset of the coronary arteries segmentation and enhance research.

The current software used in the clinical practice to segment the coronary artery from CCTA images, among other pre- and post-processing tools, are the ones provided by the CT scanners vendors such as Vitrea by Toshiba Canon [60] or Phillips Intellispace Portal [61], being the latter the one available at Hospital de Sant Pau for coronary artery segmentation. However, those software are a black box which do not allow for knowing the actual functioning of the algorithms. Moreover, the mentioned software are semi-automatic, so not only radiologist but also physicians need to check and intervene in a single patient CCTA image. Their performance can also be improved. Other recent coronary artery segmentation software developed by big companies are Acquisition and Reconstruction Techniques for Coronary CT Angiography software by Philips Healthcare Scanner Platforms [62] or the CardIQ Fusion developed by General Electric Healthcare [63].

In recent years, several AI-based software have been developed to help physicians in the CAD diagnostic in an early stage of the disease and in a non-invasive way. Some examples of these software are Caristo [64], HeartFlow [65] and Cleerly [66]. The former is able to predict the heart attack risk by detecting invisible coronary inflammation in routine CCTA images, which is the key process that drives the development of CAD but it is not detectable with current routine diagnostic tests. This software is not able to segment the coronary artery lumen, it only gives information about the pericoronary fat. On the other hand, HeartFlow is currently used in cardiac diagnostic testing. It consists of CFD-based algorithms that are able to create a colour-coded 3D model of the patient’s coronary arteries from a CCTA image, providing not only anatomical information but also functional information about the artery blockage using CFD. Cleerly can identify, characterize and quantify plaque build-up from CCTA images using AI to support physicians in determining a patient’s risk of a heart attack and developing a treatment plan to improve heart health.

None of the competitors mentioned above is integrated into the hospital’s clinical workflow, except Heartflow. Thus, physicians need to send CCTA images to the providers to segment the image, increasing the diagnosis time. Incorporating an automatic and integrated tool into the clinical workflow will not only provide better outcomes to patients but also professionals will spend less time with every single patient. Deep learning-based algorithms can solve the current bottleneck, specially state of the art ones such as Transformer Neural Networks.

3.3. Future perspectives

In light of the above, it is clear that research has to be done for the assessment in CAD patients due to the high mortality and morbidity in the whole world. There is a huge need to find new and better methods better than the current ones for the disease's diagnosis and follow-up. These methods should be fast, non-invasive, and precise. Segmentation of the coronary arteries from multimodality images, specially CCTA images since they are the ones allowing for a better assessment of CAD, provides valuable information qualitatively and quantitatively and overcomes all the limitations that the current methods present.

The market opportunity is just massive. In Europe, millions of people are at risk of developing a heart attack [67]. This prevalence of the disease in European population is increasing year after year, due to population ageing. Therefore, an early diagnosis is completely necessary not only for reducing fatal consequences in the population such as deaths but also for reducing costs [68].

Overall CAD is estimated to cost the EU economy 210 billion euros a year [67]. One of the main causes is that mainly all the CAD diagnoses are done at a late stage of the disease. Many patients do not end up getting images of their heart and arteries until they have pain or shortness of breath, which makes diagnosis and treatment reactive instead of proactive [69]. According to the Agency for Healthcare Research and Quality, a hospital stay due to a heart attack costs around 18.000€ at an average length of 5.3 days a hospital in Europe [70]. This cost can be increased to 88.000€ when some surgical procedures are involved. If CAD were detected in an early stage of the disease, these costs would not exist.

In addition, James K. Min, the CEO of Cleerly, in 2019, showed in a randomised control trial that two-thirds of the patients who are referred for an angiography did not have the disease [69]. In Spain, an angiography costs around 7.500€ while a CCTA only cost around 300 and 650€ [71]. Introducing AI-based algorithms able to segment the coronary artery and detect if there is a CAD as diagnostic tests in the clinical environment can be an ideal alternative to the current diagnostic methods and will reduce not only the costs but also the intervention risks, such as bleedings [26].

AI-based algorithms in segmentation tasks not only provide high diagnostic information for preventing heart attacks but, also, can be an outstanding clinical decision-making tool. For example, they can help professionals to decide the most optimal medical treatment for CAD patients. In this way, not only unnecessary costs will be reduced but also potentially harmful side effects in people who will not benefit from some specific treatments.

There is a huge demand in CAD market that deep learning-based algorithms can supply perfectly. They are also a significant opportunity to accelerate innovation in cardiology revolutionizing the current diagnostic tests in the cardiac clinical routine such as mammograms and colonoscopies revolutionized some years ago the early detection of breast and colon cancer, respectively.

However, their implementation in the clinical routine is not always guaranteed since the fatal consequences for the patient that the model can cause if it is not well used by the clinicians. Thus, there is a huge need to find an optimal algorithm with a good performance to overcome the ongoing limitations in which the physicians can trust. Transformers Neural Network architectures may offer a satisfactory outcome for professionals and can be integrated into the clinical workflow of a hospital.

4. CONCEPTION ENGINEERING

The workflow followed in this project is the one presented in *Figure 8.* Some decisions were needed to be taken before the execution of the project such as the segmentation AI-based algorithms and the method used for the hyperparameters configurations. They will be explained in detail in the following section.

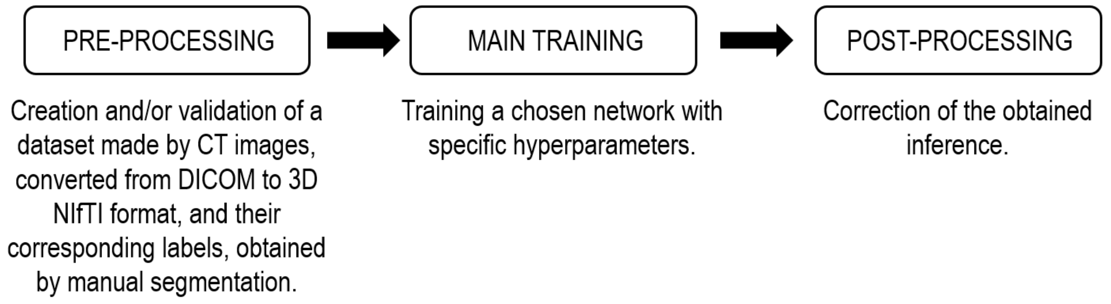

PRE-PROCESSING
MAIN TRAINING
POST-PROCESSING
Creation and/or validation of a dataset made by CT images, converted from DICOM to 3D NIfTI format, and their corresponding labels, obtained by manual segmentation.
Training a chosen network with specific hyperparameters.
Correction of the obtained inference.

Figure 8. Workflow executed during the project.

4.1. Possible segmentation AI-based algorithms

Several AI-based algorithms for the coronary artery segmentation from CCTA images will be discussed in the following section.

4.1.1. U-Net

The U-Net is a convolutional network architecture for biomedical image segmentation. It is considered one of the most successful architectures in the medical domain [72]. The U-Net consists of a contracting path, called encoder, and an expansive path, called decoder [73].

The encoder is just a traditional convolutional neural network. Each layer consists of the repeated application of two 3x3 convolutions, each followed by a ReLU and a 2x2 max-pooling operation, which is an operation that is used to downscale the image, extracting the most important features [74], as shown in *Figure 9*. The max pooling layers are able to extract the maximum value from the input matrix, taking as output its most prominent features. At each downsampling step, the number of feature channels doubles [75].

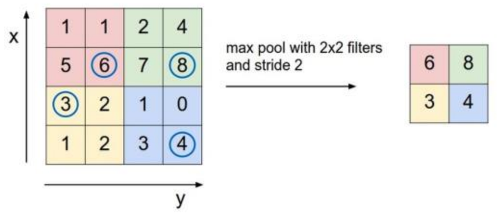


Figure 9. Example of a 2x2 max pooling layer. From Martin Jan Musiol [76].

The expansive path or decoder consists of repeated application of two 3x3 convolutions, each followed by a ReLU and a 2x2 up-convolution layer, that can halve the number of feature channels. The cropping is necessary due to the loss of border pixels in every convolution. 23 convolution layers will be done at the end of a 5-layer-deep U-Net [32]. In *Figure 10* an overview of a U-Net is represented.

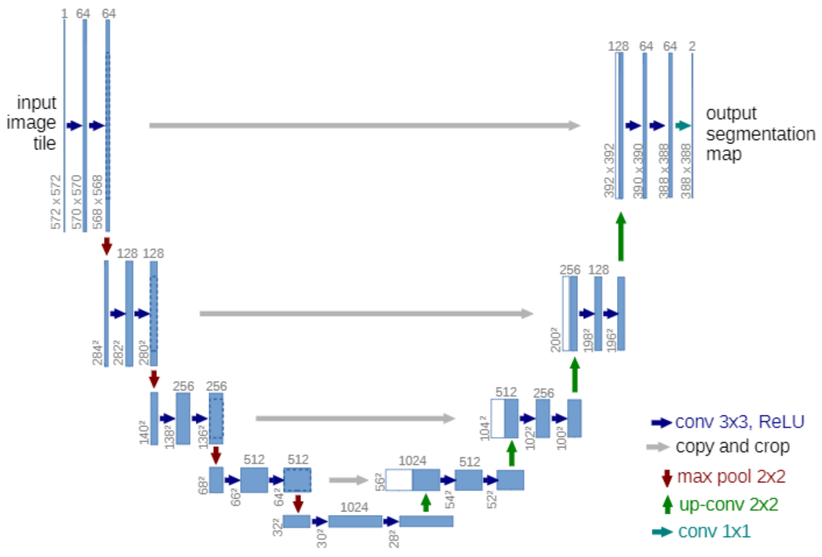


Figure 10. Example of a U-Net architecture. From Ronneberger et al. [32].

More recently, a new segmentation framework, the nnU-Net, has become a very popular choice for 3D medical image segmentation tasks [19]. It is able to configure the initial hyperparameters automatically.

The main advantages and drawbacks of U-Net are presented in *Table 2* [77].

ADVANTAGES	DISADVANTAGES
Few training images Accurate segmentation in medical multimodal images Input image size is not relevant Global and localized information caption High sensitivity	Large GPU memory for larger images High computational time

Table 2. U-net main advantages and disadvantages.

4.1.2. UNETR

In late 2021, Hatamizadeh et al. [15] introduced UNETR, a novel and powerful medical image segmentation tool. The deep learning based-architecture is formed by a transformer that works as an encoder able to learn sequential information from the embedded input volume patches and effectively captures the global information followed by a U-shaped CNN-based decoder which predicts the segmentation outputs. It uses 3D volumetric data as an input and a segmentation of this data is obtained at the end as an output. *Figure 11* shows an overview of the UNETR architecture proposed by Hatamizadeh et al. In the following section, the transformer encoder and the U-shaped decoder will be explained in detail.

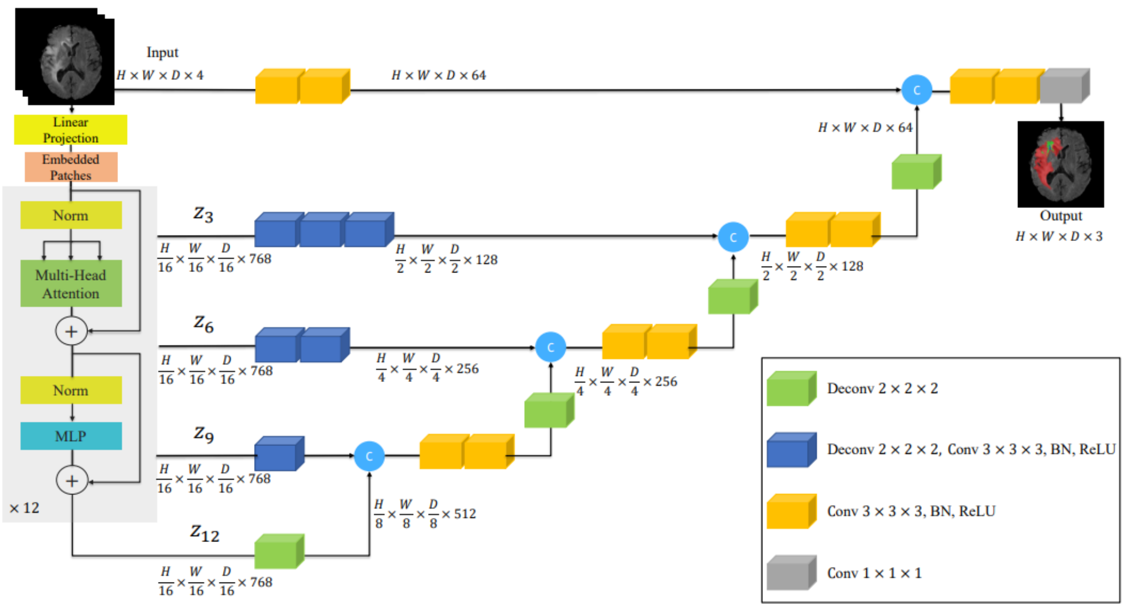

Input
$H \times W \times D \times 4$
Linear Projection
Embedded Patches
Norm
Multi-Head Attention
Norm
MLP
$\times 12$
z_3
z_6
z_9
z_{12}
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$H \times W \times D \times 64$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 256$
$\frac{H}{8} \times \frac{W}{8} \times \frac{D}{8} \times 512$
Output
$H \times W \times D \times 3$
Deconv $2 \times 2 \times 2$
Deconv $2 \times 2 \times 2$, Conv $3 \times 3 \times 3$, BN, ReLU
Conv $3 \times 3 \times 3$, BN, ReLU
Conv $1 \times 1 \times 1$

Figure 11. Example of a UNETR architecture. From Hatamizadeh et al. [15].

Transformers as encoders

In digital image processing and visual computing, encoders are able to extract features from an image, reducing the image in weight and width. In this way, the image can be modified easily without losing information and resolution, being able to operate in high speed and accuracy and without high memory storage systems [78]. In the case of segmentation networks, the decoder is the one that is able to convert the modified code back to a segmentation mapping.

In our case, the encoder is the responsible of learning global contextual representations by gradually down sampling the extracted features of the volumetric input converted into a 1D-sequence vector. The main advantage of using transformers as encoders is that they utilise self-attention modules to learn a global long-range information, highlighting the most importance sequence or features in the input sequence [79].

Originally, transformers were used in Natural Language Processing. As they are able to progressively highlight the important words in word sequences, they can learn long-range dependencies in sentences and capturing its global meaning [16]. In image processing, they are able to understand correlation between pixels in different volumetric image patches.

Transformers as encoders in UNETR first create a 1D sequence of a 3D input volume with certain resolution by dividing it into flattened uniform non-overlapping patches. Subsequently, they project the flattened patches into an embedding space, which is a low-dimensional space where high-dimensional vectors are translated in order to train the algorithm easily. After a normalization layer, embedded vectors go into the transformer block which it is formed by a specific number of multi-head self-attention sublayers (MSA) followed by multilayer perceptron (MLP).

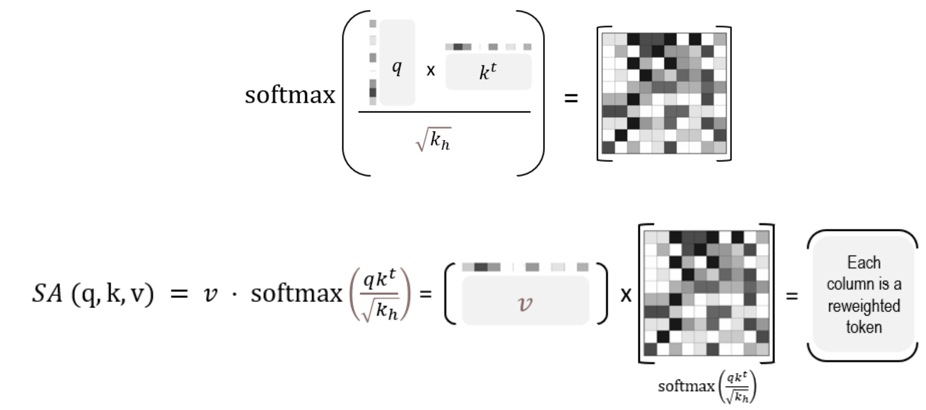

softmax
q
x
k^t
$\sqrt{k_h}$
=
$SA\ (\mathrm{q,k,v})\ =\ v\ \cdot\ \mathrm{softmax}\left(\frac{qk^t}{\sqrt{k_h}}\right)$ =
v
x
=
Each column is a reweighted token
$\mathrm{softmax}\left(\frac{qk^t}{\sqrt{k_h}}\right)$

Figure 12. Graphical representation of the SA block inside the transformer.

The MSA sublayer, specially, the Self-Attention (SA) block, the main part of the MSA sublayer, is a function that learn the mapping between the query (q), pixel intensity, and the corresponding key (k), intensities in other pixels in the input patch, and value (v), division between the query by every key in the embedded input sequence [80]. The function is shown in *Equation 2*, which is a simple multiplication of a matrices, specifically, the value v matrix and the result of the division of the query q per the transposed key k by the squared root of k_h, which is the key k value divided by n, a scaling factor for maintaining the number of parameters to a constant value with different values of the key k. The SA block is graphically represented in *Figure 12*.

$$SA(q,k,v)\ =\ v\ \cdot\ \mathrm{softmax}\left(\frac{qk^t}{\sqrt{k_h}}\right) \qquad (2)$$

In this way we obtain a matrix with the score of every combination of a pixel with other pixels in the embedded input sequence, which it will be useful to train the network in a future. Furthermore, the output of the MSA is defined in *Equation 3*, where ω are the trainable parameters weights.

$$MSA(q,k,v)\ =\ [SA_1(q,k,v);\ SA_n(q,k,v);\ \ldots;\ SA_n(q,k,v);]W_{MSA} \qquad (3)$$

Afterwards, the output of the MSA goes throughout a normalization layer followed by the MLP, which is a fully connected layer that is able to transform any input dimension to the desired dimension. In this way, we convert the output MSA matrix into a 1D sequence again, repeating the encoding process.

The transformer encoder is directly connected to a decoder via skip connections at different resolutions to compute the final segmentation output [15]. This means, that the U-shaped decoder recollects patches, with different resolution as shown in *Figure 11*, in different points of process done by the encoder. On that way, spatial information that is lost during down sampling is recovered.

<u>*U-Shaped CNN-based network as decoders*</u>

Despite the great capability of learning global information, one of the current problems of transformers is that they are unable to properly capture localized information. Therefore, an insufficient extraction of micro coronary arteries feature information and low sensitivity are caused by predicting pixels as background and not as coronary arteries [81]. That is the main reason about why a U-net is used as a decoder [15].

The main function of the U-shaped CNN-based decoder is up-sampling the extracted information at different points of the transformer in order to predict further pixel input patches and to obtain the final segmentation output by increasing the image resolution. In *Table 3*, the main advantages, and drawbacks of UNETR are presented.

ADVANTAGES	**DISADVANTAGES**
Few training images Accurate segmentation in medical multimodal images Long range pixel dependencies knowledge High resolution Relatively low GPU memory and low computational time	Insufficient localized features extraction

Table 3. UNETR main advantages and disadvantages.

