UNIVERSITAT DE BARCELONA

Final Degree Project

Biomedical Engineering Degree

Alternative data representations for a Deep Learning-based segmentation pipeline applied to fetal Doppler echocardiography

Barcelona, 08 de 06 de 2022

Author: Iago Muñoz Rodríguez

Director/s: Bart Bijnens

Guillermo Jiménez Pérez

Tutor: Fàtima Crispi

Acknowledgements

First and foremost, I would like to express my gratitude to the professionals that have guided me, helped me and supported me through the development of this project. I wish to express my sincere thanks to Dr. Bart Bijens for offering me the opportunity of developing this project under his research group, to Dr. Guillermo Jiménez Pérez for supporting, helping, guiding and teaching during all the process and to Dra. Fàtima Crispi for supporting and solving my doubts.

Abstract

Doppler echocardiography is a crucial image acquisition technique in fetal medicine that generates spectrums of blood velocities. The current pipeline for its segmentation is very reliant on manual quantification steps, resulting labour-intensive and time expensive.

Given the rise of Deep Learning in the medical image segmentation field, some initial Deep Learning based models have been trained and tested for its automatic segmentation. A project in the scope of a grant awarded by the Bill and Melinda Gates Foundation's Global Health program, has obtained some initial good results. Their baseline solution proposed uses a W-net with 6 levels and a binary mask as data representation with values of 1 from the reference line to the curve position. However, these results could be improved.

The aim of this project is to design Deep Learning based models using alternative data representations in order to find an alternative solution that overperforms the baseline solution. The dataset used contains 7063 fetal Doppler echocardiographic images which are split into training, validation and test sets. The model architectures used are U-net and W-net architectures with different levels, from 5 to 7. The data representations proposed are a binary mask around the curve position using different width values, and a linear regression. 24 models are trained combining all the architectures with the several data representations, using Dice loss for binary mask data representation models and mean square error (MSE) loss for models using linear regression. For the performance evaluation, different metrics are used when models predict unseen data from the test set.

The results show that the baseline solution overperforms the alternative solutions tested in this project. It is observed that more complex and deep architectures with a data representation based on binary masks that generate big shapes work better for these images. Further alternative solutions can be studied in order to develop a much powerful segmentation tool.

Table of contents

1. Introduction

Echocardiography is an image acquisition technique highly used in medicine as a diagnostic tool. It is based on the generation of ultrasound waves by a piezoelectric transducer. While applying short electrical pulses the piezoelectric transducer compresses generating these waves. The waves are reflected at layers between different tissues where there are acoustic impedance changes. The returning ultrasound waves arrive at the transducer triggering vibrations that are transformed into electric signals and then into a digital image. The main advantages of this technique are the quality and the security that offers, due to not being invasive and not irradiating the tissue. Thus, it is widely used and considered to be essential in fetal medicine.

Several modalities of this technique exist: M-mode, 2D and 3D. M-mode is defined as time motion display of the ultrasound wave along a chosen ultrasound line and it provides a monodimensional view of the heart. 2D echocardiography displays a cross-sectional "slice" of the beating heart, including the chambers, valves and the major blood vessels that exit from the left and right ventricle. 3D echocardiography displays a volume of the beating heart.

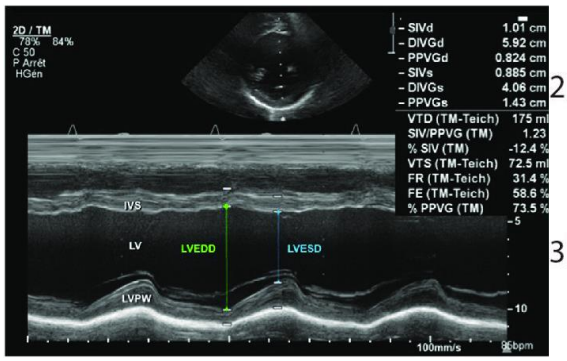


Figure 1: M-mode echocardiography [40]

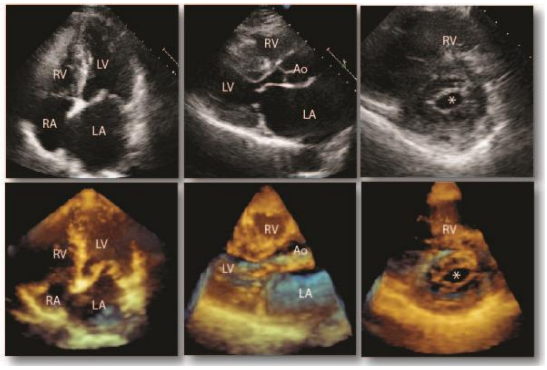


Figure 2: 2D and 3D echocardiography [41]

Doppler echocardiography allows to obtain velocity profiles. These profiles can be obtained from tissue velocity or blood velocity depending on the frequencies used. The spectrum obtained from this imaging modality presents valuable information for the assessment of various cardiac pathologies. In the case of fetuses with compromised cardiovascular systems it has been demonstrated to be a vital diagnostic and monitoring tool through the usage of various modalities.

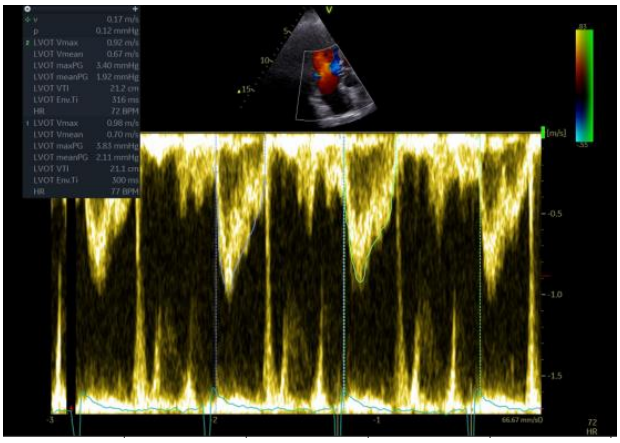


Figure 3: Doppler echocardiography with velocity profile.

The pipeline for analysing fetal images is, however, very reliant on manual quantification steps and labour-intensive, pivoting towards the markup of a series of control points on the image and the definition of the curve envelope for their posterior analysis.

To tackle this problem, machine learning (ML) algorithms for image automatic quantification must be considered. These algorithms have proved to be powerful for several problems that rely on the use of structured data. However, ML requires complex feature engineering manually performed. This process, apart from being costly, also reduces generalization capacity. Models would only be able to analyse images obtained from the same orientation of the fetal heart (aortic, mitral valve, etc.) [1].