4.2. Proposed segmentation AI-based algorithms

While U-Net architectures are better at capturing local pixel information in input patches, a UNETR architecture was chosen among U-Net because of its good capacity for learning long-range dependencies in multimodal images, improving the sensitivity and accuracy of the algorithm, and performing a precise automatic segmentation of the coronary arteries. Moreover, the main advantage which UNETRs were mainly chosen was because of their low computational time and relatively low storage GPU memory. Therefore, the entire training was able to be performed in the hospital, enabling the algorithm integration in the current hospital image segmentation workflow in the future. UNETR is also able to work well with small training sets as U-Net, thanks to the robustness provided with data augmentation in the U-shaped decoder.

4.3. Possible evaluation solutions

Different methodologies can be used for the tunning of the hyperparameters for the chosen AI-based model architecture. By tunning the hyperparameters, the most optimal architecture can be obtained and the performance of the model can be increased. The possible hyperparameter tunning options will be discussed in the following section.

4.3.1. Self-configuring method

Self-configuring methods consist of tunning manually the hyperparameters of the UNETR by changing the main hyperparameters of the network. They include the learning rate, the batch size, the patch size, and the pixel resolution of the volumetric patches in the transformer encoder as well as the percentages of training, testing, and validation datasets. The hyperparameter value can be decided by looking at the literature of similar studies or by looking at the performance of the model in previous trainings. As they are tuned manually, this technique is not only slow but also error-prone.

4.3.2. Ablation study

An ablation study has its roots in the field of experimental neuropsychology, where parts of animals' brains were removed to study the effect that they had on behaviour [82]. An ablation study in an AI algorithm consists in studying the performance of certain hyperparameters and their contribution of them to the overall model by changing certain components. Therefore, a better understanding of the network's behaviour is gained.

In our work, the contribution to the model of three hyperparameters were studied: the patch size, the batch size, and the learning rate. These were chosen because they were the ones that contribute most to the UNETR network according to Hatamizadeh et al. [15]. Moreover, resolution was the only parameter that remained constant in the ablation study since voxel size variation was proven not to be useful in Acebes' work [17]. One of them was changed while two hyperparameters maintained constant. Different combinations were obtained in order to understand the model's behaviour. Subsequently, the final model was tuned with the hyperparameters that work best for the model's performance.

4.4. Proposed evaluation solution

The ablation study was the evaluation model that was chosen. It is less error-prone than the self-configuration model. The hyperparameters were tuned understanding how they contribute on the performance in the model not as the manual tuning evaluation methods.

5. DETAILED ENGINEERING

In the following section, all the needed details to perform the complete the coronary artery segmentation algorithm using Transformer Neural Networks will be described in detail.

5.1. Dataset

The dataset was made of CCTA images and their corresponding labels, which are binary masks of the coronary artery lumen and the aorta artery. It was provided by the Cardiac Imaging Department in Hospital de la Santa Creu i Sant Pau in Barcelona in order to train the chosen neural network. The dataset used for this project was made from scratch in 2020 by César Acebes [17] in order to train the U-Net network under a nnU-Net framework he used for his Master's Thesis since there was no freely accessible dataset. *Figure 13* shows the visualization of a database CT image and its corresponding label, highlighted in red.

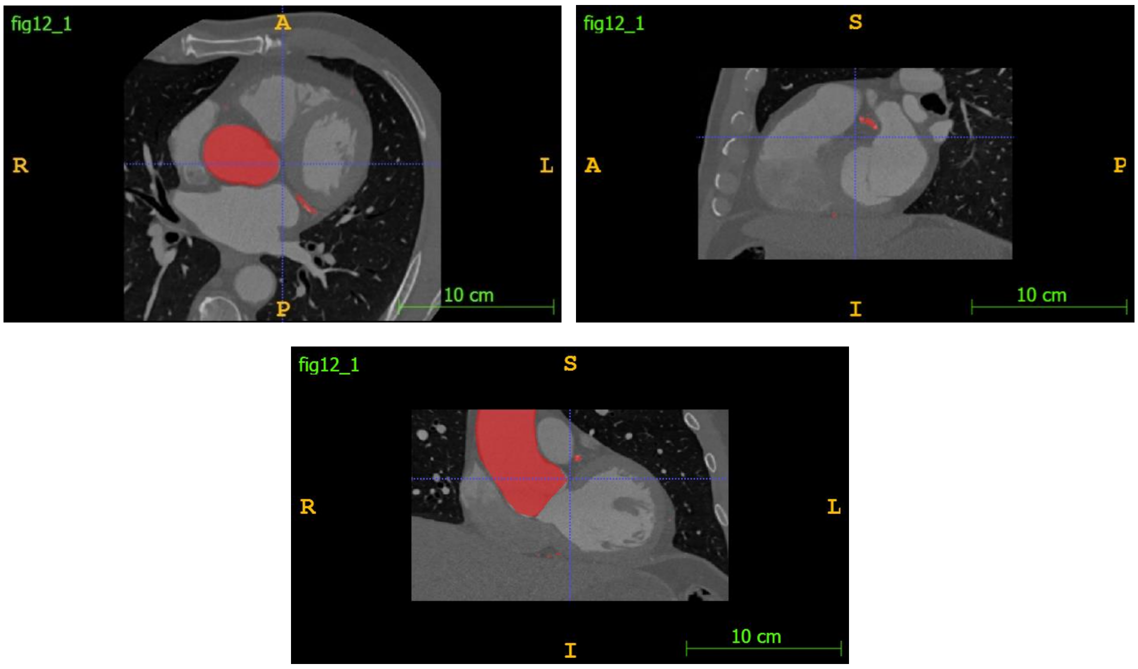


Figure 13. CCTA slice with the coronary artery and the aorta highlighted in red in the axial, sagittal and coronal planes, respectively.

The mentioned dataset was created following the next steps. First and foremost, the images were selected. Forty images from different patients were obtained using contrasted CCTA at around 75% of the R-R interval of an electrocardiogram (ECG), which corresponds to the mid-end of the diastole phase. Due to the differences in the anatomy of the patients, each image had a different number of slices and spacing among them. However, the slice size, which is the number of voxels that every slice contains, was 512x512 in every image. These images were obtained from 2006 to 2015 from two different manufacturers: Toshiba Medical Systems [83], which currently belongs to Canon Medical Systems Corporation, and Philips Healthcare [84]. Images from both manufacturers were used because of the fact that Hospital de Sant Pau works not only in its own patients' image but also the ones from Creu Blanca Clinic. In this way, the segmentation tool could be used in both hospitals. However, the generalization of the deep learning-based algorithm would be lower, according to recent studies [85].

Then, the selected images were kept in a DICOM format in the Hospital's PACS. From there, they were uploaded by a cardiologist to a visualization platform called Philips Intellispace Portal [61], where a cardiologist displayed the images one by one the images, discarded major artefacts ensuring maximum quality, and used the automatic segmentation tool of the platform, called CT Comprehensive Cardiac Analysis [86], to perform an automated segmentation of the main cardiac structures. Finally, a manual correction was done to the obtained segmentations by the software and they were exported in STL format.

The final step was converting the images and the labels to a format that the deep learning-based segmentation tool can read. Therefore, DICOM CT images and STL labels were both converted to compressed 3D NIfTI files (.nii.gz) using 3D slicer software [87]. This format also allows the anonymization of the data, avoiding any data protection problems.

5.1.1. Extended dataset

Supervised deep learning-based algorithms require a large quantity and high-quality training data to achieve accurate results [88]. To extend the dataset that the Cardiac Imaging Department created in the past year and to perform a more relevant and significant project, 200 patient images were selected. These images were obtained using contrasted CCTA at around 75% of the R-R interval by the Philips iCT 256 CT scanner from April 2021 to June 2021. Then, a cardiac imaging expert analysed and segmented the images one by one using the aforementioned segmentation tool included in Philips' software.

Afterwards, the segmentations were validated and manually corrected with Materialise Mimics software [89] version 24.0, which is integrated into the Materialise Innovation Suite. However, it also could have been done with other free software such as 3D Slicer. Finally, the CT images and the labels were converted into 3D NIfTI files.

5.1.2. Final dataset

The initial idea was using 240 images for the segmentation algorithm training, obtained from both datasets, the previous and the extended one. Thus, the algorithm would be able to segment the images in a more generalised and accurate way. However, several problems led to develop the final training of the project with only the original dataset. The first problem which occurred was that some slices in 8 images from the original dataset were missing. Therefore, once the segmentation algorithm was pre-processing the data before the training, those images could not be processed. This led to some errors during the training. As a consequence, these 8 images were not considered in the final dataset.

Another problem was that the coordinates between the labels and images 3D NIfTI files did not match due to an error when the labels were converted into the mentioned format. The lack of time at the end of the project entailed to discard the extended dataset images for training purposes. However, some of these images were used in the algorithm evaluation. In *Table 4*, the main information about the final dataset is reviewed.

Manufacturer	**Toshiba-Canon**	**Philips**
Institution	Creu Blanca Clinic	Hospital de Sant Pau
CT Scanner Model	Toshiba Aquilone ONE	Philips iCT 256
Acquisition date interval	January 2006 – April 2013	May 2010 – August 2015
Total data	7	25

Table 4. Review of deep learning-based approaches for coronary artery segmentation using CT and CCTA images.

5.2. UNETR

The training was performed by using SALMON [90], which is an open-source computational toolbox for medical imaging segmentation developed by the Centre for Medical Physics and Biomedical Engineering in the Medical University of Vienna. The toolbox include several segmentation deep learning-based algorithms, U-net and UNETR, based on MONAI 0.7.0 [91], an open-source PyTorch-based framework for deep learning in healthcare images. The MONAI codebase allows a flexible pre-processing for multi-dimensional medical imaging data, an easy integration of deep learning-based algorithms in existing workflows, as well as a domain-specific implementations for networks, losses, evaluation metrics, among others. In this section the main used Python scripts as well as their function will be described. These scripts were implemented in a virtual environment which used Python 3.8. The code is presented in the Annex.

5.2.1. organize_folder_structure.py

The main function of this Python script was to organize the data in the folder structure (training, validation, and testing) needed for the network. It also resamples and resizes to a previously fixed resolution value all the labels to their corresponding CT image in order to avoid array size conflicts. In order to execute this command, it is important to structure the data set into two main folders, one with all the images and one with all the labels, inside a folder called “Data_folder”.

5.2.2. init.py

In this command all the basic parameters are defined such as the name of the folders as well as the network hyperparameters, which are the number of input and output channels, the batch size, the patch size, the number of epochs, the number of SA blocks of the UNETR, the defined learning rate, among others. It is important to overwrite this script before every training in order to set the parameters and start the training on the correct data.

5.2.3. train.py

The main objective of this Python script is to run the training. After the first epoch, a folder called “runs” is created with all the weights of the training. Tensorboard, a Pytorch tool that provides measurement and visualizations during a deep learning workflow, is available to monitor the training by executing in the terminal the following command: `tensorboard –logdir .\runs`. It enables tracking the training metrics like the algorithm loss and Dice, also known as the bias, over time, as well as, displaying a gif with the image inference, which is the resulting binary mask after the coronary artery segmentation in every CT slice. This script needs the *networks.py* script, where the UNETR architecture is coded for the medical imaging segmentation. It is not necessary to run the last script in the terminal but they have to be in the same directory.

5.2.4. predict_single_image.py

This Python script is used to obtain the inference of a new image, which it is a binary mask of the coronary arteries. It can have different size and resolution from the training. The image will be resampled and the output will be the binary mask in a NIfTI format with the same size and resolution of the source image.

5.3. Hyperparameters

In the following section the process to choose the segmentation algorithm hyperparameters as well as the ones used in the final training will be discussed.

5.3.1. Ablation study results

As it was mentioned in the previous section, an ablation study was done to study the performance of the network hyperparameters and their contribution to the overall model by changing certain components. The combinations of the different patch and batch sizes as well as the learning rate were studied in a 150 epochs training with the 32 images final dataset in order to select the hyperparameters that fit best to the algorithm. The 32 images dataset was chosen to compute short trainings. The batch size has to be multiple of the number of images that the dataset has. Thus, almost all the possible multiples of 32 were tried in the ablation study. The patch size was 16 or 32 since these two were the most optimal according to Hatamizadeh et al. research [15]. The values of the learning rate are different multiples of 10 on the logarithmic scale. The ablation study results are shown in this section.

In *Table 5* the validation Dice score is shown, which is the metric score that compares the similarity between the ground truth and the obtained image in the validation dataset, giving us information about how similar they are. This parameter is important to avoid overfitting and it has different values depending on the patch and batch size and the learning rate (lr) chosen for every 150 epochs training. The dash means that the training was not able to be computed because the GPU was run out of memory, which is also known as Compute Unified Device Architecture (CUDA) error.

		PATCH SIZE: 16		**PATCH SIZE 32**	
	lr	**0.01**	**0.1**	**0.01**	**0.1**
BATCH SIZE	**1**	0.3954	0.4202	0.4845	0.4693
	2	0.4605	0.4008	0.46	0.519
	3	0.3845	-	0.5007	0.4336
	4	0.2982	0.3618	-	-

Table 5. Validation Dice score depending on the learning rate (lr), patch and batch size obtained in the performed ablation study. The dashes are CUDA error trainings.

Table 6 shows the testing Dice score, which is the same metric as the one explained before but in the test dataset. It gives us information about the performance of the algorithm in images that it has never seen before

		PATCH SIZE: 16		**PATCH SIZE 32**	
	lr	**0.01**	**0.1**	**0.01**	**0.1**
BATCH SIZE	**1**	0.1645	0.1431	0.2235	0.2073
	2	0.1811	0.1462	0.1897	0.2251
	3	0.1405	-	0.2238	0.1851
	4	0.109	0.1249	-	-

Table 6. Testing Dice score depending on the learning rate (lr), patch and batch size obtained in the performed ablation study. The dashes are CUDA error trainings.

The best validation Dice show the highest Dice validation metric and the epoch where it was obtained. This value can give us an idea about which are the best hyperparameters. *Table 7* shows all the best validation Dices obtained in the different trainings done during the ablation study. The epoch in which they were obtained is in braces.

		PATCH SIZE: 16		PATCH SIZE 32	
	lr	**0.01**	**0.1**	**0.01**	**0.1**
BATCH SIZE	**1**	0.4776 (139)	0.4851 (148)	0.5622 (118)	0.5944 (140)
	2	0.4625 (136)	0.4325 (118)	0.5205 (149)	0.5513 (142)
	3	0.4367 (149)	-	0.5007 (150)	0.4343 (149)
	4	0.3474 (143)	0.3618 (149)	-	-

Table 7. Best validation Dice score depending on the learning rate (lr), patch and batch size obtained in the performed ablation study. The epoch when each one was obtained is in braces. The dashes are CUDA error trainings.

To evaluate the algorithm, the computational time was also calculated, which is the time that the 150 epochs training needed. It is shown in *Table 8*.

		PATCH SIZE: 16		PATCH SIZE 32	
	lr	**0.01**	**0.1**	**0.01**	**0.1**
BATCH SIZE	**1**	19.68	19.47	19.47	19.2
	2	19.24	19.47	19.77	19.46
	3	19.16	-	19.5	19.35
	4	19.16	19.55	-	-

Table 8. Training computational time (in hours) depending on the learning rate (lr), patch and batch size obtained in the performed ablation study. The dashes are CUDA error trainings.

5.3.2. Final chosen hyperparameters

The final hyperparameters were chosen by looking at the metrics obtained in the ablation study as well as analysing the image inferences in Tensorboard, which is the binary mask created by the passing images that were not seen, the test dataset, by the network throughout the model. The computational time was not taken into consideration because the difference between each ablation study configuration was not significant. A 150-epoch training was always around 19 and 20 hours.

First, some hyperparameter configurations were rejected by looking at the metrics such as a testing and validation Dice score along with the best validation Dice score. We came up with the conclusion that a patch size of 32 is better than a patch size of 16. Moreover, the smaller the batch size is, the higher the Dice score is in the validation set and in the testing set. Thus, the similarity between the real segmentation and the one obtained after the training is higher. In addition, with small batch sizes, CUDA errors are less probable because Pytorch deals with smaller amounts of data. At the end of this process, four configurations were chosen, which were all the ones with a patch size of 32 and a batch size of 1 or 2.

Then, the binary masks that were obtained after the ablation study training were qualitatively studied for the four possible configurations. The gifs inferences created by Tensorboard were analysed by eye and it was observed that the configuration that segments the coronary arteries best is the one with a batch size of 2, a patch size of 32 and a learning rate of 0.1, as it was the one that segmented the best the coronary arteries in all the slices of the CCTA image.

The voxel size that was fixed along the training was 0.5x0.5x1 (width x length x height). The resolution was the only parameter that remained constant in the ablation study since voxel size variation was proven not to be useful in Acebes' work [17]. The percentages of images used in the final training were 80% for the training set (25 images), 15% for validation (5 images) and 5% for the test (2 images). *Table 9*, all the final hyperparameters are shown.

BATCH SIZE	PATCH SIZE	LEARNING RATE	RESOLUTION
2	32	0.1	0.5x0.5x1

Table 9. Chosen hyperparameters for the UNETR training.

5.4. UNETR Training

In this section the workflow followed to train the UNETR architecture segmentation algorithm is explained in detail.

5.4.1. Computer characteristics

The computer's characteristics play an important role in deep learning-based algorithms. All the trainings were computed in a HP Z8 G4 workstation with a Intel Xeon 4214 2400MHz processor and a 48GB NVIDIA RTX A6000 GPU. As the RAM memory is high, the trainings were able to be computed without any CUDA error and in a short period of time. Trainings need high RAM memory computers that can deal with big amounts of data. The more computer's RAM memory, the less training time is needed.

5.4.2. Initial requirements

All the computed training were done using Ubuntu 20.04. Thus, this operating system was downloaded before starting with the first training of the ablation study. Once Ubuntu was installed, before starting the ablation study, a virtual anaconda environment was created by running the first command that appears the "Requirements" section in the Annex of this project in the computer's terminal. This command also installs Python 3.8 due to MONAI works with this Python version. Then, Pytorch library was installed in the computer's terminal. Finally, SALMON GitHub was installed and all the libraries in that repository were downloaded in the same folder were the images and the labels are in our computer, as it is explained in the Annex.

5.4.3. Training workflow

In order to compute a training, the different downloaded SALMON Python scripts needed to be executed in the terminal. It is important to bear in mind that the terminal needs to be in the Anaconda virtual environment and in the same directory as the images, the labels, and the Python scripts. The first script to execute was *requirements.py*, which contains all the libraries required for the training, including MONAI. Then, the *organized_folder_structure.py* script, followed by the *init.py*, which needed to be overwritten with the desired hyperparameters before being executed, and the *train.py*. It is important to mention that all these scripts were previously understood and modified in order to achieve the main objective of the project. The code is available at the Annex of this project. Trainings follow up were done with Tensorboard, which contains all the metrics scores as well as the images inferences.

5.4.4. Number of epochs

In the ablation study, 150 epochs were done because it was the exact time in which the computer was not being used by any physician in the Department. Trainings were usually done at night and during the weekends. In the final training, 2500 epochs were done due to different factors. The physicians in the Department need Windows as an operating system to work every day. This means that the computer was unavailable to work with during all the time that the final training was been computing. Some licence of the Window's programs they need were installed in other computers to allow them to keep going with their works while the segmentation algorithm was being trained. Another limitation was the lack of time to perform the final training before the deadline of this thesis.

During the ablation study all the weights of the different ablation study configurations were saved in the `runs` folder. Therefore, all the data of the different configurations was able to be compared at the end of the study using Tensorboard. However, all the runs folder data was deleted before the final training in order to avoid CUDA errors, since this data occupy quite space in the computer's GPU.

5.5. Final training results

The final training results will be discussed in the following subsection by performing not only a quantitative analysis of the results by looking at the metrics obtained but also a qualitative analysis after performing some image post-processing.

5.5.1. Quantitative analysis

These metrics of the final training were obtained from Tensorboard. Before discussing the metrics obtained, the meaning of the different Dice scores will be reviewed:

- Training Dice score. It measures the similarity between the obtained segmentation in the images of the training set and the ground truth segmentation, which is the initial label. Thus, the higher the score is, the more similar both segments are. This metric is computed after every epoch, once all the network weights have been actualised. This information is useful for the network to train itself.

- Validation Dice score. After every epoch, once the network has learnt from the training set and its weights have been actualised, the validation set, which are images that the algorithm has never seen before, goes throughout the network and a final segmentation is obtained from that dataset. The validation Dice score measures the similarity between the obtained segmentation in the images of the validation set and the ground truth segmentation. In case the validation Dice score obtained in any epoch is higher than the previous ones, the weights will be conserved in the model. If not, the network will continue using the weights from the best model obtained to the date. This score gives us information about the algorithm's accuracy. The higher the Dice score is, the more similar both segmentations are and the better performance the algorithm has.

- Test Dice score. It measures the similarity between the obtained segmentation in the images of the testing set and the ground truth segmentation after every epoch. This score does not affect the network weights. The test Dice score gives us information about overfitting, which means that the network is learning by heart the training and the validation data set and it cannot be generalised for other images that it has never seen before.

Figure 14 shows the training, the validation and the test Dice score obtained. The plot shows the accuracy of every against the number of epochs. As it is seen in the validation Dice data, the model improves over time. The best validation model is at epoch 2284 and has a value of 0.8822. Thus, the most sensible conclusion that we can come up with is that, quantitatively, the coronary artery segmentation algorithm is significantly accurate. Qualitative results will be discussed in the following section.

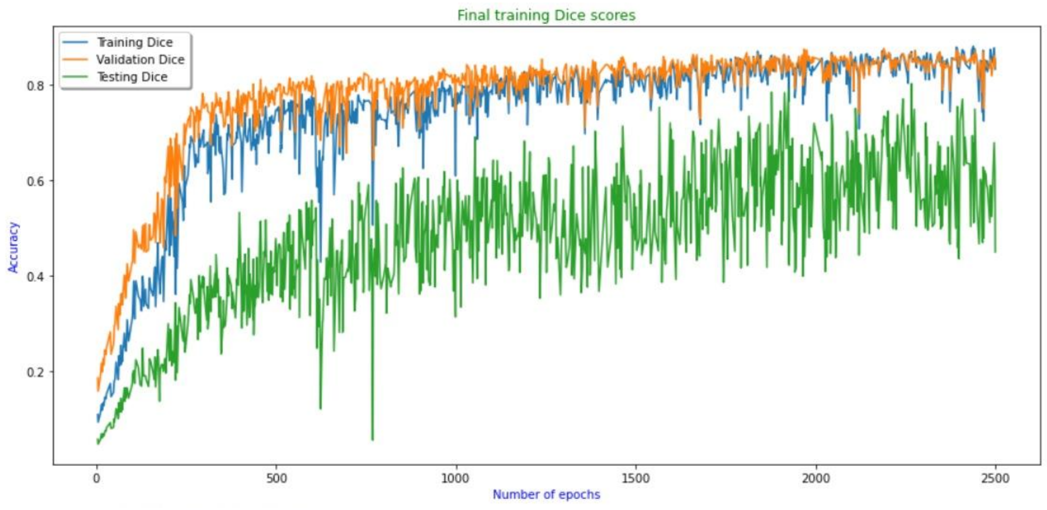


Figure 14. Training, validation, and testing Dice scores in the final training

Overfitting will be produced if in the test Dice data plot, the curve goes down at some point, which means that the model will be adapted to the training and validation dataset. Thus, we can conclude that the model does not present overfitting. However, the model has not arrived at the bias trade-off. It is the point between underfitting and overfitting, where best estimation of the model is achieved. In that case, a test Dice score curve would present an inflection point. Future lines of this project can work to achieve this point by computing a longer training.

After the training, 6 images from the extended dataset were corrected in order to compute the inferences. The Dice and the computation time to create the inference were calculated. Then, the inferences of the same images were obtained by using the nnU-Net that Acebes used in 2021 for his thesis. Thus, we are able to compare the UNETR quantitatively with the nnU-Net, which was trained almost with the same dataset but outside the Hospital's facilities. The evaluation metric used was the Dice score. The results are summarized in *Table 10.* The nnU-Net was not calculated for every epoch but the 6 inferences were generated in one minute and a half approximately, requiring 20 seconds per epoch. Means values are also presented in the Table.

	UNETR		**nnU-Net**
ID Case	***Time (s)***	***Dice***	***Dice***
1	37,4388	0,8020	0,9638
2	30,7566	0,8872	0,9626
3	30,8576	0,6598	0,9497
4	45,9536	0,7627	0,9178
5	46,015	0,5933	0,8832
6	32,8784	0,9131	0,9399
MEAN:	***35,1589***	***0,7824***	***0,9448***

Table 10. UNETR and nnU-Net inference Dice score and UNETR computational time and their mean values.

5.5.2. Qualitative analysis

The accuracy of the algorithm was also evaluated qualitatively. To do that, the *predict_single_image.py* Python script was used. It created the binary mask of 6 images that the network had never seen. First, these inferences were visualized and analysed in 3D Slicer to check if the errors produced by the algorithm can be assumed by the physician or if there exist serious problems in coronary artery segmentation. Afterwards, the segmentation could be corrected using the same software, before converting and exporting them as meshes, an STL file. This format allows for being used in computational fluid dynamics or finite element model simulations, for example. All these steps are called post-processing.

After the meshes of 6 images that the algorithm had never seen before were obtained, the performance of the UNETR algorithm was evaluated by comparing each of them with their ground truth and with the segmentation obtained using the nnU-Net . Then, the results were analysed. *Figure 15* shows the comparison between the UNETR inferences with the ground truth and the nnU-Net segmentation, visualized in 3D Slicer, in one of the 6 new images. Other cases are presented in Annex. The problems, the positive aspects to highlight as well as a comparison with the nnU-Net inference in every case are also explained in detail in the Annex.

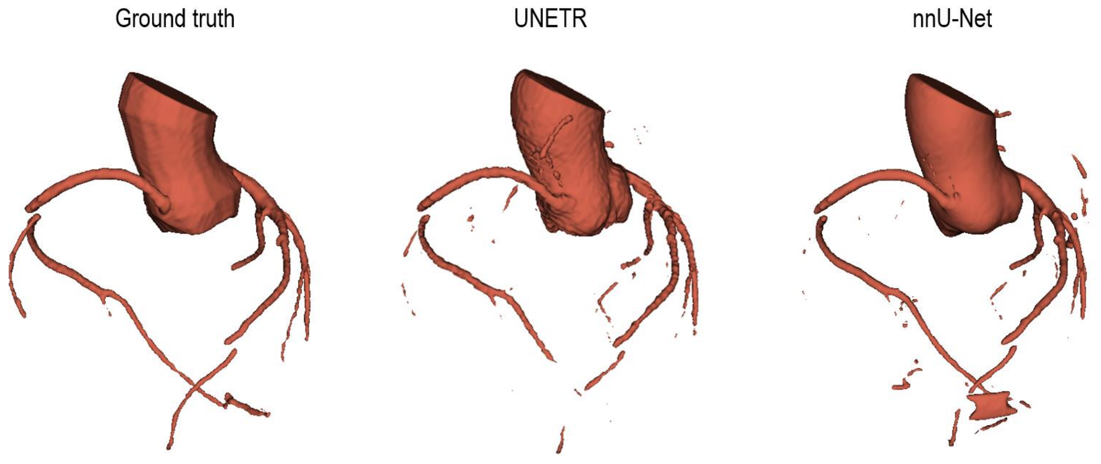


Figure 15. Comparison between the ground truth, the UNETR and the nnU-Net inference in one of the 6 new images.

Regarding the main UNETR problems, we can conclude that there exists several stops in the segmentation of the main branches, which cannot guarantee a complete analysis of the coronary arteries for the physician. Moreover, they present a large number of intra- and extra-pericardial artefacts, as it is seen in *Figure 15.* We can also appreciate certain roughness in the obtained segmentation. Last but not least, the aorta is not well segmented. However, it is not one of the most relevant problems to highlight because the main objective is the coronary artery tree segmentation.

Concerning positive aspects to highlight about the UNETR architecture, we can conclude that the main shape and contour of the coronary artery are well detected. Furthermore, it does not detect the presence of non-coronary vessels and the root of the aorta is well delineated. CT artefacts are well detected by the network as well.

Finally, in comparison with the nnU-Net, we can sum up that fewer main branches are detected with the UNETR and there is a big presence of artefacts in their segmentations. UNETR inferences are less smooth than nnU-Net ones.

As seen, the most commonly reported error that was present in each of the cases was a large number of intra- and extra-pericardial artefacts. Therefore, a future line of this project can be developing a pericardial segmentation model to identify the pericardial volume to exclude them. Other problems could be solved with a longer training.

5.6. Discussion. Limitations and future implementations.

Having a well-done segmentation in a fully automated way is fundamental to simplify the long and tedious segmentation labours done by physicians and to improve the diagnosis of the diseases. Therefore, incorporating it into the current clinical practise would be largely beneficial, especially for saving physicians' time. The UNETR presented segmentations with the coronary artery shape and contours well-defined but with several stops in the segmentation of the main branches. However, these could be corrected by healthcare professionals after obtaining the inference. Moreover, there was a huge presence of intra- and extra-pericardial artefacts in the obtained UNETR segmentations, which could have difficulted the diagnosis in case the algorithm was implemented in the clinical practice. The use of post-processing tools will help to enhance the segmentation quality.

The limitations of the algorithm were produced by different factors. The most important one was that the training was used with a small dataset. It would be necessary to increase the training dataset and retrain the network in order to obtain more accurate segmentations before incorporating the model into the Hospital. This would ensure the presence of coronary arteries with rare anatomies in the training set, the model would have a higher generalisation and better outcomes would be obtained.

In addition, a short training was done due to the lack of time to present this thesis and the short availability of the Hospital's computer, which was been used by the physicians for a period of time. The number of epochs in the algorithm training needs to be increased to obtain better outputs. The ideal would be to reach the bias trade-off point. The main branches' stops as well as the non-well segmentation of the aorta are consequences of a short training. It would be necessary to retrain the network with more than 2500 epochs before incorporating the model into the Hospital.

Other causes of the model's drawbacks are that only a few combinations of hyperparameters in the ablation study were performed and the labels contained not only the coronary arteries but also the aorta, which led to problems in the inferences creation. Thus, in future lines, more combinations of hyperparameters could be done as well as performing the training only with coronary artery labels, which are the anatomical structure that we are interested in. In this way, rejecting the aorta, the model would be more specific. Furthermore, the computer did not use 100% of its capacity to perform the training. As a consequence, faster training could have been obtained.

When comparing the performance of the segmentation methodology developed in this work with the nnU-Net that Acebes used in his work last year, it can be found that the detection of the main branches is higher and the artefacts are lower in the nnU-Net. It is since the small features, such as small coronary vessels, are better detected in a U-Net based architecture rather than in a UNETR one.

However, one of the strongest points of this project is that it could be implemented in the clinical practice and the training was performed inside the Hospital's facilities, not as the nnU-Net training and its inferences obtention, which was done in the Universitat Pompeu Fabra High Performance Computing Cluster. This dependence in an external institution makes the workflow slower.

Additionally, the algorithm needs to be encapsulated into an end-to-end interface if it is implemented in clinical practice. Thus, the model would be more user-friendly for physicians, which are the main users of the segmentation algorithm. The application must be able to obtain the inference of the image and convert the NIfTI inference into STL, which is a widely used mesh format. In this way, they would only have to validate and correct the errors. One of the future lines of this project would be using a deploy tool that MONAI offers in order to create this interface. MONAI has a tool called MONAI deploy that can help in the development of the interface.

Regarding the above, we can conclude that the feasibility of the project is high, provided that the limitations are easy to overcome and then, it could be implemented in the clinical workflow. The extended dataset can be used if the errors are solved and future lines can work on creating the interface. The fact that the segmentation algorithm is able to generalize well even without a big dataset is very encouraging. The created algorithm may work as a support tool for the semi-automated segmentation done by healthcare professionals nowadays.

6. EXECUTION CHRONOGRAM

In this section, tasks will be hierarchical decomposed using a Work Breakdown Structure (WBS) schedule as well as they will be detailly explained in order to achieve the main objective of the project. Then, GANTT diagrams will be presented to get a clear perspective of the work and timings within the project.

6.1. Work Breakdown Structure (WBS)

Figure 16 shows the WBS followed to achieve the main goals of the project, where the main tasks and sub-tasks are decomposed.

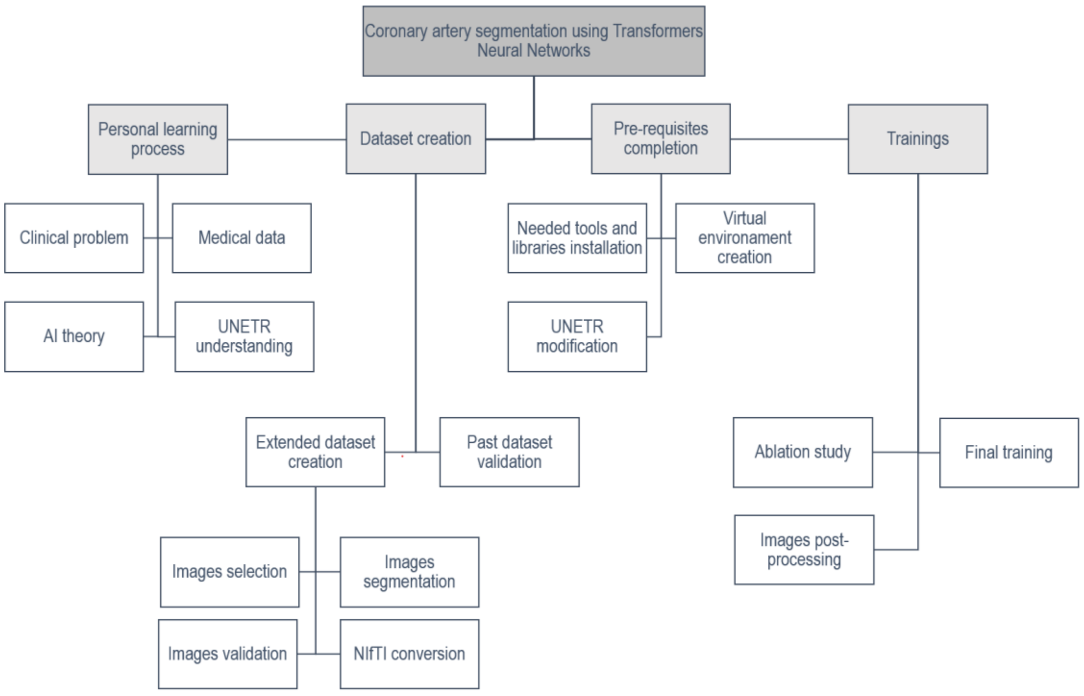


Figure 16. Project's WBS.

6.1.1. WBS dictionary

The segmentation of the coronary artery using Transformers Neural Networks is the ultimate objective of this project. The sub-tasks are explained in this subsection.

i. Personal learning process. Some literature was needed to be read before starting the project. Then, the background and state of the art were written in this thesis. The knowledge acquired can be divided into 4 main areas:
   - Clinical problem. Coronary Artery Disease (CAD) was deeply studied to understand the main medical cause behind the main goal of the project.
   - Medical data. The different data types obtained along the current CAD diagnosis workflow as well as the medical data needed for the algorithm training were understood.
   - AI theory. Basic AI and deep learning theory concepts were studied to know how the deep learning-based algorithm works.
   - UNETR. The chosen architecture was studied layer by layer theoretically and computationally to understand how it exactly works. MONAI and SALMON were also studied.

ii. Dataset creation. First, the past dataset validation created by the Cardiac Imaging Department was validated in order to guarantee that all the labels were correct and all the images were able to be read by the segmentation algorithm. Then, the extended dataset was created although it was not finally used in training purposes. This latest task can be split into the following parts:
- Image selection. It consists of selecting the desired amount of CCTA images from the Hospital's PACS.
- Image segmentation. A cardiac imaging expert segmented manually all the main cardiac structures in the images one by one using the automatic segmentation tool that Philips Intellispace Portal 10.1 platform offers, which is called CT Comprehensive Cardiac Analysis. The main artefacts were also discarded to ensure the maximum quality.
- Images validation. A manual correction was done to all the segmentations obtained by the software. They were exported in STL format.
- NIfTI conversion. The images and the labels were converted into a 3D NIfTI format to be able to be read by the deep learning-based segmentation tool.

iii. Pre-requisites completion. The main sub-tasks that needed to be done before starting training the algorithm were:
- Needed tools and libraries installation, such as Ubuntu, MONAI, among other required libraries in the training.
- Virtual environment creation. The virtual environment with Python 3.8 was installed on the computer where the training was done.
- UNETR modification. The UNETR code provided by SALMON that was previously studied was modified in order to read CT images. Hyperparameters were also overwritten and some useless parts, were deleted to reduce computational complexity.

iv. Trainings. First, an ablation study was done to study the performance of the network hyperparameters and their contribution to the overall model by changing certain components. The combinations of the different patch and batch size, as well as the learning rate, were studied in a 150 epochs training with the final 32 images dataset to select the hyperparameters that fit best to the segmentation algorithm were selected. Subsequently, a 2500 epochs final training was performed with the chosen hyperparameters. Finally, the visualization of the inference as well as the post-processing was done with 3D Slicer.

6.2. GANTT execution chronogram

The global project had a duration of 8 months, starting the 4th of October 2021, and finishing the 30th of May 2022. In order to develop the project in time, the mentioned tasks and subtasks were organised in time. The personal learning process, the dataset creation and the pre-requisites completion tasks were planned to be done in the first semester of the academic year 2021-2021, starting the 4th of October and finishing the 20th of December, before the Christmas break.

Then, the training tasks such as the ablation study and the final training, as well as the post-processing part were planned to be done in the second semester of the same academic year.