Furthermore, analysing the complexity of the problem and the large amount of images available, deep learning (DL) could be a much more suitable option [2].

Given the rise and consolidation of DL based solutions on many industrial settings, the application of DL on fetal echocardiographic data is starting to be explored. This artificial intelligence (AI) branch has proven to be remarkably powerful regarding segmentation problems with large datasets. However, DL-based models are oftentimes regarded as black-box models that provide an output while hiding the decision process. This can have detrimental side effects such as bias in the decision process or due to processing data outside of the inherent variability of the training data pool.

The development of an automatic segmentation tool for this type of medical images can bring substantial benefits. To begin with, the acquisition time and the working time of the professionals would be reduced leading to a larger number of tests being performed in the same period of time. Furthermore, human errors could be reduced as the tool performs the technical part of the diagnostic which allows to improve accuracy and reproducibility. Finally, this automatic tool would allow to obtain and analyse a bigger amount of information from the image. Currently, maximum velocity and the pulsatility index are the parameters measured, although other values such as shape or acceleration could contain important information that nowadays is not being considered.

Based on the previous benefits observed, some initial DL models have been trained and tested in a project involving the processing and analysis of fetal flow waveforms to predict adverse perinatal outcomes with machine learning, in the scope of a grant awarded by the Bill and Melinda Gates Foundation's Global Health program, through Aga Khan University. The goal is to obtain a model with optimal performance to integrate it in a platform of fetal analisis [3].

This project was devised as an opportunity to research alternative solutions to the existing pipeline in use, in collaboration with the project awarded by the Bill and Melinda Gates Foudation. The current solution, which will be detailed in section *4.1.1.3* presents a correct performance, but alternative data representations could have been employed. This project is based on implementing other approaches to this problem with the aim of addressing their potential as better and more optimal performing solutions.

1.1 Objective

The initial hypothesis of this project is that exist different DL based algorithms for the segmentation of fetal Doppler echocardiographic images that improve the performance from the one already implemented.

The main objective of this project is to develop several DL based algorithm that automatically delineates fetal Doppler echocardiography images for and evaluate their performance.

In order to accomplish the first goal, secondary objectives have been stablished:

- Design, train and evaluate the performance of different deep learning architectures.
- Find an effective data representation.

1.2 Description of the situation

This project is developed as the final Biomedical Engineering degree project during its final year. It can be considered as a part of a research project by the *Translational Computation in Cardiology* group from Clinic-IDIBAPS. Concretely, it branches out form a project involving the processing and analysis of fetal flow waveforms to predict adverse perinatal outcomes with machine learning, in the scope of a grant awarded by the Bill and Melinda Gates Foundation's Global Health program, through Aga Khan University.

As it is completely computational, the development of the project and the meetings are done online. This kind of workflow requires access to a computational environment with graphic processing units (GPU) such as the high performance computing (HPC) cluster from the University Pompeu Fabra (UPF) via VPN.

Initially, this project was meant to be performed during the first semester, however the late start on the computational work made the project be postponed for the second semester. This was due to problems having access to the high performance computing environment which the University of Barcelona (UB) did not allow to access.

Furthermore, with the aim of upgrading the initial knowledge around deep learning, some online courses were taken before starting the project: *Introduction to Deep Learning (MIT 6.S191)* and *Intro to Deep Learning with Pytorch (Udacity).*

From a personal academic point of view, the main objective of this project is to get familiarized with artificial intelligence implementations and research. This would involve all the process from learning to use programming libraries such as PyTorch and data pre-processing metrics, to understanding all the steps from the design to the implementation of an artificial intelligence model. Furthermore, interacting with professionals share a common research goal is something really motivating.

1.3 Scope and span

The scope of this project is involved in different fields such as artificial intelligence (AI) algorithm development and diagnostic of fetal cardiac diseases.

Doppler echocardiographic images contain velocity profiles that have valuable clinical information. This information is obtained by the delineation of this spectrum and obtaining a unidimensional

curve (*Figure 4*). Currently, this delineation is performed manually by professionals which is a tedious task. The development of this DL based algorithm would automatise the analysis of the spectrum and this would release the professionals from the manual segmentation task, leading to less human errors, and also allowing to obtain much more information and data that currently is not being obtained.

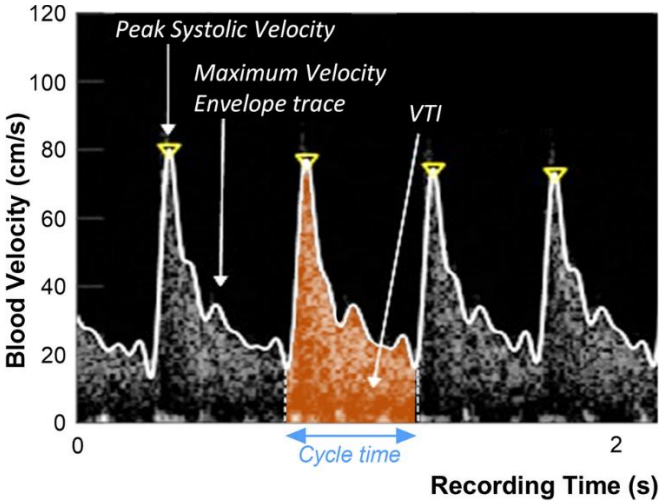


Figure 4: Delineation of a doppler spectrum [4].

Spectral Doppler from echocardiography provides information about the velocity of blood flow over time. Therefore, it allows the measurement of maximal (highest point of the curve) and mean velocity. Doppler curves also provide valuable information about the timing of events, especially when relating the Doppler curves to the ECG tracing. For instance, it is possible to measure time intervals such as the ejection period, isovolumetric relaxation time, or diastolic filling time. Also, by viewing the slope of the curves (i.e. deceleration and acceleration times), velocity changes can be determined [5].

In order to develop this algorithm, the scope of the project encompasses: diving into the DL world and its applications related to echocardiographic images for fetal medicine, design of convenient data representation method in order to support better training, design of several DL architectures to be trained to achieve the purposed goal, evaluate the effectiveness of the models trained.

2. Background

2.1 Echocardiography

Echocardiography is a non-invasive test that uses sound waves to create moving pictures of the heart. Nowadays, it is a basic technique in fetal medicine to detect cardiovascular pathologies. Fetal echocardiography is used to diagnose cardiac conditions in the fetal stage, which are amongst the most common birth defects. The diagnosis of these conditions in the fetal stage is vital to clinically act as soon as possible and reduce harm in the fetus and mother.