Table 11 and *Table 12* show the time associated to every subtask mentioned in the WBS for the first and second semester, respectively, as well as their programmed start date. It is important to mention that the image selection and the image segmentation of the extended dataset was done by a Cardiac Imaging expert before the project started. Thus, they are not going to be taken into consideration in the project's GANTT diagram.

TASK	DESCRIPTION	DURATION (WEEKS)	START DATE
PERSONAL LEARNING PROCESS			
A	Clinical problem	2	4th October 2021
B	Medical data	1	18th October 2021
C	AI theory	2	25th October 2021
D	UNETR understanding	1	8th November 2021
DATASET CREATION			
E	Past dataset validation	1	15th November 2021
F	Extended dataset creation. Images selection	1	29th March 2021
G	Extended dataset creation. Images segmentation	12	5th April 2021
H	Extended dataset creation. Images validation	3	22nd November 2021
I	Extended dataset creation. NIfTI conversion	1	6th December 2021
PRE-REQUISITES COMPLETION			
J	Needed tools and libraries installation	1	8th November 2021
K	Virtual environment creation	1	8th November 2021
L	UNETR modification	1	8th November 2021

Table 11. Tasks involved the project in the first semester of the academic year 2021-2022 and their associated timing.

TASK	DESCRIPTION	DURATION (WEEKS)	START DATE
TRAININGS			
M	Ablation study	12	24th January 20222
N	Final training	2	2nd May 2022
O	Images post-processing	2	16th May 2022

Table 12. Tasks involved the project in the second semester of the academic year 2021-2022 and their associated timing.

In order to get a clear perspective of the work and timings within the project, a GANTT diagram was executed. *Figure 17* and *Figure 18* show the programmed timing for the global project is visualised in a GANTT execution chronogram for the second and first semester, respectively. As it is seen in *Figure 19* there was a break the last two weeks of April since it was exams period.

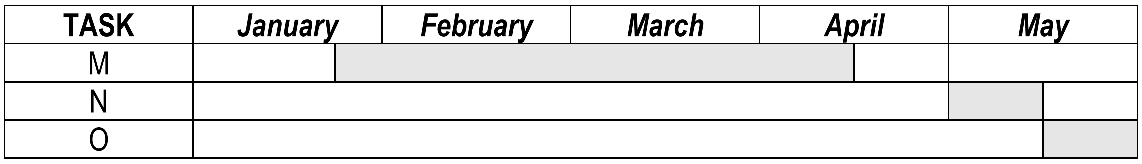


Figure 17. GANTT diagram of the project in the second semester of the academic year 2021-2022.

	Oct 2021				***Nov 2021***					***Dec 2021***	
TASK	4th	11th	18th	25th	1st	8th	15th	22nd	29th	6th	13th
A	▨	▨									
B			▨								
C				▨	▨						
D						▨					
E							▨				
H								▨	▨	▨	
I											▨
J						▨					
K						▨					
L						▨					

Figure 18. GANTT diagram of the project in the first semester of the academic year 2021-2022.

7. TECHNICAL FEASIBILITY

This chapter presents a strategic analysis to the project regarding technical viability. SWOT analysis, shown in *Figure 19* , is the tool that was used to evaluate independently the strengths, weaknesses, opportunities, and threats of the global project. They will be explained in detail in the following subsections.

STRENGTHS	**WEAKNESSES**
- Uniqueness. - Independence on external vendors. - Less time-consuming. - Less error-prone. - Execution without coding knowledge.	- Small dataset. - Short training. - Two different CT scanners manufacturers. - No software interface. - Big RAM memory computer.
OPPORTUNITIES	**THREATS**
- Integration to the Hospital's workflow. - Development of a personalized workflow. - Other anatomical structure segmentation. - Interested companies.	- Similar solution in the market.

Figure 19. SWOT analysis of the project

7.1. Strengths

- Uniqueness. UNETR architecture as a coronary artery segmentation algorithm has only been assessed in the Cardiac Imaging Department in Hospital de Sant Pau. No software that integrates this automated segmentation tool has been launched to the market.

- Independence of external vendors. The final inference can be obtained in the same hospital, without sending the images to external companies.

- Less time-consuming. Current segmentation methods are tedious and time-consuming tasks that are usually done by healthcare professionals instead of spending their time doing more specialized tasks. Therefore, the implementation of the UNETR as an automatic segmentation tool would facilitate professionals' jobs.

- Less error-prone. Current segmentation methods are manual tasks that are usually done by healthcare professionals, which can lead to some human errors. Replacing the current method with an automatic tool like UNETR would significantly reduce the possibility of making errors during the segmentation process.

- Execution without coding knowledge. It is not required a big knowledge of computing to obtain the final segmentation using the network. Physicians, as an example, are able to run the scripts in order, as it was explained in the Detailed Engineering section.

7.2. Weaknesses

- Small dataset. Although the results were pretty accurate and no overfitting was produced, the final training was done with a very small dataset. Thus, the model's complexity is small and more images will be needed for a better segmentation algorithm.

- Short training. Based on the obtained results, the performance of the algorithm would be better if a longer training had been done.

- Two different CT scanner manufacturers. They were used to acquire the dataset images. Even though the algorithm could have been used for Creu Blanca Clinics and Hospital de Sant Pau, the generalization of the deep learning-based algorithm would be low.

- No software interface. The segmentation algorithm is the only thing that was developed in this project. A software interface is a future line of this work, which would make the algorithm more user friendly for physicians and other healthcare professionals.

- Big RAM memory computer. A computer with a specific features such as a big RAM memory and a GPU was needed for the training. Other computers as well as Cloud tools such as Google Collaborate led to CUDA errors.

7.3. Opportunities

- Integration to the Hospital's workflow. The fact that the algorithm can be used in the same Hospital because of the potential features of the used computer, without depending on other supercomputing centres and universities, facilitate the daily work of professionals. This is a new solution for the market.

- Development of a personalized workflow. The segmentation tool can be adapted to the workflow in every clinic.

- Other anatomical structures segmentation. In this work only the coronary arteries were segmented but other cardiac anatomical structures can be segmented with the same deep learning architecture if the model is re-trained.

- Interested companies. Big computer graphics companies such as NVIDIA may be interested in providing support and equipment to the same project if it is done with a bigger dataset and a longer training.

7.4. Threats

- Similar solution in the market. Cleerly and Heartflow also offer related techniques.

8. ECONOMICAL FEASIBILITY

In this section, the total economic costs of the project are reviewed in *Table 13* and they will be explained in detail in the following subsections. The costs include all the software and hardware needed to develop the global project, as well as the salary of the workers involved. They have been approximated by the information provided by the Cardiac Imaging Department in Hospital de la Santa Creu I Sant Pau.

ITEM	COST
Labour	
Biomedical Engineer student	3900 €
Senior R&D Project engineer	1750 €
Cardiac Imaging expert	560 €
Software	
Philips Intellispace Portal 9	15,000 €
Mimics Innovation Suite annual license	13,000 €
Office 365 annual license	90 €
Ubuntu, MONAI, SALMON, 3D Slicer	0 €
Hardware	
HP Z8 G4 90 workstation	7760 €
48GB NVIDIA RTX A6000 GPU	
ASUS ZenBook UX381U	940 €
TOTAL:	43,000 €

Table 13. Expenses of the project based on the type of cost.

8.1. Labour costs

A Biomedical Engineering Student, a Cardiac Imaging expert and a Senior R&D Project engineer were the needed professionals to develop the project.

To estimate the cost of the student, the main developer of the project, it has been considered that the net annual salary of a Junior Engineer, a graduate engineer with less than 2 years of experience, is around 25.000€/year in Spain. This leads to a salary of 13€/h. The final cost was also calculated considering that the student must dedicate at least 300 hours developing the project.

The cost of the labour of the Senior engineer, the main supervisor of the project, was calculated considering that the net annual salary of a Senior R&D Project Engineer is around 50.000€/year in Spain, which it is an approximate amount of 25€/h. It was also taken into consideration that the Senior engineer only was involved in the project in a weekly two-hour meeting where the execution of the 35 weeks project was planned, controlled, and evaluated.

The cardiac imaging expert was responsible for segmenting the CCTA images in order to generate the extended dataset, working for 5 hours a day for three months before the start of the project. The cost of the Cardiac Imaging expert was estimated considering that its net annual salary is around 50.000€/year in Spain, which is an equivalent hourly rate of 25€/h.

8.2. Software costs

The main software used was Philips Intellispace Portal 9 and it include the CT Comprehensive Cardiac Analysis tool as one of its packages. It is the highest expense involved in the project, and it was used to segment the CT images for the labelled dataset creation. However, it is a tool which is usually used in the clinical practise in medical imaging departments.

Materialise Mimics was the software used to correct the labels of the past and the extended dataset that were not well correctly segmented by the Cardiac Imaging expert with the semi-automatic tool that the Philips software provided. As it was said before, other free-source software such as 3D Slicer could have been also used but Mimics was the one that it was implemented in the Hospital since it is a faster and simpler tool. It is also used to segment other anatomical structures for 3D printing project that have been also developed in the Department.

The 365 Office licence was needed to write this project as well as to keep the anonymised dataset in the cloud.

All the software that have been used during the development of the project but they are not specified in *Table 11* are free and open-source, such as 3D Slicer, MONAI, SALMON, Ubuntu, among others.

8.3. Hardware cost

As hardware items, the HP Z8 G4 workstation with an Intel Xeon 4214 2400MHz processor and a 48GB NVIDIA RTX A6000 GPU was used because it provides a big RAM memory to train the segmentation deep learning-based algorithm, allowing a faster training with less CUDA errors. It was also used because, as it is in the Hospital, the algorithm could have been integrated into the clinical workflow, avoiding relying on external companies, research centers and super-computing facilities. However, the computer was also used by the Cardiac Imaging experts in their daily work. Thus, an ASUS laptop as a personal computer was also needed to validate the labels, correct them, overwrite the algorithm code, and write the project's thesis.

Regarding the two CT scans, the Philips and the Toshiba one, were also used for other specialties in the Hospital and in Creu Blanca Clinic, respectively, as diagnosis tools. Although they are the main tool to acquire the CCTA images, the key item in the overall project, they are not considered as expenses of the project because it is assumed that the Hospital had them from before It is also used by other specialties. As well as the PACS, which was also one of the key elements of the global project since the selected images were extracted from there.

8.4. Final budget

Considering the above, the estimated cost for this project is around 43,000 €. We can conclude that the final budget is a pretty affordable amount for a deep learning project considering that the most expensive expenses correspond to already implemented software in the Hospital.

9. REGULATIONS AND LEGAL ASPECTS

In this section, not only the definition of a software medical device will be clarified and several regulatory and legal aspects that the Transformers Neural Network-based algorithm for coronary artery segmentation would have to fulfil if it would be launch to the market will be discussed.

9.1. Definition of medical software device

In order to be qualified as a Medical Device Software (MDSW), the product must first fulfil the definition of software and the definition of a medical device according to the Article 103 of the European Union Regulation (EU) 2017/745 by the Medical Device Coordination Group (MDCG) [92]. The mentioned articles replaced the former EU Medical Device Direction, imposing stringent regulatory requirements that need to be met before any medical device, including AI software tools, can be used in clinical practice [93].

On one hand, a software is defined as a set of instructions that processes input data, such as images for medical purposes, and creates output data, a binary mask of the segmentation. On the other hand, a medical device is any instrument, apparatus, appliance, or software, among others, intended by the manufacturer to be used, alone or in combination, for human beings for one or more specific medical purposes such as diagnosis, prevention, monitoring, prediction, prognosis, treatment, etc. and which does not archive its intended action by pharmacological, immunological, or metabolic means [92]. The Transformer Neural Network-based algorithm for the coronary artery segmentation can be considered as a MDSW according to the previous definitions because, although it has no interface which will allow its easy use in the clinical field, it transforms CCTA DICOM files into a screen display with the segmentation of the coronary arteries to diagnose CAD individual patients, providing information and support for healthcare professionals, especially in therapeutical decisions. Therefore, as our algorithm is qualified as a MDSW, it falls under the EU Medical Devices Regulation (MDR) 2017/745. *Figure 20* shows the decision steps to assist the qualification of MDSW.

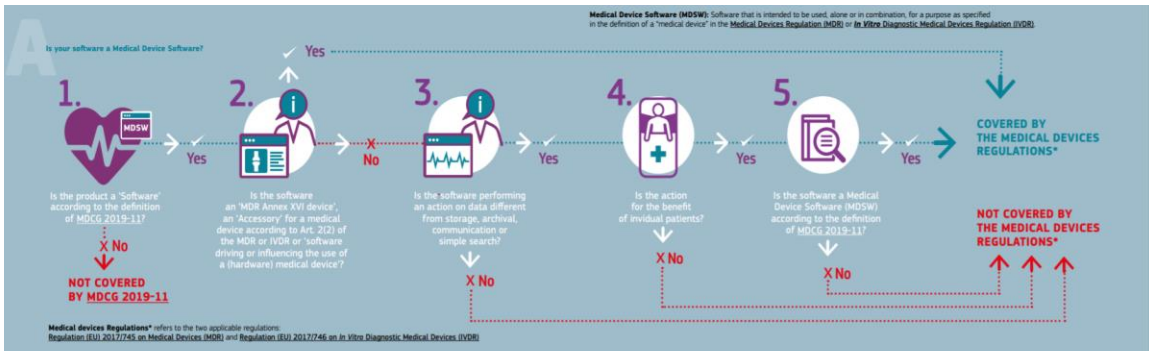

A
Is your software a Medical Device Software?
Medical Device Software (MDSW): Software that is intended to be used, alone or in combination, for a purpose as specified in the definition of a "medical device" in the Medical Devices Regulation (MDR) or In Vitro Diagnostic Medical Devices Regulation (IVDR)
1.
MDSW
Is the product a 'Software' according to the definition of MDCG 2019-11?
Yes
X No
NOT COVERED BY MDCG 2019-11
2.
Is the software an 'MDR Annex XVI device', an 'Accessory' for a medical device according to Art. 2(2) of the MDR or IVDR or 'software driving or influencing the use of a (hardware) medical device'?
Yes
No
3.
Is the software performing an action on data different from storage, archival, communication or simple search?
Yes
X No
4.
Is the action for the benefit of invidual patients?
Yes
X No
5.
Is the software a Medical Device Software (MDSW) according to the definition of MDCG 2019-11?
Yes
X No
COVERED BY THE MEDICAL DEVICES REGULATIONS*
NOT COVERED BY THE MEDICAL DEVICES REGULATIONS*
Medical devices Regulations* refers to the two applicable regulations: Regulation (EU) 2017/745 on Medical Devices (MDR) and Regulation (EU) 2017/746 on In Vitro Diagnostic Medical Devices (IVDR)

Figure 20. Scheme followed to determine if a medical device software is covered by the European Medical Device Regulations. From the European Commission [94].

9.2. Regulation (EU) 2017/745

All implementing rules in Annex VII of EU Regulation 2017/745 [93] shall be considered in medical software devices. It is especially needed to explain the intended use of the medical software device and to classify the software under the EU MDR classification scheme, mentioned in rule 11 of EU Regulation 2017/745. Its classification depends on the patient's condition and the information provided by the device for the clinical decision.

Deeply, rule 11 states [93]:

"Software intended to provide information which is used to take decisions with diagnosis or therapeutic
purposes is classified as class IIa, except if such decisions have an impact that may cause:

- *death or an irreversible deterioration of a person's state of health, in which case it is in class III; or*
- *a serious deterioration of a person's state of health or a surgical intervention, in which case it is classified as class IIb.*

Software intended to monitor physiological processes is classified as class IIa, except if it is intended for monitoring of vital physiological parameters, where the nature of variations of those parameters is such that it could result in immediate danger to the patient, in which case it is classified as class IIb."

In our case, our MDSW is a Class IIa since it is used by patients under non-serious conditions and there is high information provided to provide the diagnosis.

Moreover, in case it was placed in the market with other medical devices, such as hardware, it shall be classified in its own right and the risk class shall not be lower than the risk class of the hardware medical device, as the implementing rule 3.3 of Regulation (EU) 2017/745 considers. In theory, our product is the MDSW without any other medical device so this regulation won't affect our product.

Although rules 9, 10 and 12 mainly categorize the risk related to the exchange of energy/substances between the body and the diagnostic device, in our case the majority of the risk associated with the device are associated with wrong use of the information by health professionals, especially in wrong therapeutical decisions, and not with the direct risk that the device can cause to patients. Manufacturers should bear in mind the mentioned rules but they are not directly related to the main purpose of the medical device.

9.3. General Safety and Performance Requirements

General Safety and Performance Requirements (GSPRs) provide standards and industry guidance under the current EU Regulation 2017/745 definitions, establishing conformity with the Medical Device Directive (MDD, 93/42/EEC) and Active Implantable Medical Device Directive (AIMDD, 90/385/EEC) [95]. They provide essential requirements for the design and development of medical software devices, among others, that manufacturers must bear in mind.

For MDSW like ours, we should pay special attention to the following requirements [95]:

- GSPR 14.2 is addressed to software risks. The software validation, the consideration of cybersecurity and network potential risks cannot be validated by the manufacturer. These issues are currently somewhat addressed in EN 60601-1 (requirements for medical electrical equipment), in IEC/ISO 80001 series (risk management for IT networks incorporating medical devices) and IEC 82304-1 (health software). The latest standard applies to the safety and security of health software products designed to operate on general computing platforms and intended to be placed on the market without dedicated hardware, and its primary focus is on the requirements for manufacturers. It covers the entire lifecycle including design, development, validation installation, maintenance, and disposal of health software products [96].

- GSPR 17 defines requirements for electronic programmable systems and software considered to be a medical device in itself. Manufacturers must satisfy the standard EN 60601-1+A1 (requirements for medical electrical equipment). This SPR also adds additional detail addressing cyber security and data protection. Manufacturers should consider these aspects including data encryption, levels of access and username/password formats.

9.4. Market implementation regulations

If hypothetically the Transformers Neural Networks-based algorithm was introduced in the market, as a medical device, it would have to satisfy general requirements to obtain CE Marking. A CE Mark is your declaration that the product complies with the Essential Requirements of the relevant European Legislation. To obtain the CE mark, the manufacturer has to deliver technical documentation and write the Declaration of Conformity, where the manufacturer confirms that the product complies with the EU MDR [97]. By having a CE Mark, you will have immediate access to all EU and EEA markets as well as any other international markets where CE Marking is accepted, so it will guarantee a safety and a proper commercialization of the product around the European market.

10. CONCLUSIONS

The purpose of this project was to develop a coronary artery segmentation algorithm using Transformers Neural Networks, specifically using a UNETR architecture, and study its performance to define if it would be brought into clinical practice.

Different options to develop the segmentation algorithm were studied by analysing the already existing deep learning-based medical imaging segmentation tools as well as possible configurations for the network itself. It was relevant to consider that the final aim of the algorithm would be a diagnostic tool that could help clinicians to diagnose coronary artery disease in order to choose the best possible network and its best configuration. Thus, the performance of the algorithm needed to be as accurate as possible.

UNETR was chosen for its excellent performance in medical segmentation since its architecture is able to predict long-range pixel dependencies. Moreover, it requires a lower GPU memory and a lower computational time than other approaches. The model was configurated by an ablation study because it was less error-prone than other self-configuration models.

Regarding the objectives, they were successfully accomplished. Previous experience in the programming language as well as the basic knowledge of medical imaging help to achieve the main goal of the project.

Firstly, the transformer-based algorithm for the segmentation of the coronary artery from CCTA images was successfully developed. The results of the obtained inferences after the segmentation were presented. They provided reasonable results qualitatively and qualitatively. We can conclude that the segmentation algorithm can generalize well even without a big dataset, which is very encouraging for project's future lines.

Subsequently, a comparison analysis of the trained algorithm with a different approach was also done, the nnU-Net. We can conclude that the UNETR qualitative results were significantly worse since nnU-Net is better extracting small image features. However, the nnU-Net training depended on external institutions, making impossible its implementation in a hospital.

Besides the study of the viability of implementing the created algorithm into a clinical workflow, the most sensible conclusion we can come up with is that the feasibility of incorporating the project in the clinical workflow is high, provided that its limitations can be overcome. By increasing the training dataset and retraining the UNETR in a longer training, as well as creating an end-to-end interface able to compute the segmentation using the model and perform the following post-processing, it would be possible to integrate the model into day-to-day clinical practice.

By way of conclusion, the UNETR showed segmentations with the coronary artery shape and contours well-defined but with several stops in the segmentation of the main branches and a huge presence of intra- and extra-pericardial artefacts. However, implementing this advanced segmentation algorithm in the clinical practice would automate processes, especially obtaining the computational markers of disease necessary for diagnosis. Thus, the workload of the healthcare professionals would decrease, as they would only be responsible for the post-processing part and validation of the segmentation. Therefore, they would be able to devote more time studying these markers to enhance the diagnosis of patients and provide a more personalised treatment.

REFERENCES

[1] "Cardiovascular diseases (CVDs)." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). [Accessed: 15-Dec-2021].

[2] G. A. Roth *et al.*, "Global, Regional, and National Burden of Cardiovascular Diseases for 10 Causes, 1990 to 2015," *J. Am. Coll. Cardiol.*, vol. 70, no. 1, pp. 1–25, Jul. 2017.

[3] A. Mangla, E. Oliveros, K. A. Williams, and D. K. Kalra, "Cardiac Imaging in the Diagnosis of Coronary Artery Disease," *Curr. Probl. Cardiol.*, vol. 42, no. 10, pp. 316–366, Oct. 2017.

[4] C. Chen *et al.*, "Deep Learning for Cardiac Image Segmentation: A Review," *Front. Cardiovasc. Med.*, vol. 7, p. 25, Mar. 2020.

[5] X. Tang, "The role of artificial intelligence in medical imaging research," *BJR Open*, vol. 2, no. 1, p. 20190031, Nov. 2020.

[6] X. Tizon and Ö. Smedby, "Segmentation with gray-scale connectedness can separate arteries and veins in MRA," *J. Magn. Reson. Imaging*, vol. 15, no. 4, pp. 438–445, 2002.

[7] Y. Tian, Y. Pan, F. Duan, S. Zhao, Q. Wang, and W. Wang, "Automated segmentation of coronary arteries based on statistical region growing and heuristic decision method," *Biomed Res. Int.*, vol. 2016, 2016.

[8] A. C. S. Chung, J. A. Noble, and P. Summers, "Vascular segmentation of phase contrast magnetic resonance angiograms based on statistical mixture modeling and local phase coherence," *IEEE Trans. Med. Imaging*, vol. 23, no. 12, pp. 1490–1507, Dec. 2004.

[9] N. Flasque, M. Desvignes, J. M. Constans, and M. Revenu, "Acquisition, segmentation and tracking of the cerebral vascular tree on 3D magnetic resonance angiography images," *Med. Image Anal.*, vol. 5, no. 3, pp. 173–183, Sep. 2001.

[10] Y. Sato *et al.*, "Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images," *Med. Image Anal.*, vol. 2, no. 2, pp. 143–168, 1998.

[11] B. Bouraoui *et al.*, "3D segmentation of coronary arteries based on advanced mathematical morphology techniques," *Comput. Med-ical Imaging Graph.*, vol. 34, no. 5, pp. 377–387, 2010.

[12] M. A. Gülsün, G. Funka-Lea, P. Sharma, S. Rapaka, and Y. Zheng, "Coronary Centerline Extraction via Optimal Flow Paths and CNN Path Pruning," *undefined*, vol. 9902 LNCS, pp. 317–325, 2016.

[13] P. Moeskops *et al.*, "Deep Learning for Multi-Task Medical Image Segmentation in Multiple Modalities," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9901 LNCS, pp. 478–486, Apr. 2017.

[14] J. M. Wolterink, T. Leiner, B. D. de Vos, R. W. van Hamersvelt, M. A. Viergever, and I. Išgum, "Automatic coronary artery calcium scoring in cardiac CT angiography using paired convolutional neural networks," *Med. Image Anal.*, vol. 34, pp. 123–136, Dec. 2016.

[15] A. Hatamizadeh *et al.*, "UNETR: Transformers for 3D Medical Image Segmentation," Mar. 2021.

[16] A. Vaswani *et al.*, "Attention Is All You Need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, pp. 5999–6009, Jun. 2017.

[17] C. Acebes Pinilla and A. H. Moustafa David Viladés Rubén Leta, “Artificial Intelligence integration within a clinical cardiac imaging workflow Application to coronary artery segmentation,” 2021.

[18] “Download Ubuntu Desktop | Download | Ubuntu.” [Online]. Available: https://ubuntu.com/download/desktop. [Accessed: 15-Dec-2021].

[19] F. Isensee, P. F. Jaeger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, “nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation,” *Nat. Methods 2020 182*, vol. 18, no. 2, pp. 203–211, Dec. 2020.

[20] I. Ogobuiro, C. J. Wehrle, and F. Tuma, “Anatomy, Thorax, Heart Coronary Arteries,” *StatPearls*, Jul. 2021.

[21] Patrick J. Lynch, “Anatomy of the coronary tree ,” 2010. [Online]. Available: https://commons.wikimedia.org/wiki/File:Coronary_arteries.svg. [Accessed: 24-Nov-2021].

[22] J. F. Bentzon, F. Otsuka, R. Virmani, and E. Falk, “Mechanisms of plaque formation and rupture,” *Circ. Res.*, vol. 114, no. 12, pp. 1852–1866, Jun. 2014.

[23] A. MacDowall, M. Skeppholm, L. Lindhagen, Y. Robinson, and C. Olerud, “Effects of preoperative mental distress versus surgical modality, arthroplasty, or fusion on long-term outcome in patients with cervical radiculopathy,” *J. Neurosurg. Spine*, vol. 29, no. 4, pp. 371–379, Oct. 2018.

[24] University of Rochester Medical Center, “Problems associated to atherosclerosis.” [Online]. Available: https://www.urmc.rochester.edu/MediaLibraries/URMCMedia/hh/services-centers/cardiology/Cholesterol.jpg. [Accessed: 24-Nov-2021].

[25] B. A. Ference *et al.*, “Low-density lipoproteins cause atherosclerotic cardiovascular disease. 1. Evidence from genetic, epidemiologic, and clinical studies. A consensus statement from the European Atherosclerosis Society Consensus Panel,” *Eur. Heart J.*, vol. 38, no. 32, pp. 2459–2472, Aug. 2017.

[26] M. Tavakol, S. Ashraf, and S. J. Brener, “Risks and Complications of Coronary Angiography: A Comprehensive Review,” *Glob. J. Health Sci.*, vol. 4, no. 1, p. 65, 2012.

[27] M. Zreik *et al.*, “Deep Learning Analysis of Coronary Arteries in Cardiac CT Angiography for Detection of Patients Requiring Invasive Coronary Angiography,” *IEEE Trans. Med. Imaging*, vol. 39, no. 5, pp. 1545–1557, May 2020.

[28] K. M. Abdelrahman *et al.*, “Computed Tomography Angiography From Clinical Uses to Emerging Technologies: JACC State-of-the-Art Review,” *J. Am. Coll. Cardiol.*, vol. 76, no. 10, p. 1226, Sep. 2020.

[29] F. R. de Graaf *et al.*, “Clinical application of CT coronary angiography: state of the art,” *Heart. Lung Circ.*, vol. 19, no. 3, pp. 107–116, Mar. 2010.

[30] “Understanding the difference between AI, ML, and DL | TechGig.” [Online]. Available: https://content.techgig.com/understanding-the-difference-between-ai-ml-and-dl/articleshow/75493798.cms. [Accessed: 16-Jan-2022].

[31] “FCN Explained | Papers With Code.” [Online]. Available: https://paperswithcode.com/method/fcn. [Accessed: 15-Dec-2021].

[32] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell.*

Lect. Notes Bioinformatics), vol. 9351, pp. 234–241, 2015.

[33] P. Sardar, J. D. Abbott, A. Kundu, H. D. Aronow, J. F. Granada, and J. Giri, "Impact of Artificial Intelligence on Interventional Cardiology: From Decision-Making Aid to Advanced Interventional Procedure Assistance," vol. 12, no. 14, pp. 1293–1303, Jul. 2019.

[34] M. Peng, C. Wang, T. Chen, G. Liu, and X. Fu, "Dual temporal scale convolutional neural network for micro-expression recognition," *Front. Psychol.*, vol. 8, no. OCT, Oct. 2017.

[35] "The math behind Gradient Descent and Backpropagation | by Enghin Omer | Towards Data Science." [Online]. Available: https://towardsdatascience.com/the-math-behind-gradient-descent-and-backpropagation-code-example-in-java-using-deeplearning4j-f7340f137ca5. [Accessed: 15-Dec-2021].

[36] "Convolutional Neural Networks (CNN) | by Manav Mandal | Medium." [Online]. Available: https://manavmandal.medium.com/convolutional-neural-networks-cnn-d88e7f9329eb. [Accessed: 15-Dec-2021].

[37] G. Litjens *et al.*, "State-of-the-Art Deep Learning in Cardiovascular Image Analysis," *JACC Cardiovasc. Imaging*, vol. 12, no. 8, pp. 1549–1565, Aug. 2019.

[38] W. Huang *et al.*, "Coronary Artery Segmentation by Deep Learning Neural Networks on Computed Tomographic Coronary Angiographic Images," *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, vol. 2018-July, pp. 608–611, Oct. 2018.

[39] D. P. Zhang and D. P. Zhang, "Coronary Artery Segmentation and Motion Modelling," 2011.

[40] Z. Guo *et al.*, "DeepCenterline: a Multi-task Fully Convolutional Network for Centerline Extraction," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11492 LNCS, pp. 441–453, Mar. 2019.

[41] J. Merkow, A. Marsden, D. Kriegman, and Z. Tu, "Dense Volume-to-Volume Vascular Boundary Detection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9902 LNCS, pp. 371–379, May 2016.

[42] Y. Shen, Z. Fang, Y. Gao, N. Xiong, C. Zhong, and X. Tang, "Coronary Arteries Segmentation Based on 3D FCN with Attention Gate and Level Set Function," *IEEE Access*, vol. 7, pp. 42826–42835, 2019.

[43] L. S. Pan, C. W. Li, S. F. Su, S. Y. Tay, Q. V. Tran, and W. P. Chan, "Coronary artery segmentation under class imbalance using a U-Net based architecture on computed tomography angiography images," *Sci. Reports 2021 111*, vol. 11, no. 1, pp. 1–7, Jul. 2021.

[44] M. C. H. Lee, K. Petersen, N. Pawlowski, B. Glocker, and M. Schaap, "TeTrIS: Template Transformer Networks for Image Segmentation with Shape Priors," *IEEE Trans. Med. Imaging*, vol. 38, no. 11, pp. 2596–2606, Nov. 2019.

[45] N. Lessmann *et al.*, "Automatic Calcium Scoring in Low-Dose Chest CT Using Deep Neural Networks with Dilated Convolutions," *IEEE Trans. Med. Imaging*, vol. 37, no. 2, pp. 615–625, Feb. 2018.

[46] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, Nov. 2017.

[47] B. D. de Vos, J. M. Wolterink, T. Leiner, P. A. de Jong, N. Lessmann, and I. Isgum, "Direct Automatic Coronary Calcium Scoring in Cardiac and Chest CT," *IEEE Trans. Med. Imaging*,

vol. 38, no. 9, pp. 2127–2138, Sep. 2019.

[48] C. Cano-Espinosa, G. González, G. R. Washko, M. Cazorla, and R. S. J. Estépar, "Automated Agatston Score Computation in non-ECG Gated CT Scans Using Deep Learning," *Proc. SPIE--the Int. Soc. Opt. Eng.*, vol. 10574, p. 91, Mar. 2018.

[49] M. JayaSree and L. Koteswara Rao, "Survey on - Identification of Coronary Artery Disease using Deep Learning," *Mater. Today Proc.*, Oct. 2020.

[50] S. Candemir *et al.*, "Automated coronary artery atherosclerosis detection and weakly supervised localization on coronary CT angiography with a deep 3-dimensional convolutional neural network," *Comput. Med. Imaging Graph.*, vol. 83, p. 101721, Jul. 2020.

[51] T. Du, X. Liu, H. Zhang, and B. Xu, "Real-time Lesion Detection of Cardiac Coronary Artery Using Deep Neural Networks," *undefined*, pp. 150–154, Nov. 2018.

[52] N. Hampe, J. M. Wolterink, S. G. M. van Velzen, T. Leiner, and I. Išgum, "Machine Learning for Assessment of Coronary Artery Disease in Cardiac CT: A Survey," *Front. Cardiovasc. Med.*, vol. 6, p. 172, Nov. 2019.

[53] K. Lekadir *et al.*, "A Convolutional Neural Network for Automatic Characterization of Plaque Composition in Carotid Ultrasound," *IEEE J. Biomed. Heal. informatics*, vol. 21, no. 1, pp. 48–55, Jan. 2017.

[54] F. Zhao *et al.*, "An automatic multi-class coronary atherosclerosis plaque detection and classification framework," *Med. Biol. Eng. Comput. 2018 571*, vol. 57, no. 1, pp. 245–257, Aug. 2018.

[55] M. A. Zuluaga, D. Hush, E. J. F. Delgado Leyton, M. H. Hoyos, and M. Orkisz, "Learning from Only Positive and Unlabeled Data to Detect Lesions in Vascular CT Images," *undefined*, vol. 6893 LNCS, no. PART 3, pp. 9–16, 2011.

[56] "Heart Disease and Stroke Statistics-2021 Update A Report from the American Heart Association," *Circulation*, pp. E254–E743, 2021.

[57] "Coronary heart disease | Health Knowledge." [Online]. Available: https://www.healthknowledge.org.uk/public-health-textbook/disease-causation-diagnostic/2b-epidemiology-diseases-phs/chronic-diseases/coronary-heart-disease. [Accessed: 08-Dec-2021].

[58] "PubMed." [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/. [Accessed: 15-Dec-2021].

[59] "Home - Grand Challenge." [Online]. Available: https://asoca.grand-challenge.org/. [Accessed: 15-Dec-2021].

[60] "HII - Vitrea Advanced Visualization - Canon Medical Systems Spain." [Online]. Available: https://es.medical.canon/hii-vitrea-advanced-visualization/. [Accessed: 15-Dec-2021].

[61] "IntelliSpace Portal 9.0 | Advanced visual analysis." [Online]. Available: https://www.philips.es/healthcare/product/HC881072/intellispace-portal-solucin-de-visualizacin-avanzada?origin=7____&dmcm=Cj0KCQiAweaNBhDEARIsAJ5hwbcNjXnaNA2Sut1B9Jcun4lDoZnlf2KT_zyc33egkpy2LXfONSWmcnMaAu9bEALw_wcB&gclid=Cj0KCQiAweaNBhDEARIsAJ5hwbcNjXnaNA2Sut1B9Jcun4lDoZnlf2KT_zyc33egkpy2LXfONSWmcnMaAu9bEALw_wcB&gclsrc=aw.ds. [Accessed: 15-Dec-2021].

[62] J. Leipsic, P. Healthcare, and S. Platforms, "Edited and Approved by Acquisition and

Reconstruction Techniques for Coronary CT Angiography."

[63] "CardIQ Fusion | GE Healthcare (Spain)." [Online]. Available: https://www.gehealthcare.es/products/advanced-visualization/all-applications/cardiq-fusion. [Accessed: 15-Dec-2021].

[64] "www.caristo.com." [Online]. Available: https://www.caristo.com/. [Accessed: 03-Feb-2022].

[65] "Transforming the Diagnosis & Management of Coronary Artery Disease | HeartFlow." [Online]. Available: https://www.heartflow.com/. [Accessed: 06-Feb-2022].

[66] "Cleerly, Inc." [Online]. Available: https://cleerlyhealth.com/. [Accessed: 03-Feb-2022].

[67] "CVD Statistics." [Online]. Available: https://ehnheart.org/cvd-statistics.html. [Accessed: 06-Feb-2022].

[68] A. Timmis *et al.*, "European Society of Cardiology: cardiovascular disease statistics 2021," *Eur. Heart J.*, Jan. 2022.

[69] H. J. Chang *et al.*, "Selective Referral Using CCTA Versus Direct Referral for Individuals Referred to Invasive Coronary Angiography for Suspected CAD: A Randomized, Controlled, Open-Label Trial," *JACC Cardiovasc. Imaging*, vol. 12, no. 7, pp. 1303–1312, Jul. 2019.

[70] "The Cost of a Heart Attack - Preventative Diagnostic Center." [Online]. Available: https://www.pdcenterlv.com/blog/the-cost-of-a-heart-attack/. [Accessed: 06-Feb-2022].

[71] "Compare The Cost Of Coronary Angioplasty | Treatment Abroad." [Online]. Available: https://www.treatmentabroad.com/costs/cardiac-surgery/coronary-angioplasty. [Accessed: 06-Feb-2022].

[72] F. Isensee and K. H. Maier-Hein, "An attempt at beating the 3D U-Net," Aug. 2019.

[73] "Understanding Semantic Segmentation with UNET | by Harshall Lamba | Towards Data Science." [Online]. Available: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47. [Accessed: 06-Feb-2022].

[74] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9351, pp. 234–241, May 2015.

[75] "CNN | Introduction to Pooling Layer - GeeksforGeeks." [Online]. Available: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/. [Accessed: 06-Feb-2022].

[76] "(PDF) Speeding up Deep Learning Computational Aspects of Machine Learning." [Online]. Available: https://www.researchgate.net/publication/308414212_Speeding_up_Deep_Learning_Computational_Aspects_of_Machine_Learning. [Accessed: 06-Feb-2022].

[77] "Image Segementation using U-net | Aishwarya Mali." [Online]. Available: https://ashm8206.github.io/2018/03/07/Ultrasound-Image-Segementation-Using-Unet.html. [Accessed: 06-Feb-2022].