This non-radiant imaging technique shows the size and shape of the heart and show how well your heart's chambers and valves are working. Echo also can pinpoint areas of heart muscle that aren't contracting well because of poor blood flow or injury from a previous heart attack. It can detect possible blood clots inside the heart, fluid build-up in the pericardium (the sac around the heart), and problems with the aorta. The aorta is the main artery that carries oxygen-rich blood from your heart to your body [6].

The process to obtain ultrasound images is the following. A sound wave is produced by a piezoelectric transducer. Piezoelectric materials are compressed with a given electrical charge and return to their original shape when the charge is removed. Therefore, strong and short electrical pulses are applied that create the ultrasound wave. Materials at the end of the transducer and a watery gel that is placed between the patient's skin and the probe permit an efficient transmission of the ultrasound, as air causes total reflection of ultrasound. The ultrasound is reflected at layers between different tissues, specifically where there are acoustic impedance changes in the body. The piezoelectric effect of the transducer also works in reverse and the return of the ultrasound provokes its vibration, which is converted into electrical pulses which are processed and transformed into a digital image. Depending on the time it took for the echo to be received and how strong this echo was, the corresponding pixel is illuminated with the corresponding intensity in the image.

Compared to other imaging techniques such as MRI, echo provides a much better temporal resolution which allows to detect the pumping of the heart as the acquisition is immediate. This, however, comes at the expense of the spatial resolution, which is reduced. Nevertheless, for cardiac hemodynamics it is crucial to have high temporal resolution and this is the reason of the importance of this technique.

Doppler ultrasonography measures blood flow velocities based on the Doppler effect, which is the changes in sound frequency as a sound source moves from the observer point of view. When an ultrasound beam is directed toward moving targets (red blood cells), the transducer determines the frequency shift which that is related to the velocity of the moving target, transmitted frequency and the angle between the direction of the ultrasound beam and the direction of the moving target. It is used for fetal hemodynamic and cardiac function and to study the movement of the baby and the blood inside him. The spectral Doppler shows blood flow velocities plotted against time. The information that can be obtained from a spectral Doppler trace includes flow velocity, direction of flow, the timing of the signal with cardiac events and intensity of the flow signal [7].

2.2 Artificial intelligence and deep learning

Artificial intelligence (AI) is a discipline within the computer science field that aims to simulate human intelligence in machines. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable [8].

Machine learning (ML) is a subset of AI that uses algorithms to parse data, learn from it, and then make a determination or prediction about something in the world. So rather than hand-coding software routines with a specific set of instructions to accomplish a particular task, the machine is "trained" using large amounts of data and algorithms that give it the ability to learn how to perform the task [9].

In order to build a ML model, there are 7 main steps to follow [10]:

- **Data collection:** determine type of data to be collected and labels (for supervised training).
- **Data preparation**: it consists on applying different steps in order to be able to train the model with this data. Usually, the data must be normalized, cleaned from errors or duplicates, and also visualized in order to recognize and minimize potential biases in the dataset. Finally, the data should be split in training and validation sets.
- **Choose a model**: select the algorithm that fits better for the required task.
- **Training**: the goal here is that the algorithm learns the required task as often as possible. In this process the data is shown to the model several times (training steps) and the algorithm progressively tunes the parameters to obtain a more accurate output. This step is usually performed by introducing the data from the training set.
- **Evaluation**: once the model is trained, unseen data from the validation set is introduced to the model in order to check its performance.
- **Parameter tuning**: the goal here is to modify model parameters seeking a performance improvement.
- **Make predictions**: using further datasets, a better approximation of how the model will perform in the real world is obtained.

Deep Learning (DL) is a subfield of ML based on algorithms inspired by the structure and function of the brain called neural networks. These neural networks (NN) attempt to simulate the behaviour of the human brain allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help approximate more complex input-output mappings of data.

DL has evolved hand-in-hand with the digital era, which has brought about huge amounts of data that traditional algorithms cannot take full advantage of. In addition, DL has benefitted from advanced computer hardware, such as graphical processing units (GPUs) and tensor processing units (TPUs), making it possible to train large NNs.

The difference with ML, is that these algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts [11].

2.2.1 Artificial neural networks

ANNs are computational architectures inspired by the way biological nervous systems, such as the brain, process information [12]. It consists of multiple layers of simple processing elements called as neurons which perform two functions: the collection of inputs and the generation of outputs.

In ANNs each node performs some simple non-linear computations and each connection conveys a signal from one node to another, labelled by a number called the "weight". A node or neuron only depends on the information locally available, either stored internally or arriving via the weighted connections. A network formed by a single processing unit is called a perceptron and it is composed by three main parts.

First, the inputs and the bias term that arrive through the weighted connections. Secondly, a weighted sum of the inputs is performed inside the node, which is known as linear combination. Finally, an activation function is applied which maps the output inside known range. Commonly, the function is non-linear which helps the model to generalize or adapt with variety of data and to differentiate between the output.

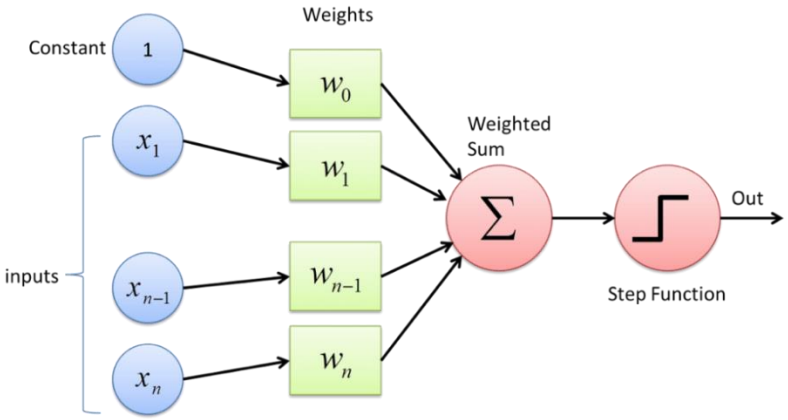


Figure 5: Perceptron structure.