[78] "Auto Encoders For Computer Vision | What are Auto Encoders." [Online]. Available: https://www.analyticsvidhya.com/blog/2021/01/auto-encoders-for-computer-vision-an-endless-world-of-possibilities/. [Accessed: 30-Mar-2022].

[79] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” *ICLR 2021*, Oct. 2020.

[80] “Multi-head attention mechanism: ‘queries’, ‘keys’, and ‘values,’ over and over again - Data Science Blog.” [Online]. Available: https://data-science-blog.com/blog/2021/04/07/multi-head-attention-mechanism/. [Accessed: 30-Mar-2022].

[81] H. Cao *et al.*, “Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation.”

[82] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, “Ablation Studies in Artificial Neural Networks,” Jan. 2019.

[83] “Computed Tomography – Products | Canon Medical Systems USA.” [Online]. Available: https://us.medical.canon/products/computed-tomography/. [Accessed: 23-May-2022].

[84] “Philips CT Scanners | Solutions overview | Philips Healthcare.” [Online]. Available: https://www.usa.philips.com/healthcare/solutions/computed-tomography/ct-scanners. [Accessed: 23-May-2022].

[85] W. Yan *et al.*, “MRI Manufacturer Shift and Adaptation: Increasing the Generalizability of Deep Learning Segmentation for MR Images Acquired with Different Scanners,” *https://doi.org/10.1148/ryai.2020190195*, vol. 2, no. 4, p. e190195, Jul. 2020.

[86] “CT Comprehensive Cardiac Analysis (CCA) | Philips Healthcare.” [Online]. Available: https://www.philips.co.uk/healthcare/product/HCAPP006/-ct-comprehensive-cardiac-analysis-cca-. [Accessed: 23-May-2022].

[87] “3D Slicer image computing platform | 3D Slicer.” [Online]. Available: https://www.slicer.org/. [Accessed: 23-May-2022].

[88] H. Luan *et al.*, “Challenges and Future Directions of Big Data and Artificial Intelligence in Education,” *Front. Psychol.*, vol. 11, p. 2748, Oct. 2020.

[89] “Materialise Mimics | 3D Medical Image Processing Software.” [Online]. Available: https://www.materialise.com/en/medical/mimics-innovation-suite/mimics. [Accessed: 23-May-2022].

[90] “GitHub - davidiommi/Pytorch--3D-Medical-Images-Segmentation--SALMON: Segmentation deep learning ALgorithm based on MONai toolbox: single and multi-label segmentation software developed by QIMP team-Vienna.” [Online]. Available: https://github.com/davidiommi/Pytorch--3D-Medical-Images-Segmentation--SALMON. [Accessed: 23-May-2022].

[91] “MONAI - Home.” [Online]. Available: https://monai.io/. [Accessed: 23-May-2022].

[92] MDCG, “Guidance on Qualification and Classification of Software in Regulation (EU) 2017/745 – MDR and Regulation (EU) 2017/746 – IVDR,” pp. 1–28, 2019.

[93] “REGULATION (EU) 2017/ 746 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL - of 5 April 2017 - on in vitro diagnostic medical devices and repealing Directive 98/ 79/ EC and Commission Decision 2010/ 227/ EU,” 2017.

[94] European Comission, “Medical Device?,” 2017.

[95] L. Macomber, “General Safety and Performance Requirements (Annex I) in the New Medical Device Regulation Comparison with the Essential Requirements of the Medical

Device Directive and Active Implantable Device Directive.”

[96] “ISO - IEC 82304-1:2016 - Health software — Part 1: General requirements for product safety.” [Online]. Available: https://www.iso.org/standard/59543.html. [Accessed: 17-Feb-2022].

[97] “CE marking – obtaining the certificate, EU requirements - Your Europe.” [Online]. Available: https://europa.eu/youreurope/business/product-requirements/labels-markings/ce-marking/index_en.htm. [Accessed: 17-Feb-2022].

ANNEX

1. Initial requirements installation.

The computed training in this project required to install tools and libraries as commands in the computer's terminal, as well as a virtual environment before starting with the first training of the ablation study. The procedure that was followed is presented in this section.

1. Anaconda environment set up. This command also installs Python 3.8 due to MONAI works with a Python version.
   Commands:
   ```
   - conda create -n <name_environment> python=3.8
   - conda activate <name_environment>
   ```

2. Install Pytorch for cuda and for the computer's CPU, respectively.
   Commands:
   ```
   - conda install pytorch==1.5.0 torchvision==0.6.0 cudatoolkit=10.1
     -c pytorch
   - conda install pytorch==1.5.0 torchvision==0.6.0 cpuonly -c
     pytorch
   ```

3. Install Github.
   Command:
   ```
   - conda install git pip
   ```

4. Download SALMON libraries from a Github repository.
   Command:
   ```
   - pip install git+https://github.com/davidiommi/MONAI_0_7_0.git
   ```

5. Download the needed libraries one by one.
   Command:
   ```
   - pip install <library>
   ```
   The libraries that were installed are: simpleITK==2.1.0, torchsummaryX, nibabel, pillow, tensorboard, gdown, pytorh-ignite==0.4.4, itk, tqdm, lmdb, psutil, pandas, einops and scikit-image.

2. UNETR algorithm code.

The segmentation algorithm was done using SALMON, an open-source computational toolbox for medical imaging segmentation. The scripts provided by SALMON was overwritten in order to achieve the main objective of the project such as changing directories, eliminating lines of codes that were not needed in our project such as the U-Net network that SALMON offers in order to reduce the computational cost of the algorithm, as well as defining the hyperparameters, among others. They were all computed in the terminal. In the following section, all the Python transcripts used are presented in the same order they were executed in the terminal.

2.1. organize_folder_structure.py

The main function of this Python script was to organize the data in the folder structure (training, validation, and testing) needed for the network. It also resamples and resizes to a previously fixed resolution value all the labels to their corresponding CT image in order to avoid array size conflicts. In order to execute this command, it is important to structure the data set into two main folders, one with all the images and one with all the labels, inside a folder called "Data_folder".

```
import os
import re
import argparse
import SimpleITK as sitk
import numpy as np
import random
from utils import *


if __name__ == "__main__":

    parser = argparse.ArgumentParser()
    parser.add_argument('--
images', default='Data_folder/all_images', help='path to the images')
    parser.add_argument('--
labels', default='./Data_folder/all_labels', help='path to the labels'
)
    parser.add_argument('--
split_val', default= 5, help='number of images for validation')
    parser.add_argument('--
split_test', default= 2, help='number of images for testing')
    parser.add_argument('--
resolution', default=[0.5, 0.5, 1], help='New Resolution to resample t
he data to same spacing')
    parser.add_argument('--
smooth', default=False, help='Set True if you want to smooth a bit the
 binary mask')
    args = parser.parse_args()

    list_images = lstFiles(args.images)
    list_labels = lstFiles(args.labels)

    mapIndexPosition = list(zip(list_images, list_labels))  #shuffle o
rder list
    random.shuffle(mapIndexPosition)
    list_images, list_labels = zip(*mapIndexPosition)

    os.mkdir('./Data_folder/images')
    os.mkdir('./Data_folder/labels')

    # 1
    if not os.path.isdir('./Data_folder/images/train'):
        os.mkdir('./Data_folder/images/train/')
    # 2
    if not os.path.isdir('./Data_folder/images/val'):
        os.mkdir('./Data_folder/images/val')

    # 3
    if not os.path.isdir('./Data_folder/images/test'):
        os.mkdir('./Data_folder/images/test')

    # 4
    if not os.path.isdir('./Data_folder/labels/train'):
        os.mkdir('./Data_folder/labels/train')

    # 5
    if not os.path.isdir('./Data_folder/labels/val'):
        os.mkdir('./Data_folder/labels/val')

    # 6
    if not os.path.isdir('./Data_folder/labels/test'):
        os.mkdir('./Data_folder/labels/test')

    for i in range(len(list_images)-
int(args.split_test + args.split_val)):

        a = list_images[int(args.split_test + args.split_val)+i]
        b = list_labels[int(args.split_test + args.split_val)+i]

        print('train',i, a,b)

        label = sitk.ReadImage(b)
        image = sitk.ReadImage(a)

        image = resample_sitk_image(image, spacing=args.resolution, in
terpolator='linear', fill_value=0)
        image, label = uniform_img_dimensions(image, label, nearest=Tr
ue)
        if args.smooth is True:
            label = gaussian2(label)

        image_directory = os.path.join('./Data_folder/images/train', f
"image{i:d}.nii")
        label_directory = os.path.join('./Data_folder/labels/train', f
"label{i:d}.nii")

        sitk.WriteImage(image, image_directory)
        sitk.WriteImage(label, label_directory)

    for i in range(int(args.split_val)):

        a = list_images[int(args.split_test)+i]
        b = list_labels[int(args.split_test)+i]

        print('val',i, a,b)

        label = sitk.ReadImage(b)
        image = sitk.ReadImage(a)
        image = resample_sitk_image(image, spacing=args.resolution, in
terpolator='linear', fill_value=0)
        image, label = uniform_img_dimensions(image, label, nearest=Tr
ue)
        if args.smooth is True:
            label = gaussian2(label)

        image_directory = os.path.join('./Data_folder/images/val', f"i
mage{i:d}.nii")
        label_directory = os.path.join('./Data_folder/labels/val', f"l
abel{i:d}.nii")

        sitk.WriteImage(image, image_directory)
        sitk.WriteImage(label, label_directory)

    for i in range(int(args.split_test)):

        a = list_images[i]
        b = list_labels[i]

        print('test',i,a,b)

        label = sitk.ReadImage(b)
        image = sitk.ReadImage(a)

        image = resample_sitk_image(image, spacing=args.resolution, in
terpolator='linear', fill_value=0)
        image, label = uniform_img_dimensions(image, label, nearest=Tr
ue)
        if args.smooth is True:
            label = gaussian2(label)

        image_directory = os.path.join('./Data_folder/images/test', f"
image{i:d}.nii")
        label_directory = os.path.join('./Data_folder/labels/test', f"
label{i:d}.nii")

        sitk.WriteImage(image, image_directory)
        sitk.WriteImage(label, label_directory)
```

2.2. init.py

In this command all the basic parameters are defined such as the name of the folders as well as the network hyperparameters which are number of input and output channels, the batch size, the patch size, the number of epochs, the number of SA blocks of the UNETR, the defined learning rate, among others. It is important to overwrite this script before every training in order to set the parameters and start the training on the correct data.

```
import argparse
import os


class Options():

    """This class defines options used during both training and tes
t time."""

    def __init__(self):
        """Reset the class; indicates the class hasn't been initail
ized"""
        self.initialized = False

    def initialize(self, parser):

        # basic parameters
        parser.add_argument('--
images_folder', type=str, default='./Data_folder/images')
        parser.add_argument('--
labels_folder', type=str, default='./Data_folder/labels')
        parser.add_argument('--
increase_factor_data',  default=1, help='Increase data number per e
poch')
        parser.add_argument('--preload', type=str, default=None)
        parser.add_argument('--
gpu_ids', type=str, default='0', help='gpu ids: e.g. 0  0,1,2, 0,2.
 use -1 for CPU')
        parser.add_argument('--
workers', default=0, type=int, help='number of data loading workers
')

        # dataset parameters
        parser.add_argument('--
network', default='unetr', help='nnunet, unetr')
        parser.add_argument('--
patch_size', default=(256, 256, 32), help='Size of the patches extr
acted from the image')
        parser.add_argument('--
spacing', default=[0.5,0.5,1], help='Original Resolution')
        parser.add_argument('--
resolution', default=None, help='New Resolution, if you want to res
ample the data in training. I suggest to resample in organize_folde
r_structure.py, otherwise in train resampling is slower')
        parser.add_argument('--
batch_size', type=int, default=2, help='batch size, depends on your
 machine')
        parser.add_argument('--
in_channels', default=1, type=int, help='Channels of the input')
```

```
        parser.add_argument('--
out_channels', default=1, type=int, help='Channels of the output')

        # training parameters
        parser.add_argument('--
epochs', default=150, help='Number of epochs')
        parser.add_argument('--
lr', default=0.1, help='Learning rate')
        parser.add_argument('--benchmark', default=True)

        # Inference
        # This is just a trick to make the predict script working,
do not touch it now for the training.
        parser.add_argument('--
result', default=None, help='Keep this empty and go to predict_sing
le_image script')
        parser.add_argument('--
weights', default=None, help='Keep this empty and go to predict_sin
gle_image script')

        self.initialized = True
        return parser

    def parse(self):
        if not self.initialized:
            parser = argparse.ArgumentParser(formatter_class=argpar
se.ArgumentDefaultsHelpFormatter)
            parser = self.initialize(parser)
        opt = parser.parse_args()
        # set gpu ids
        if opt.gpu_ids != '-1':
            os.environ["CUDA_VISIBLE_DEVICES"] = opt.gpu_ids
        return opt
```