The activation function controls the amplitude of the neuron's output. Thus, the output of the neuron is a function of the linear combination of the inputs with the weighted connections.

$$y = f(z) = f\left(\sum_i x_i w_i\right) = f(W^T x + b)$$

Some of the most used activation functions are [13]:

- **Step function**: allows to only activate the neuron in a situation where the input value is higher than a given threshold value or leave it deactivated when the condition is not met.

$$f(x) = \begin{cases} 0, & x < threshold \\ 1, & x > threshold \end{cases}$$

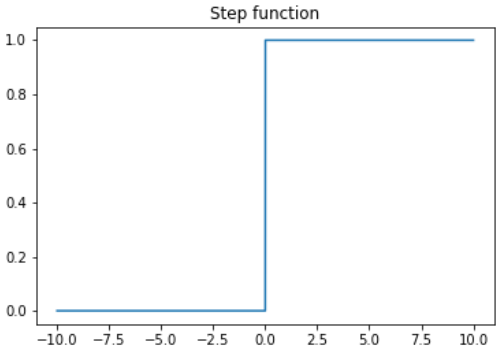


Figure 6: Step function

- **Sigmoid function**: this function also presents a bounded range of activation.

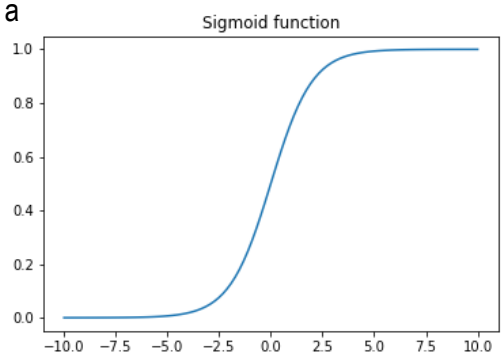


Figure 7: Sigmoid function

- **Hyperbolic tangent function**: presents a bounded range of activation but also considering negative values. Hence, negative inputs of the hyperbolic functions will be mapped to a negative output as well as the input values that are nearing zero will also be mapped to output values nearing zero.

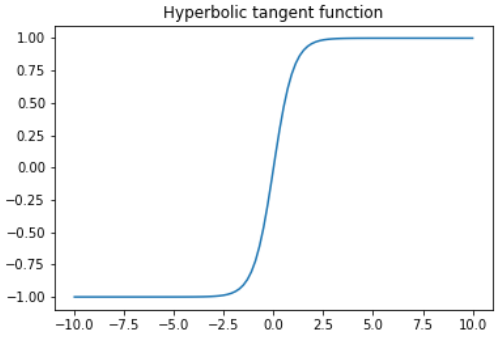


Figure 8: Hyperbolic tangent function

- **Rectified Linear Unit function (ReLU)**: This type of activation function is responsible for transforming the weighted input that is summed up from the node to the strict output or proportional sum. These functions are piecewise linear functions that usually output the positive input directly; otherwise, the output is zero.

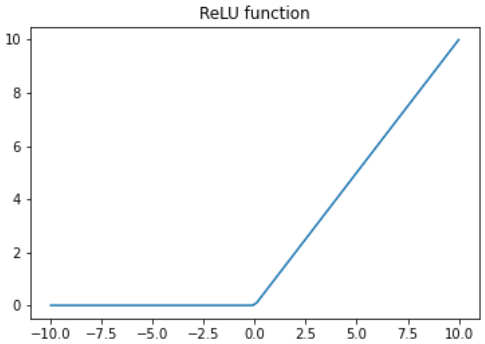


Figure 9: ReLU function

By itself, a single processing element is not very powerful as it only generates a scalar output. The power of the system emerges from the combinations of many units in an appropriate way and the non-linearity. Following this idea, ANNs architectures have several node layers where the initial is known as input layer and the last one as output layer. These structures follow a feed forward pipeline (Fig. 2) where a node can only connect forward to nodes from the next layer. Deeper node layers allow higher level feature extraction.

2.2.2 Image segmentation

Image segmentation aims to extract regions of interest from an image. The goal in the medical field, would be to extract or detect anatomical or pathological structures. It often plays a key role in computer aided diagnosis and smart medicine due to the great improvement in diagnostic efficiency and accuracy. Within the image segmentation field, one can identify the following types: semantic

segmentation and instance segmentation. The first type is a pixel- level classification that assigns a corresponding category to each pixel in an image. The second type not only needs to achieve pixel- level classification, but also needs to distinguish instances on the basis of specific categories.

Popular medical image segmentation tasks include liver and liver-tumour segmentation, brain and brain-tumour segmentation, optic disc segmentation, cell segmentation, lung segmentation and pulmonary nodules, etc [14].

Several families of ANNs architectures that can be applied to cardiac image segmentation problems [15].

2.2.2.1 Convolutional Neural Networks (CNNs)

A standard CNN consists of an input layer, an output layer and a stack of functional layers in between that transform an input into an output in a specific form. These functional layers often contain convolutional layers, pooling layers and/or fully connected layers. In general, a convolutional layer contains convolution filters, which is followed by a normalization layer and a non-linear activation function to extract feature maps from the input. These feature maps are then downsampled by pooling layers, which remove redundant features to improve the statistical efficiency and model generalization. After that, fully connected layers are applied to reduce the dimension of features from its previous layer and find the most task-relevant features for inference.

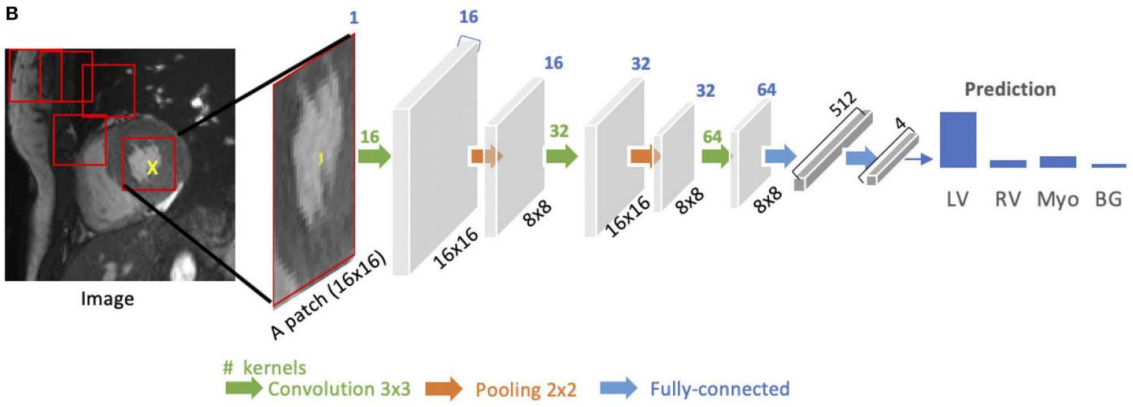


Figure 10: Patch-based segmentation method based on a CNN classifier [15].

2.2.2.2 Fully Convolutional Neural Networks (FCNs)

FCNs are a special type of CNNs that are designed to have an encoder-decoder structure such that the input and the output have the same size and only use convolutional operations. The encoder transforms the input image into a high-level feature representation whereas the decoder interprets the feature maps and recovers spatial details back to the image space by upsampling and convolution operations.

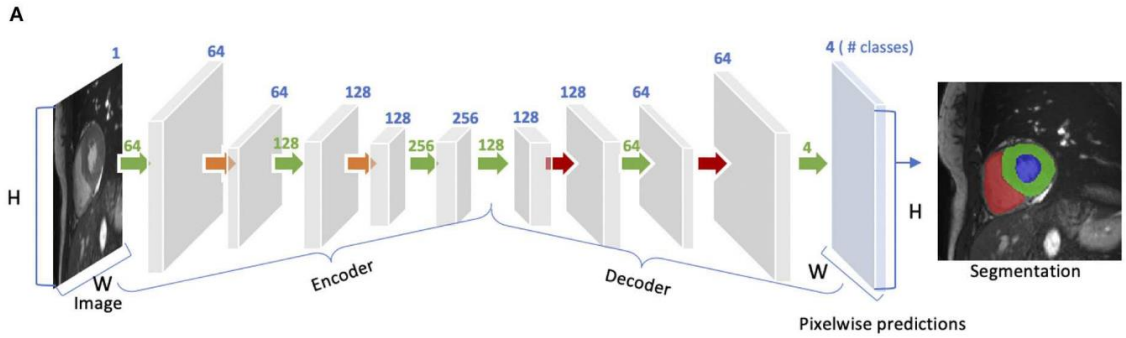


Figure 11: Architecture of an FCN.

Compared to patch-based method for segmentation using CNNs, FCNs are trained and applied to the entire image removing the need for patch selection. However, these architectures may be limited to capture detailed context information as low level features can be eliminated by the pooling layers in the encoder.

In [16] the U-net architecture is proposed as a solution for this problem. The skip connections that appear in this architecture concatenate channels from the encoder to the decoder at the same level so low-level features are not removed and spatial context is not lost.

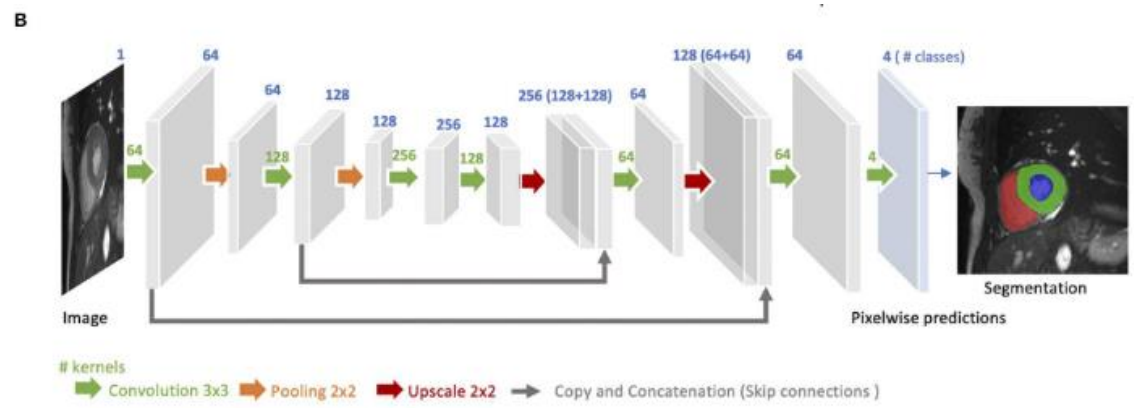


Figure 12: Architecture of a U-net.

2.2.2.3 Generative Adversarial Networks (GANs)

GANs are a type of generative models that learn to model the data distribution of real data and thus are able to create new image examples. These architectures consist of two networks: a generator and a discriminator. During training, the two networks compete against each other. The generator produces fake images aiming to fool the discriminator, and the last one tries to identify the real images from the fake ones. In segmentation, the discriminator would distinguish the segmentation maps from the ground truth maps encouraging the segmentation network (generator) to produce more accurate outputs.

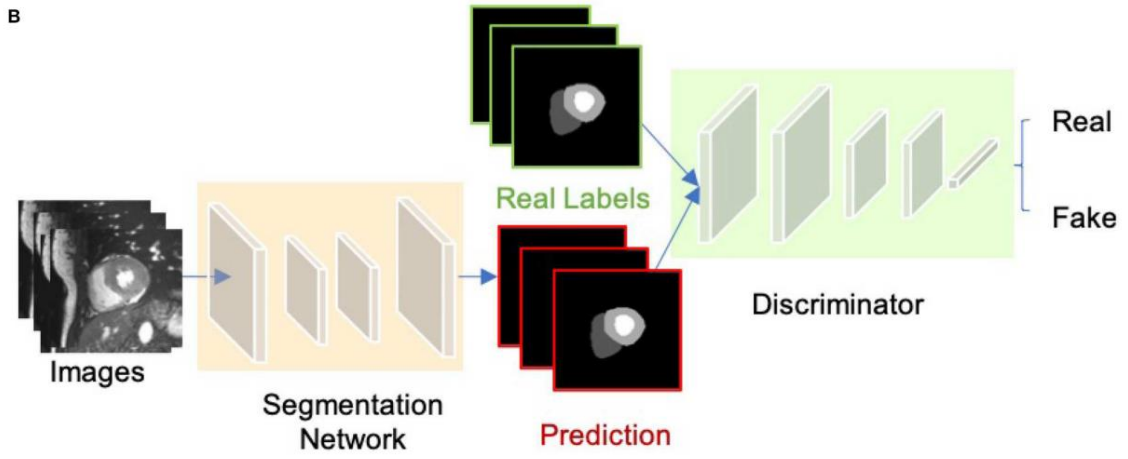


Figure 13: GAN for segmentation applications.

2.2.3 Training ANNs

1. **Input data**: the training dataset previously processed is introduced into the model.
2. **Feedforward**: this step generates an output with the model. Here the data is going in the forward direction applying the activation function after each node and the regularization method chosen. After each iteration, this output will be more accurate and the error will decrease as the weights update.
3. **Loss function**: computes the error between the output obtained and the expected result.
4. **Backpropagation**: activates the optimizer and updates the weights in order to decrease the error (goes in the backward direction).

The training process is iterative which means that the steps will be repeated until convergence is reached, or training is stopped.