2.3. train.py

The main objective of this Python script is to run the training. After the first epoch, a folder called "runs" is created with all the weights of the training. Tensorboard, a Pytorch tool that provides measurement and visualizations during a deep learning workflow, is available to monitor the training. It enables tracking the training metrics like the algorithm loss and Dice, also known as the bias, over time, as well as, displaying a gif with the image inference, which is the resulting binary mask after the coronary artery segmentation in every CT slice. This script needs the *networks.py* script, where the UNETR architecture is coded for the medical imaging segmentation. It is not necessary to run the last script in the terminal but they have to be in the same directory.

```
from init import Options
from networks import build_net, update_learning_rate, build_UNETR
```

```
# from networks import build_net
import logging
import os
import sys
import tempfile
from glob import glob

import nibabel as nib
import numpy as np
import torch
from torch.utils.data import DataLoader
from torch.utils.tensorboard import SummaryWriter

import monai
from monai.data import create_test_image_3d, list_data_collate, dec
ollate_batch
from monai.inferers import sliding_window_inference
from monai.metrics import DiceMetric
from monai.transforms import (EnsureType, Compose, LoadImaged, AddC
hanneld, Transpose,Activations,AsDiscrete, RandGaussianSmoothd, Cro
pForegroundd, SpatialPadd,
                              ScaleIntensityd, ToTensord, RandSpati
alCropd, Rand3DElasticd, RandAffined, RandZoomd,
    Spacingd, Orientationd, Resized, ThresholdIntensityd, RandShift
Intensityd, BorderPadd, RandGaussianNoised, RandAdjustContrastd,Nor
malizeIntensityd,RandFlipd)

from monai.visualize import plot_2d_or_3d_image

def main():
    opt = Options().parse()
    # monai.config.print_config()
    logging.basicConfig(stream=sys.stdout, level=logging.INFO)

    # check gpus
    if opt.gpu_ids != '-1':
        num_gpus = len(opt.gpu_ids.split(','))
    else:
        num_gpus = 0
    print('number of GPU:', num_gpus)

    # Data loader creation
    # train images
    train_images = sorted(glob(os.path.join(opt.images_folder, 'tra
in', 'image*.nii')))
    train_segs = sorted(glob(os.path.join(opt.labels_folder, 'train
', 'label*.nii')))
```

```
    train_images_for_dice = sorted(glob(os.path.join(opt.images_fol
der, 'train', 'image*.nii')))
    train_segs_for_dice = sorted(glob(os.path.join(opt.labels_folde
r, 'train', 'label*.nii')))

    # validation images
    val_images = sorted(glob(os.path.join(opt.images_folder, 'val',
 'image*.nii')))
    val_segs = sorted(glob(os.path.join(opt.labels_folder, 'val', '
label*.nii')))

    # test images
    test_images = sorted(glob(os.path.join(opt.images_folder, 'test
', 'image*.nii')))
    test_segs = sorted(glob(os.path.join(opt.labels_folder, 'test',
 'label*.nii')))

    # augment the data list for training
    for i in range(int(opt.increase_factor_data)):

        train_images.extend(train_images)
        train_segs.extend(train_segs)

    print('Number of training patches per epoch:', len(train_images
))
    print('Number of training images per epoch:', len(train_images_
for_dice))
    print('Number of validation images per epoch:', len(val_images)
)
    print('Number of test images per epoch:', len(test_images))

    # Creation of data directories for data_loader

    train_dicts = [{'image': image_name, 'label': label_name}
                   for image_name, label_name in zip(train_images, t
rain_segs)]

    train_dice_dicts = [{'image': image_name, 'label': label_name}
                        for image_name, label_name in zip(train_images_f
or_dice, train_segs_for_dice)]

    val_dicts = [{'image': image_name, 'label': label_name}
                        for image_name, label_name in zip(val_images, va
l_segs)]

    test_dicts = [{'image': image_name, 'label': label_name}
                  for image_name, label_name in zip(test_images, tes
t_segs)]
```

```
    # Transforms list

    if opt.resolution is not None:
        train_transforms = [
            LoadImaged(keys=['image', 'label']),
            AddChanneld(keys=['image', 'label']),
            # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),  # CT HU filter
            # ThresholdIntensityd(keys=['image'], threshold=215, ab
ove=False, cval=215),
            CropForegroundd(keys=['image', 'label'], source_key='im
age'),               # crop CropForeground

            NormalizeIntensityd(keys=['image']),
                       # augmentation
            ScaleIntensityd(keys=['image']),
                       # intensity
            Spacingd(keys=['image', 'label'], pixdim=opt.resolution
, mode=('bilinear', 'nearest')),  # resolution

            RandFlipd(keys=['image', 'label'], prob=0.15, spatial_a
xis=1),
            RandFlipd(keys=['image', 'label'], prob=0.15, spatial_a
xis=0),
            RandFlipd(keys=['image', 'label'], prob=0.15, spatial_a
xis=2),
            RandAffined(keys=['image', 'label'], mode=('bilinear',
'nearest'), prob=0.1,
                        rotate_range=(np.pi / 36, np.pi / 36, np.pi
 * 2), padding_mode="zeros"),
            RandAffined(keys=['image', 'label'], mode=('bilinear',
'nearest'), prob=0.1,
                        rotate_range=(np.pi / 36, np.pi / 2, np.pi
/ 36), padding_mode="zeros"),
            RandAffined(keys=['image', 'label'], mode=('bilinear',
'nearest'), prob=0.1,
                        rotate_range=(np.pi / 2, np.pi / 36, np.pi
/ 36), padding_mode="zeros"),
            Rand3DElasticd(keys=['image', 'label'], mode=('bilinear
', 'nearest'), prob=0.1,
                           sigma_range=(5, 8), magnitude_range=(100
, 200), scale_range=(0.15, 0.15, 0.15),
                           padding_mode="zeros"),
            RandGaussianSmoothd(keys=["image"], sigma_x=(0.5, 1.15)
, sigma_y=(0.5, 1.15), sigma_z=(0.5, 1.15), prob=0.1,),
            RandAdjustContrastd(keys=['image'], gamma=(0.5, 2.5), p
rob=0.1),
            RandGaussianNoised(keys=['image'], prob=0.1, mean=np.ra
ndom.uniform(0, 0.5), std=np.random.uniform(0, 15)),
            RandShiftIntensityd(keys=['image'], offsets=np.random.u
niform(0,0.3), prob=0.1),

            SpatialPadd(keys=['image', 'label'], spatial_size=opt.p
atch_size, method= 'end'),  # pad if the image is smaller than patc
h
            RandSpatialCropd(keys=['image', 'label'], roi_size=opt.
patch_size, random_size=False),
            ToTensord(keys=['image', 'label'])
        ]

        val_transforms = [
            LoadImaged(keys=['image', 'label']),
            AddChanneld(keys=['image', 'label']),
            # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),
            # ThresholdIntensityd(keys=['image'], threshold=215, ab
ove=False, cval=215),
            CropForegroundd(keys=['image', 'label'], source_key='im
age'),                    # crop CropForeground

            NormalizeIntensityd(keys=['image']),
                   # intensity
            ScaleIntensityd(keys=['image']),
            Spacingd(keys=['image', 'label'], pixdim=opt.resolution
, mode=('bilinear', 'nearest')),   # resolution

            SpatialPadd(keys=['image', 'label'], spatial_size=opt.p
atch_size, method= 'end'),  # pad if the image is smaller than patc
h
            ToTensord(keys=['image', 'label'])
        ]
    else:
        train_transforms = [
            LoadImaged(keys=['image', 'label']),
            AddChanneld(keys=['image', 'label']),
            # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),
            # ThresholdIntensityd(keys=['image'], threshold=215, ab
ove=False, cval=215),
            CropForegroundd(keys=['image', 'label'], source_key='im
age'),                # crop CropForeground

            NormalizeIntensityd(keys=['image']),
                       # augmentation
            ScaleIntensityd(keys=['image']),
                       # intensity
            RandFlipd(keys=['image', 'label'], prob=0.15, spatial_a
xis=1),
            RandFlipd(keys=['image', 'label'], prob=0.15, spatial_a
xis=0),
            RandFlipd(keys=['image', 'label'], prob=0.15, spatial_a
xis=2),
            RandAffined(keys=['image', 'label'], mode=('bilinear',
'nearest'), prob=0.1,
                        rotate_range=(np.pi / 36, np.pi / 36, np.pi
 * 2), padding_mode="zeros"),
            RandAffined(keys=['image', 'label'], mode=('bilinear',
'nearest'), prob=0.1,
                        rotate_range=(np.pi / 36, np.pi / 2, np.pi
/ 36), padding_mode="zeros"),
            RandAffined(keys=['image', 'label'], mode=('bilinear',
'nearest'), prob=0.1,
                        rotate_range=(np.pi / 2, np.pi / 36, np.pi
/ 36), padding_mode="zeros"),
            Rand3DElasticd(keys=['image', 'label'], mode=('bilinear
', 'nearest'), prob=0.1,
                           sigma_range=(5, 8), magnitude_range=(100
, 200), scale_range=(0.15, 0.15, 0.15),
                           padding_mode="zeros"),
            RandGaussianSmoothd(keys=["image"], sigma_x=(0.5, 1.15)
, sigma_y=(0.5, 1.15), sigma_z=(0.5, 1.15), prob=0.1,),
            RandAdjustContrastd(keys=['image'], gamma=(0.5, 2.5), p
rob=0.1),
            RandGaussianNoised(keys=['image'], prob=0.1, mean=np.ra
ndom.uniform(0, 0.5), std=np.random.uniform(0, 1)),
            RandShiftIntensityd(keys=['image'], offsets=np.random.u
niform(0,0.3), prob=0.1),

            SpatialPadd(keys=['image', 'label'], spatial_size=opt.p
atch_size, method= 'end'),  # pad if the image is smaller than patc
h
            RandSpatialCropd(keys=['image', 'label'], roi_size=opt.
patch_size, random_size=False),
            ToTensord(keys=['image', 'label'])
        ]

        val_transforms = [
            LoadImaged(keys=['image', 'label']),
            AddChanneld(keys=['image', 'label']),
            # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),
            # ThresholdIntensityd(keys=['image'], threshold=215, ab
ove=False, cval=215),
            CropForegroundd(keys=['image', 'label'], source_key='im
age'),                      # crop CropForeground
```

```
            NormalizeIntensityd(keys=['image']),
                   # intensity
            ScaleIntensityd(keys=['image']),

            SpatialPadd(keys=['image', 'label'], spatial_size=opt.p
atch_size, method= 'end'),  # pad if the image is smaller than patc
h
            ToTensord(keys=['image', 'label'])
        ]

    train_transforms = Compose(train_transforms)
    val_transforms = Compose(val_transforms)

    # create a training data loader
    check_train = monai.data.Dataset(data=train_dicts, transform=tr
ain_transforms)
    train_loader = DataLoader(check_train, batch_size=opt.batch_siz
e, shuffle=True, collate_fn=list_data_collate, num_workers=opt.work
ers, pin_memory=False)

    # create a training_dice data loader
    check_val = monai.data.Dataset(data=train_dice_dicts, transform
=val_transforms)
    train_dice_loader = DataLoader(check_val, batch_size=1, num_wor
kers=opt.workers, collate_fn=list_data_collate, pin_memory=False)

    # create a validation data loader
    check_val = monai.data.Dataset(data=val_dicts, transform=val_tr
ansforms)
    val_loader = DataLoader(check_val, batch_size=1, num_workers=op
t.workers, collate_fn=list_data_collate, pin_memory=False)

    # create a validation data loader
    check_val = monai.data.Dataset(data=test_dicts, transform=val_t
ransforms)
    test_loader = DataLoader(check_val, batch_size=1, num_workers=o
pt.workers, collate_fn=list_data_collate, pin_memory=False)

    # build the network
    if opt.network is 'nnunet':
        net = build_net()   # nn build_net
    elif opt.network is 'unetr':
        net = build_UNETR() # UneTR
    net.cuda()

    if num_gpus > 1:
        net = torch.nn.DataParallel(net)
    if opt.preload is not None:
        net.load_state_dict(torch.load(opt.preload))

    dice_metric = DiceMetric(include_background=True, reduction="me
an", get_not_nans=False)
    post_trans = Compose([EnsureType(), Activations(sigmoid=True),
AsDiscrete(threshold_values=True)])

    loss_function = monai.losses.DiceCELoss(sigmoid=True)
    torch.backends.cudnn.benchmark = opt.benchmark


    if opt.network is 'nnunet':

        optim = torch.optim.SGD(net.parameters(), lr=opt.lr, moment
um=0.99, weight_decay=3e-5, nesterov=True,)
        net_scheduler = torch.optim.lr_scheduler.LambdaLR(optim, lr
_lambda=lambda epoch: (1 - epoch / opt.epochs) ** 0.9)

    elif opt.network is 'unetr':

        optim = torch.optim.AdamW(net.parameters(), lr=1e-
4, weight_decay=1e-5)

    # start a typical PyTorch training
    val_interval = 1
    best_metric = -1
    best_metric_epoch = -1
    epoch_loss_values = list()
    writer = SummaryWriter()
    for epoch in range(opt.epochs):
        print("-" * 10)
        print(f"epoch {epoch + 1}/{opt.epochs}")
        net.train()
        epoch_loss = 0
        step = 0
        for batch_data in train_loader:
            step += 1
            inputs, labels = batch_data["image"].cuda(), batch_data
["label"].cuda()
            optim.zero_grad()
            outputs = net(inputs)
            loss = loss_function(outputs, labels)
            loss.backward()
            optim.step()
            epoch_loss += loss.item()
            epoch_len = len(check_train) // train_loader.batch_size
            print(f"{step}/{epoch_len}, train_loss: {loss.item():.4
f}")
```

```
            writer.add_scalar("train_loss", loss.item(), epoch_len
* epoch + step)
        epoch_loss /= step
        epoch_loss_values.append(epoch_loss)
        print(f"epoch {epoch + 1} average loss: {epoch_loss:.4f}")
        if opt.network is 'nnunet':
            update_learning_rate(net_scheduler, optim)

        if (epoch + 1) % val_interval == 0:
            net.eval()
            with torch.no_grad():

                def plot_dice(images_loader):

                    val_images = None
                    val_labels = None
                    val_outputs = None
                    for data in images_loader:
                        val_images, val_labels = data["image"].cuda
(), data["label"].cuda()
                        roi_size = opt.patch_size
                        sw_batch_size = 4
                        val_outputs = sliding_window_inference(val_
images, roi_size, sw_batch_size, net)
                        val_outputs = [post_trans(i) for i in decol
late_batch(val_outputs)]
                        dice_metric(y_pred=val_outputs, y=val_label
s)

                    # aggregate the final mean dice result
                    metric = dice_metric.aggregate().item()
                    # reset the status for next validation round
                    dice_metric.reset()

                    return metric, val_images, val_labels, val_outp
uts

                metric, val_images, val_labels, val_outputs = plot_
dice(val_loader)

                # Save best model
                if metric > best_metric:
                    best_metric = metric
                    best_metric_epoch = epoch + 1
                    torch.save(net.state_dict(), "best_metric_model
.pth")
                    print("saved new best metric model")
                metric_train, train_images, train_labels, train_out
puts = plot_dice(train_dice_loader)
                metric_test, test_images, test_labels, test_outputs
 = plot_dice(test_loader)

                # Logger bar
                print(
                    "current epoch: {} Training dice: {:.4f} Valida
tion dice: {:.4f} Testing dice: {:.4f} Best Validation dice: {:.4f}
 at epoch {}".format(
                        epoch + 1, metric_train, metric, metric_tes
t, best_metric, best_metric_epoch
                    )
                )

                writer.add_scalar("Mean_epoch_loss", epoch_loss, ep
och + 1)
                writer.add_scalar("Testing_dice", metric_test, epoc
h + 1)
                writer.add_scalar("Training_dice", metric_train, ep
och + 1)
                writer.add_scalar("Validation_dice", metric, epoch
+ 1)
                # plot the last model output as GIF image in Tensor
Board with the corresponding image and label
                # val_outputs = (val_outputs.sigmoid() >= 0.5).floa
t()
                plot_2d_or_3d_image(val_images, epoch + 1, writer,
index=0, tag="validation image")
                plot_2d_or_3d_image(val_labels, epoch + 1, writer,
index=0, tag="validation label")
                plot_2d_or_3d_image(val_outputs, epoch + 1, writer,
 index=0, tag="validation inference")
                plot_2d_or_3d_image(test_images, epoch + 1, writer,
 index=0, tag="test image")
                plot_2d_or_3d_image(test_labels, epoch + 1, writer,
 index=0, tag="test label")
                plot_2d_or_3d_image(test_outputs, epoch + 1, writer
, index=0, tag="test inference")

    print(f"train completed, best_metric: {best_metric:.4f} at epoc
h: {best_metric_epoch}")
    writer.close()


if __name__ == "__main__":
    main()
```

2.4. predict_single_image.py

This Python script is used to obtain the inference of a new image, which it is a binary mask of the coronary arteries. It can have different size and resolution from the training. The image will be resampled and the output will be the binary mask in a NIfTI format with the same size and resolution of the source image.

```
from utils import *
import timeit
import argparse
from networks import build_net, build_UNETR
from monai.inferers import sliding_window_inference
from monai.metrics import DiceMetric
from monai.data import NiftiSaver, create_test_image_3d, list_data_
collate, decollate_batch
from monai.transforms import (EnsureType, Compose, LoadImaged, AddC
hanneld, Transpose,Activations,AsDiscrete, RandGaussianSmoothd, Cro
pForegroundd, SpatialPadd,
                              ScaleIntensityd, ToTensord, RandSpati
alCropd, Rand3DElasticd, RandAffined, RandZoomd,
                              Spacingd, Orientationd, Resized, Thre
sholdIntensityd, RandShiftIntensityd, BorderPadd, RandGaussianNoise
d, RandAdjustContrastd,NormalizeIntensityd,RandFlipd)


def segment(image, label, result, weights, resolution, patch_size,
network, gpu_ids):

    logging.basicConfig(stream=sys.stdout, level=logging.INFO)

    if label is not None:
        uniform_img_dimensions_internal(image, label, True)
        files = [{"image": image, "label": label}]
    else:
        files = [{"image": image}]

    # original size, size after crop_background, cropped roi coordi
nates, cropped resampled roi size
    original_shape, crop_shape, coord1, coord2, resampled_size, ori
ginal_resolution = statistics_crop(image, resolution)

    # -------------------------------

    if label is not None:
        if resolution is not None:

            val_transforms = Compose([
                LoadImaged(keys=['image', 'label']),
                AddChanneld(keys=['image', 'label']),
                # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),  # Threshold CT
                # ThresholdIntensityd(keys=['image'], threshold=215
, above=False, cval=215),
                CropForegroundd(keys=['image', 'label'], source_key
='image'),  # crop CropForeground

                NormalizeIntensityd(keys=['image']),  # intensity
                ScaleIntensityd(keys=['image']),
                Spacingd(keys=['image', 'label'], pixdim=resolution
, mode=('bilinear', 'nearest')),  # resolution

                SpatialPadd(keys=['image', 'label'], spatial_size=p
atch_size, method= 'end'),
                ToTensord(keys=['image', 'label'])])
        else:

            val_transforms = Compose([
                LoadImaged(keys=['image', 'label']),
                AddChanneld(keys=['image', 'label']),
                # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),  # Threshold CT
                # ThresholdIntensityd(keys=['image'], threshold=215
, above=False, cval=215),
                CropForegroundd(keys=['image', 'label'], source_key
='image'),  # crop CropForeground

                NormalizeIntensityd(keys=['image']),  # intensity
                ScaleIntensityd(keys=['image']),

                SpatialPadd(keys=['image', 'label'], spatial_size=p
atch_size, method='end'),  # pad if the image is smaller than patch
                ToTensord(keys=['image', 'label'])])

    else:
        if resolution is not None:

            val_transforms = Compose([
                LoadImaged(keys=['image']),
                AddChanneld(keys=['image']),
                # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),  # Threshold CT
                # ThresholdIntensityd(keys=['image'], threshold=215
, above=False, cval=215),
                CropForegroundd(keys=['image'], source_key='image')
,  # crop CropForeground

                NormalizeIntensityd(keys=['image']),  # intensity
                ScaleIntensityd(keys=['image']),
                Spacingd(keys=['image'], pixdim=resolution, mode=('
bilinear')),  # resolution

                SpatialPadd(keys=['image'], spatial_size=patch_size
, method= 'end'),  # pad if the image is smaller than patch
                ToTensord(keys=['image'])])
        else:

            val_transforms = Compose([
                LoadImaged(keys=['image']),
                AddChanneld(keys=['image']),
                # ThresholdIntensityd(keys=['image'], threshold=-
135, above=True, cval=-135),  # Threshold CT
                # ThresholdIntensityd(keys=['image'], threshold=215
, above=False, cval=215),
                CropForegroundd(keys=['image'], source_key='image')
,  # crop CropForeground

                NormalizeIntensityd(keys=['image']),  # intensity
                ScaleIntensityd(keys=['image']),

                SpatialPadd(keys=['image'], spatial_size=patch_size
, method='end'), # pad if the image is smaller than patch
                ToTensord(keys=['image'])])

    val_ds = monai.data.Dataset(data=files, transform=val_transform
s)
    val_loader = DataLoader(val_ds, batch_size=1, num_workers=0, co
llate_fn=list_data_collate, pin_memory=False)

    dice_metric = DiceMetric(include_background=True, reduction="me
an", get_not_nans=False)
    post_trans = Compose([EnsureType(), Activations(sigmoid=True),
AsDiscrete(threshold_values=True)])

    if gpu_ids != '-1':

        # try to use all the available GPUs
        os.environ['CUDA_VISIBLE_DEVICES'] = gpu_ids
        device = torch.device("cuda" if torch.cuda.is_available() e
lse "cpu")

    else:
        device = torch.device("cpu")

    # build the network
    if network == 'nnunet':
        net = build_net()  # nn build_net
    elif network == 'unetr':
        net = build_UNETR() # UneTR

    net = net.to(device)

    if gpu_ids == '-1':

        net.load_state_dict(new_state_dict_cpu(weights))

    else:

        net.load_state_dict(new_state_dict(weights))

    # define sliding window size and batch size for windows inferen
ce
    roi_size = patch_size
    sw_batch_size = 2

    net.eval()
    with torch.no_grad():

        if label is None:
            for val_data in val_loader:
                val_images = val_data["image"].to(device)
                val_outputs = sliding_window_inference(val_images,
roi_size, sw_batch_size, net)
                val_outputs = [post_trans(i) for i in decollate_bat
ch(val_outputs)]

        else:
            for val_data in val_loader:
                val_images, val_labels = val_data["image"].to(devic
e), val_data["label"].to(device)
                val_outputs = sliding_window_inference(val_images,
roi_size, sw_batch_size, net)
                val_outputs = [post_trans(i) for i in decollate_bat
ch(val_outputs)]
                dice_metric(y_pred=val_outputs, y=val_labels)

            metric = dice_metric.aggregate().item()
            print("Evaluation Metric (Dice):", metric)

        result_array = val_outputs[0].squeeze().data.cpu().numpy()
        # Remove the pad if the image was smaller than the patch in
 some directions
        result_array = result_array[0:resampled_size[0],0:resampled
_size[1],0:resampled_size[2]]

        # resample back to the original resolution
        if resolution is not None:

            result_array_np = np.transpose(result_array, (2, 1, 0))
            result_array_temp = sitk.GetImageFromArray(result_array
_np)
            result_array_temp.SetSpacing(resolution)

            # save temporary label
            writer = sitk.ImageFileWriter()
            writer.SetFileName('temp_seg.nii')
            writer.Execute(result_array_temp)

            files = [{"image": 'temp_seg.nii'}]

            files_transforms = Compose([
                LoadImaged(keys=['image']),
                AddChanneld(keys=['image']),
                Spacingd(keys=['image'], pixdim=original_resolution
, mode=('nearest')),
                Resized(keys=['image'], spatial_size=crop_shape, mo
de=('nearest')),
            ])

            files_ds = Dataset(data=files, transform=files_transfor
ms)
            files_loader = DataLoader(files_ds, batch_size=1, num_w
orkers=0)

            for files_data in files_loader:
                files_images = files_data["image"]

                res = files_images.squeeze().data.numpy()

            result_array = np.rint(res)

            os.remove('./temp_seg.nii')

        # recover the cropped background before saving the image
        empty_array = np.zeros(original_shape)
        empty_array[coord1[0]:coord2[0],coord1[1]:coord2[1],coord1[
2]:coord2[2]] = result_array

        result_seg = from_numpy_to_itk(empty_array, image)

        # save label
        writer = sitk.ImageFileWriter()
        writer.SetFileName(result)
        writer.Execute(result_seg)
        print("Saved Result at:", str(result))
```

```
if __name__ == "__main__":

    start = timeit.default_timer()
    parser = argparse.ArgumentParser()
    parser.add_argument("--
image", type=str, default='./Data_folder/images/test/image0.nii', h
elp='source image' )
    parser.add_argument("--
label", type=str, default='./Data_folder/labels/test/label0.nii', h
elp='source label, if you want to compute dice. None for new case')
    parser.add_argument("--
result", type=str, default='./Data_folder/inferences/inf0_b2_lr01.n
ii', help='path to the .nii result to save')
    parser.add_argument("--
weights", type=str, default='./best_metric_b2_lr01.pth', help='netw
ork weights to load')
    parser.add_argument("--
resolution", default=[0.5, 0.5, 1], help='Resolution used in traini
ng phase')
    parser.add_argument("--
patch_size", type=int, nargs=3, default=(256, 256, 32), help="Input
 dimension for the generator, same of training")
    parser.add_argument('--
network', default='unetr', help='nnunet, unetr')
    parser.add_argument('--
gpu_ids', type=str, default='0', help='gpu ids: e.g. 0  0,1,2, 0,2.
 use -1 for CPU')
    args = parser.parse_args()

    segment(args.image, args.label, args.result, args.weights, args
.resolution, args.patch_size, args.network, args.gpu_ids)
    stop = timeit.default_timer()
    print('Time: ', stop - start)
```

2.5. Other needed Python scripts

The scripts above need other scripts called *check_loader_patches.py*, *utils.py* and *networks.py*. They were not overwritten and they can be found in SALMON Github [90].