2.2.3.1 Regularization

The objective of a neural network is to have a final model that performs well both on the data used to train it and on the new data on which will be used to make predictions. Train a model that generalizes to unseen data it is a complex task. Underfitting corresponds to a situation where the model has not been trained enough, whereas overfitting corresponds to a situation where the model has lost the ability to generalize, and it performs well on the training set but poorly on new data [17].

Regularization attempts to handle overfitting problems using different methods:

- **Weight regularization**: penalises the model during training based on the magnitude of the weights. This will encourage the model to map the inputs to the outputs of the training dataset in such way that the weights of the model are kept small.
- **Activity regularization**: penalises the model during training based on the magnitude of the activations.
- **Weight constraint**: constrains the magnitudes of the weights to be within a range or below a limit.
- **Dropout**: based on probability deactivates nodes during training.

- **Noise**: adds statistical noise to inputs during training.
- **Early stopping**: monitors model performance on a validation set and stops training when performance decreases.
- **Data augmentation**: it is a set of techniques to artificially increase the amount of data generating new items from existing data.

2.2.3.2 Loss function

The loss function addresses directly to the main goal of the network, which is to generate an output that solves the required task it is being trained for. This function corresponds to the mathematical representation of objectives and has an impact in accuracy and velocity for the learning process. It is a method of evaluating the error of the algorithm when it models the dataset. The value of the loss function indicates whether if the model is closer to an optimal performance or not.

In order to choose a loss function for a model, it is connected by design to the activation function used in the output layer of the neural network.

- **Mean squared error (MSE)**: for regression problems with linear activation functions.
- **Cross entropy**: for binary classification problems with sigmoid activation functions or multiclass classification problems with softmax activation functions.
- **Dice score**: for image segmentation problems as a region-based loss function.

2.2.3.3 Optimization

During training, the network updates the weights in response to the errors the made on the training dataset. Updates are made to continually reduce this error until either a good enough model is found, or the learning process gets stuck and stops.

The goal of an optimization algorithm is minimizing the loss function. This function will appear to be a non-convex surface with many "valleys" and so finding the global minima turns to be a challenging problem.

- **Gradient descent (GD)**: this algorithm will take the loss function of all the initialized weights and compute the gradient. This gradient will point to the steepest part of the loss function, which is the path to follow in order to minimize the loss function.

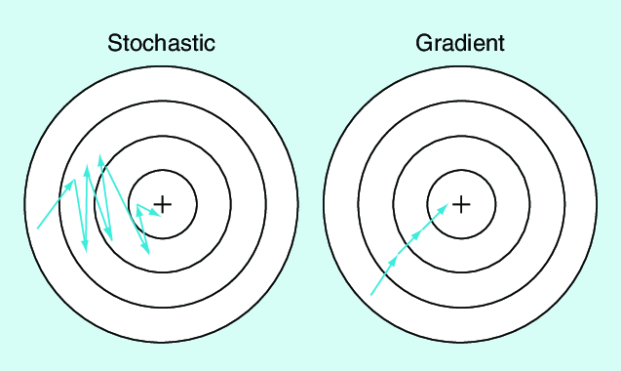


Figure 14: SGD scheme [42].

- **Stochastic gradient descent (SGD)**: this algorithm is a drastic simplification of GD. It appears to be more efficient as it estimates the gradient from the loss function of a randomly picked example on each iteration. This gradient will not be as accurate as in GD and more iterations will be taken, however this method is much faster as it saves computational costs with the loss functions calculations. [18]

- **Adam**: this algorithm is for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. It combines the advantages of AdaGrad, which works well with sparse gradients, and RMSProp, which works well in on-line and non-stationary settings. This method has proven to be effective as it takes big steps when far from the minima and smaller precise steps when closer to the minima of the loss function [19].

2.2.3.4 Backpropagation

The backpropagation (BP) algorithm is commonly used for training ANN. Training is usually done by iterative updating of weights employing the loss function. This error signal is then backpropagated to the lower layers. Traditionally, two parameters, called learning rate (LR) and momentum factor (MF), are used for controlling the weight adjustment along the steepest descent direction and for dampening oscillations [20].

In order to control the training procedure, the learning rate (α) plays a key role. This hyperparameter controls how much the weights (w) of the network are adjusted in each iteration with respect the loss gradient ($\nabla L(w)$). This value will be between 0 and 1:

$$w_{t+1} = w_t - \alpha * \nabla L(w)$$

On the one hand, a high value could overshoot the minimum and fail to converge or even diverge, on the other hand a low value could take a long time to converge and even get stuck on a plateau region [21].

In case of Adam optimizer, the learning rates are managed on a per-weight basis.

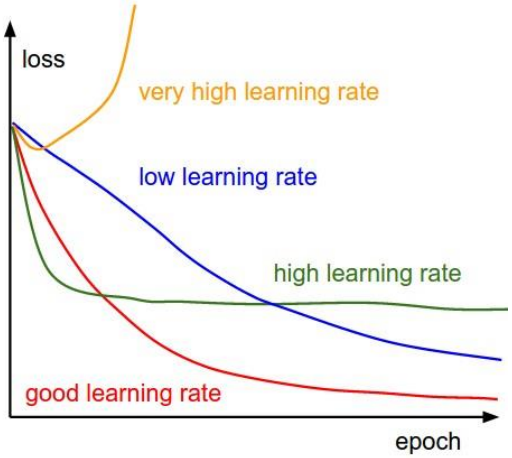


Figure 15: Effect of various learning rates on convergence.

3. Market analysis

3.1 Addressed sector

As stated before, the delineation algorithm for Doppler echocardiographic images is mainly addressed to cardiac fetal medicine as an improvement for Doppler ultrasonography. Congenital heart disease is the most frequently occurring congenital disorder in new-borns and is the most frequent cause of infant death from birth defects [22]. These diseases affect nearly 1% of the births per year in the United States and the prevalence is increasing [23]. The increase in the diagnosis of heart defects has probably been influenced by improvements in diagnostic methods, such as the widespread use of echocardiography.

3.2 Market evolution

The market evolution of Doppler Ultrasound Systems is expected to gain market growth in the forecast period of 2021 to 2028. The growing acceptance of handheld devices will help in escalating the growth of the Doppler ultrasound systems market. The major factors that are expected to boost the growth of these systems market are the growing indices of cardiovascular diseases and the advances in the ultrasound diagnostic technologies [24].

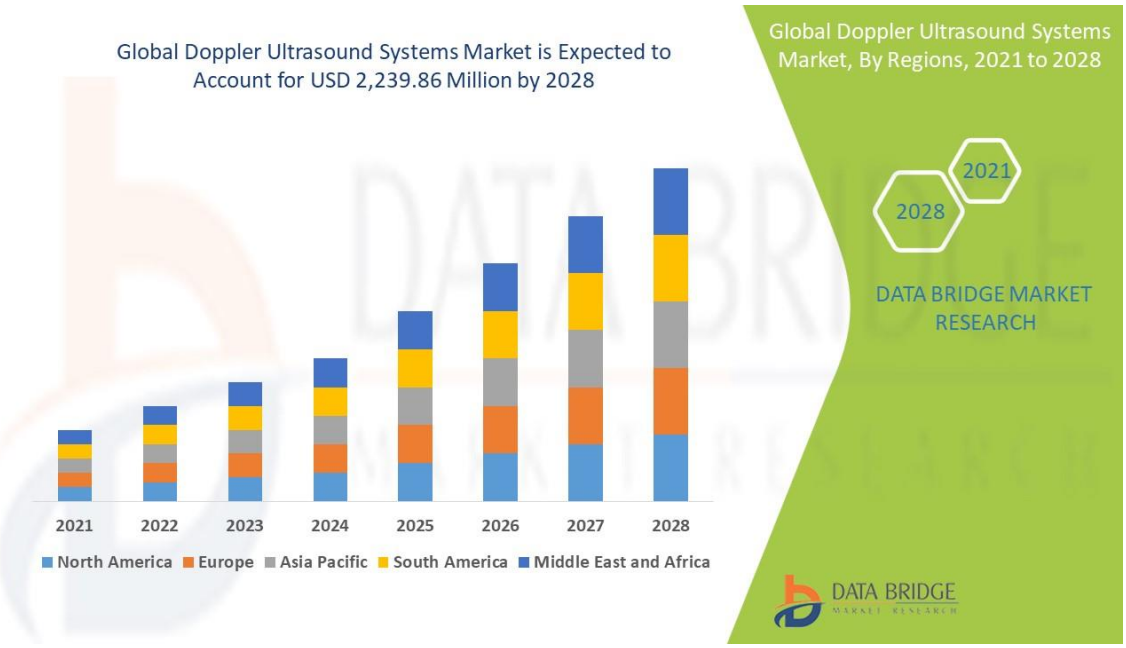


Figure 16: Market evolution [24].

3.3 Competition

The potential competitors that this product can face in the marketplace are AI algorithms that offer an automatization method for delineation on Doppler echocardiographic images. These competitors are mostly on research phase. For this reason, a PubMed search has been performed with the key words “doppler echocardiography automatic segmentation” and articles older than 2017 were not considered. In [25] the adoption of a supervised classifier trained with the samples representing the upper and lower velocity envelopes obtained from the Doppler spectrum achieved significant results. In [26] it is reviewed the automatic methods for analysing echocardiography data. On the Doppler spectral envelopes segmentation section, a conventional ML model presents promising results although it is only trained for mitral valve images and using only 25 patients. In

[27], a supervised classifier (a simple ANN) is used to identify measurable fetal heartbeats from the Doppler envelope and the results appear to have some limitations.

3.4 Product environment

The environment that is facing this product is potentially favourable. It can be observed that most of the approaches inside the competition do not apply DL architectures despite its superior performance. This can be explained due to the recent rise of this AI subset. Also, DL needs long training processes, large sets of annotated data and increased processing power. Fortunately, this product can cover these needs which can help to create a gap in this market.

4. Conception engineering

4.1 Materials

The data from which our model will be trained, evaluated and tested is provided by the *Transversal Computation in Cardiology (IDIBAPS)* research group. The dataset is private and previously a confidentiality agreement had to be signed.

Regarding the dataset, it contains a total of 7063 doppler echocardiographic images. These images come from 2 different studies performed in fetal patients. These images where obtained from nine specific views of the cardiac anatomy, which can be observed in *Table 1*. The amount of image variability makes it a strong dataset as it might help promote model generalization and avoid overfitting.

Table 1: Dataset summary

Specific view	Number of images
Aortic isthmus	883
Ductus arteriosus	879
Ductus venosus	808
Right ventricular outflow track	890
Tricuspid valve	739
Left ventricular outflow track	891
Mitral valve	741
Middle cerebral artery	884
Umbilical artery	888

Furthermore, as the training strategy followed is based on a supervised training, all the images must be labelled with its ground truth. In this case, the ground truth corresponds to the manual segmentation of a single cardiac cycle on each image performed previously by an expert cardiologist.

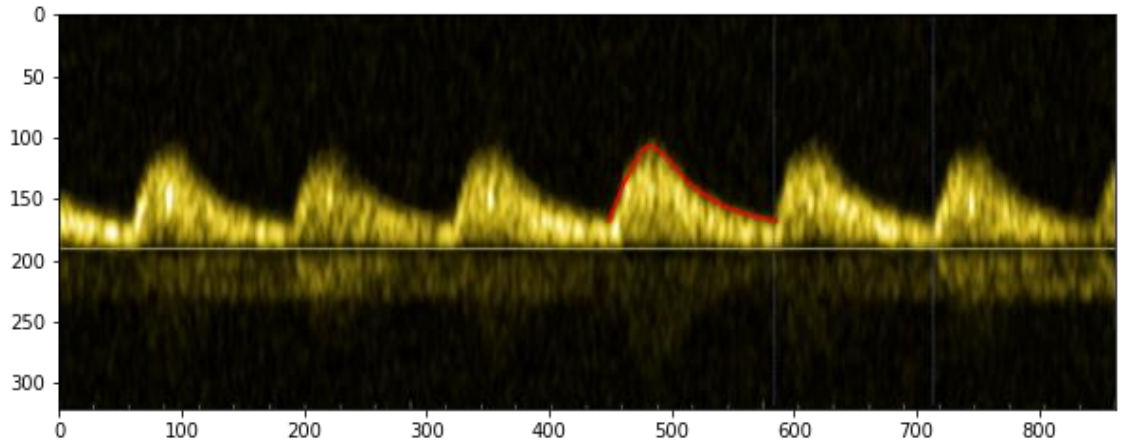


Figure 17: Doppler echocardiographic image with ground truth superposed.

Raw images that come out from a medical imaging device have the DICOM format. Each image anonymized and labelled with a unique identification number. The ground truth values are stored

in two csv file, one named y_coordinates.csv containing the values of the curves regarding the y axis and another one named x_coordinates.csv containing the values of the curves from the x axis.