3. Analysis of the segmentation algorithm.

In this section, the performance of the UNETR algorithm is evaluated and analysed by comparing the inference obtained with the ground truth label and with the segmentation obtained using the nnU-Net. The drawbacks, the positive aspects to highlight and the differences with the nnU-Net will be explain in detail.

In the first case presented in *Figure A1*, we can observe that there is a stop in the anterior descendent coronary artery common trunk. Several intra-pericardial structures are also segmented such as the left atrium, shown in *Figure A2*. Despite these limitations, the coronary arteries are well segmented and the root of the aorta is better defined than the ground truth, where we can perfectly observe the aortic sinus. Comparing the UNETR inference with the nnU-Net one we can conclude that the segmentation is less smooth in the former, it has more artefacts and there are fewer small branches segmented.

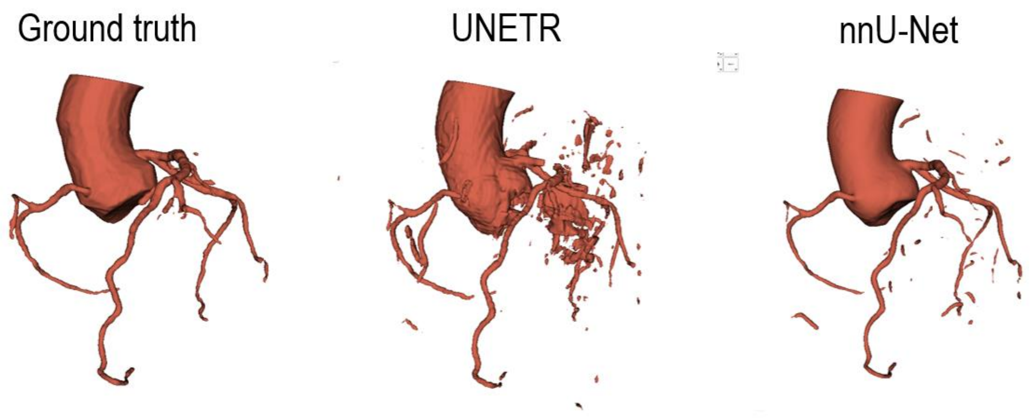


Figure A1. Comparison between the ground truth, the UNETR and the nnU-Net inference.

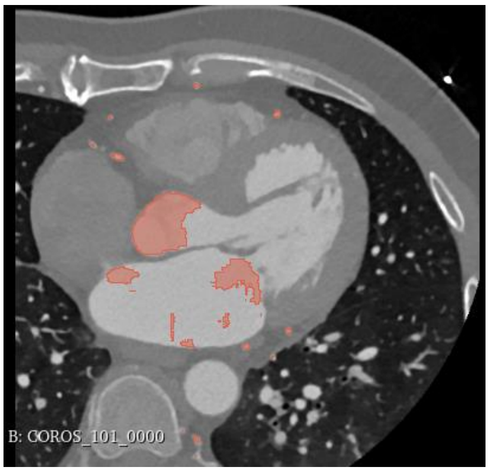


Figure A2. CCTA image with the coronary artery, the aorta and the left atrium highlighted in red in the axial plane.

Figure A3 shows the inferences in the second case. There exist a small stop in the right coronary sinus (shown in *Figure A4*) and there is a huge presence of intra- and extra-pericardial artefacts. However, the main shape of the coronary arteries is well defined and the intra-coronary calcium is segmented (shown in *Figure A5*). There is more artefacts, a worse segmentation of the circumflex artery and a rougher segmentation in the UNETR inference than in the nnU-Net one.

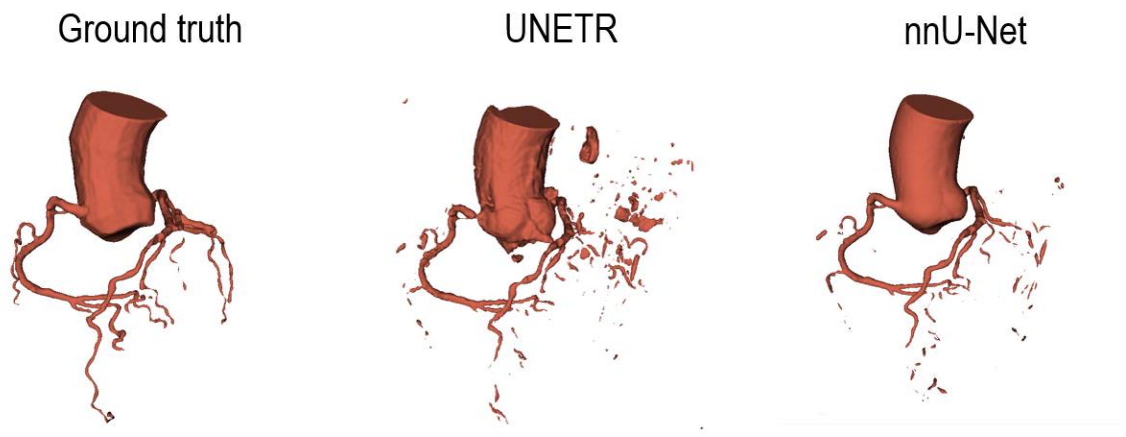


Figure A3. Comparison between the ground truth, the UNETR and the nnU-Net inference.

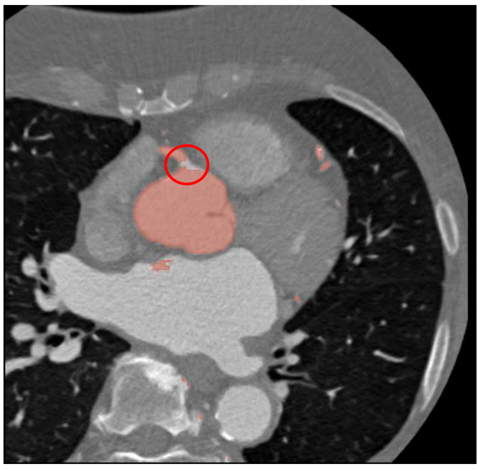


Figure A4. Segmentation stop in the right coronary sinus shown in a CCTA image in the axial plane.

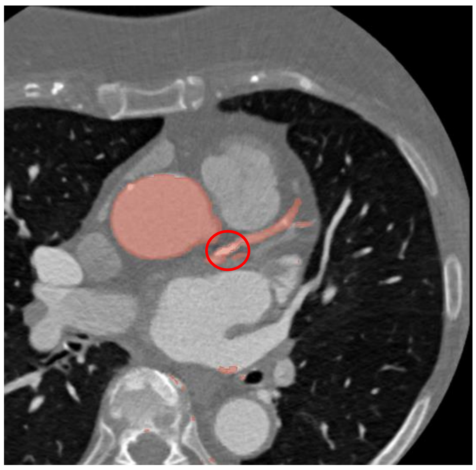


Figure A5. Segmentation of the calcium in the common trunk of the left coronary artery viewed in a CCTA image in the axial plane.

We can observe different problems in *Figure A6*. First, the right and the left atriums as well as the right ventricle are segmented because there is iodine contrast in these structures, as we can see in the CT image presented in *Figure A7*.

Moreover, there is a stop in the anterior descending because the calcium is not well recognised (shown in *Figure A8)* and there right coronary sinus is not well defined. Nevertheless, in this case, there exist less artifacts produced by the segmentation of other coronary vessels. Comparing the UNETR inference with the nnU-Net one we can conclude that there exist a worse segmentation of the main coronary branches in the former.

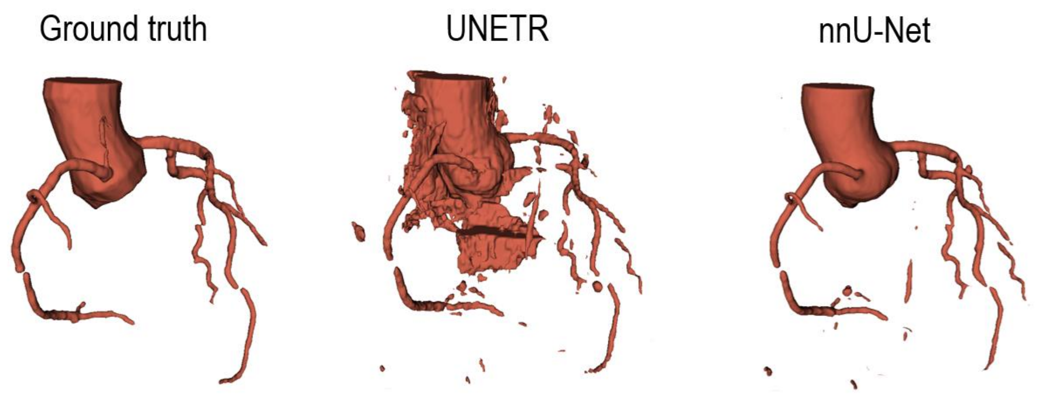


Figure A6. Comparison between the ground truth, the UNETR and the nnU-Net inference.

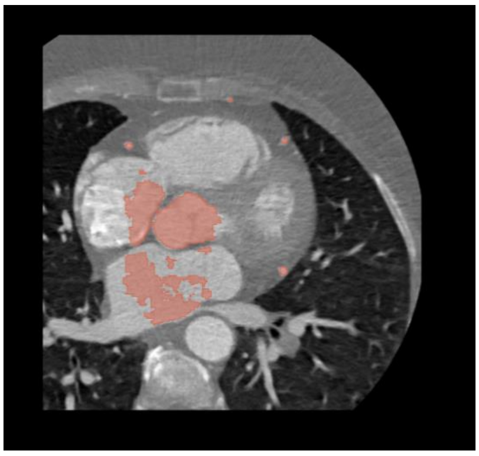


Figure A7. *Segmentation of the right and left atrium viewed in a CCTA image in the axial plane.*

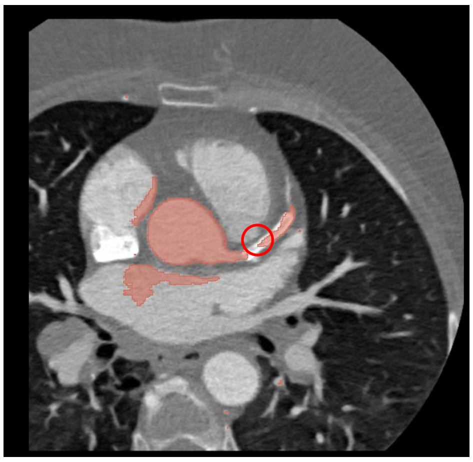


Figure A8. *Non detection of the calcium in a in a CCTA image in the axial plane.*

In the case presented in *Figure A9*, we can see an uncompleted ascending aorta, some segmentation stops in the main coronary branches and some artefacts produced by the UNETR patches. These drawbacks do not happen in the nnU-Net inference. The root of the aorta is well contoured in the UNETR segmentation (shown in *Figure A10).*

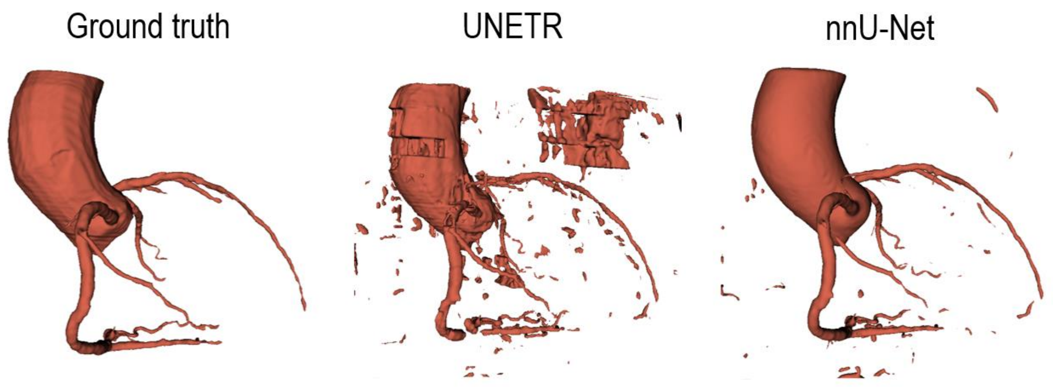


Figure A9. Comparison between the ground truth, the UNETR and the nnU-Net inference.

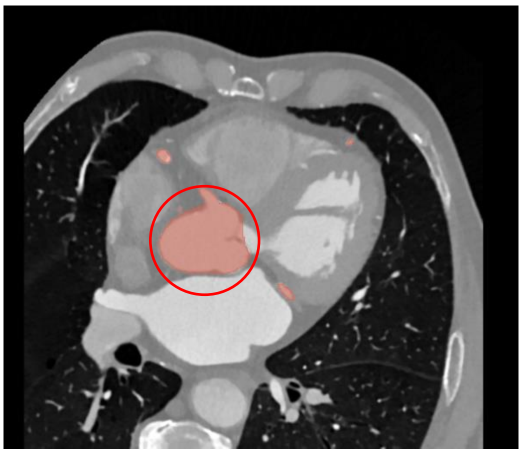


Figure A10. Root of the aorta well contoured highlighted in red in a CCTA image in the sagittal plane.

Figure A11 shows an inference with an uncompleted aorta, a stop in the circumflex artery, a mammary vessel segmented and a big presence of artefacts. Nevertheless, the anterior descendent artery is segmented very precisely. We can see that there is less segmentation of non-coronary arteries, as exception of the mammary vessel, in the UNETR than in the nnU-Net. However, vessels are worse delineated in the UNETR segmentation.

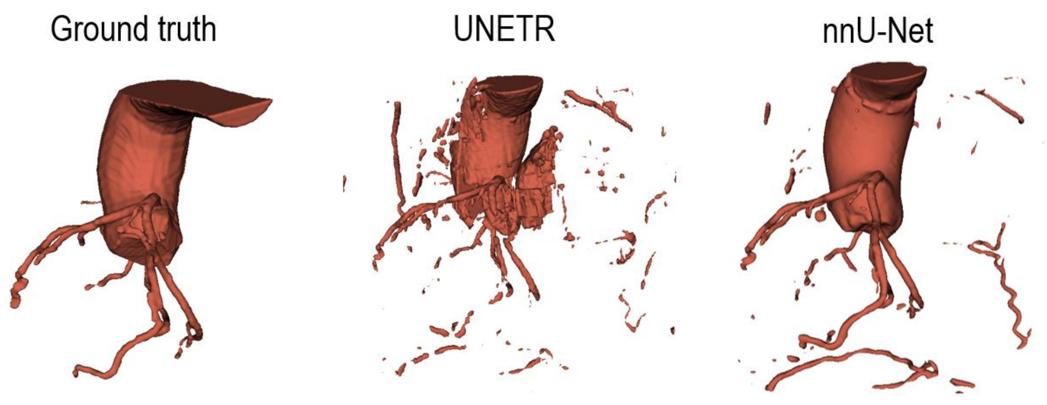


Figure A11. Comparison between the ground truth, the UNETR and the nnU-Net inference.

We can observe a rough UNETR segmentation with some stops in the distal part of the coronary arteries in *Figure A12*. However, the aorta and the main branches of the coronary arteries are well segmented, there is a poor presence of external artifact, the UNETR has correctly segmented the coronary arteries although the CT artefact (shown in *Figure A13)*. and the conal artery is well detected. Despite the inference has a better segmentation of the root of the aorta in the nnU-Net , the UNETR model has a better conal artery detection and less artefacts in this case.

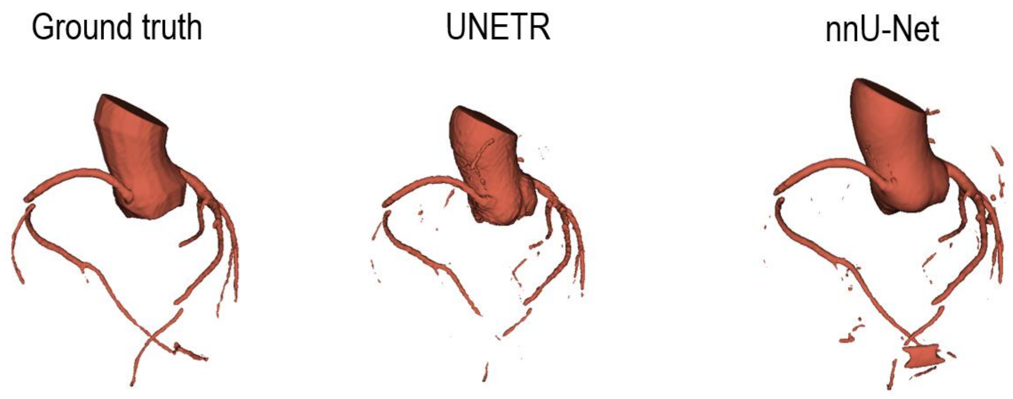


Figure A12. Comparison between the ground truth, the UNETR and the nnU-Net inference.

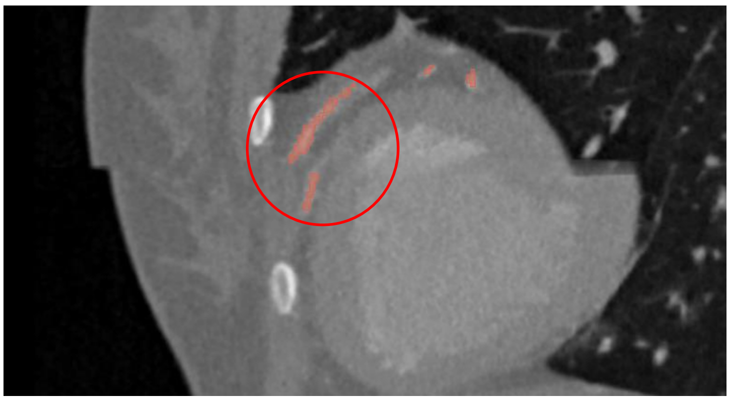


Figure A13. Coronary arteries segmentation in a CCTA image with an artefact in the sagittal plane.