4.1.1 Data pre-processing

As previously mentioned, images initially are stored with the DICOM format, which is the one generated after the acquisition with the echocardiography. However, the models can not read data in this format, images and ground truths must be in Tensor objects for the PyTorch library. Thus, all the data needs to go through a series of steps to apply a chosen data representation that models can read.

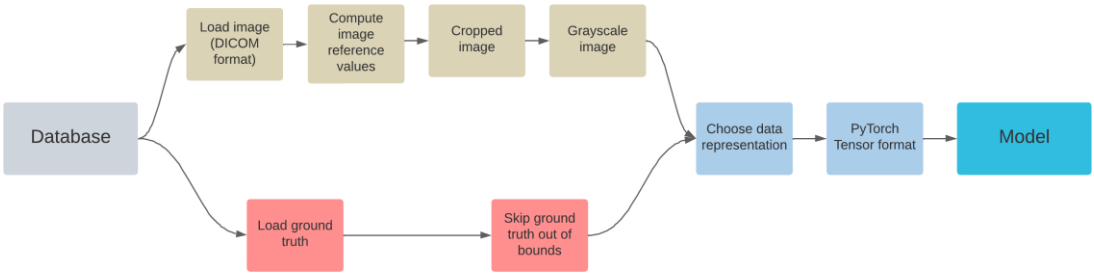


Figure 18: Scheme of the pre-processing steps.

In *Figure 18*, two different paths of data pre-processing are shown. The upper path corresponds to the pre-processing steps of the raw images, whereas the lower path corresponds to the pre-processing of the ground truth curves.

4.1.1.1 Pre-processing of raw images

- **Load images**: according to the initial format (DICOM), images are loaded using the dicom library. The visualization of the images is possible with the matplotlib library using the imshow() method.
- **Compute image reference values**: from the initial image is necessary to implement a function that computes the reference values of the image. These correspond to the coordinates of the doppler region and the coordinates of the reference line of the spectrum. These values are needed for cropping and for data representation of the ground truths.
- **Crop images**: As it can be observed in *Figure 19*, the original images obtained from the acquisition contain the 2D echocardiography, several indicators and the doppler spectrum. Our region of interest is only the doppler spectrum and this is the only part of the image that should be transferred to the models. Consequently, images must me cropped to only contain the doppler region (*Figure 20*). This is done using the coordinates previously obtained that determine the limits of the doppler region.

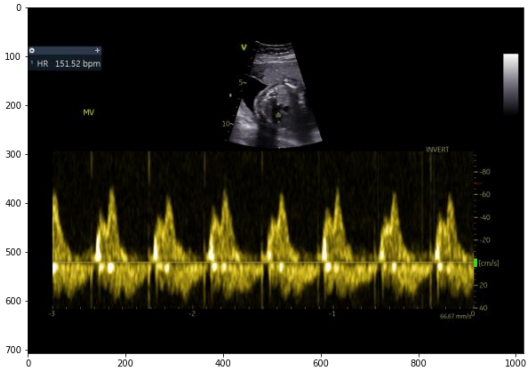


Figure 19: Original image.

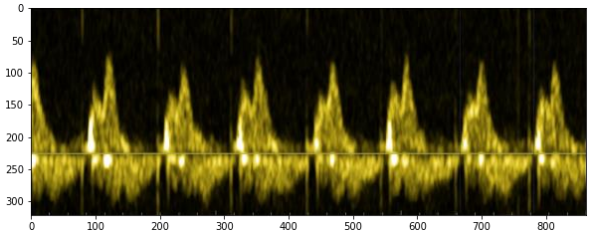


Figure 20: Cropped image.

- **Convert to grayscale**: Convert image to grayscale in order to have only one colour channel as an input for the model. Also, pixel values are converted to float values in range between 0 and 1 as opposed to the uint8 range between 0 and 255.

4.1.1.2 Pre-processing of ground truth

- **Load ground truth curves**: From the dataset the ground truths corresponding to each image are also obtained. As mentioned before these curves have been delineated by an expert cardiologist and correspond to a single cardiac cycle.
- **Skip unwanted ground truths**: Once the curves are loaded, some of those need to be removed in case the curve is out of the doppler region, which corresponds to a bad manual segmentation and might not contain a full cardiac cycle.

4.1.1.3 Choose data representation

Once the images with their corresponding ground truths are pre-processed, a data representation needs to be chosen. The goal of this step is to optimize the performance of the model. The expected output is a full delineation of all the doppler spectrum. Thus, the data received by the model must be an image containing the doppler spectrum and the corresponding label containing a full delineation of the image.

In *Figure 21* it is shown how the data looks once it has gone through the pre-processing steps.

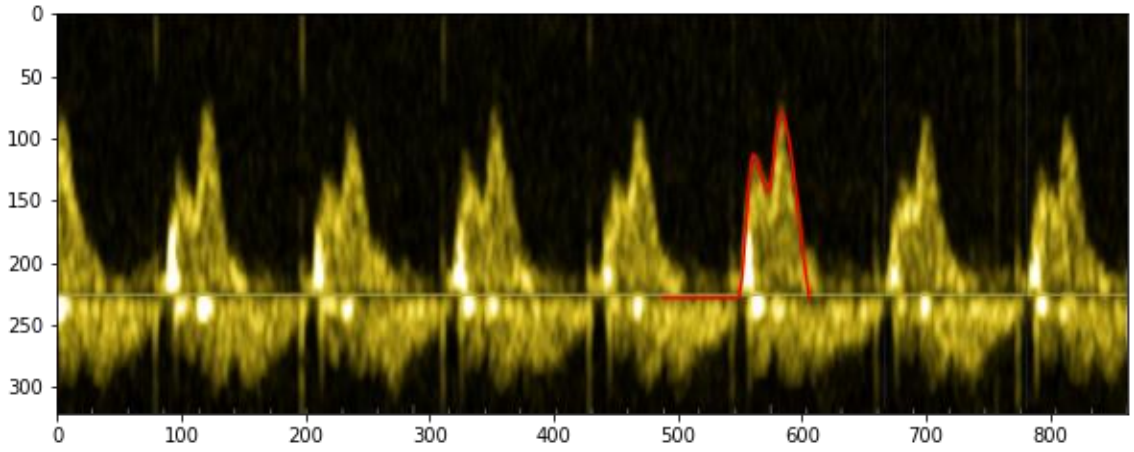


Figure 21: Image with ground truth of a full cardiac cycle.

A clear problem is observed. The ground truth corresponds to the delineation of a single cardiac cycle; however, the model must receive an image labelled with its full delineation.

The research group responsible of this project, proposed a solution for this problem which was already implemented in the first models.

This solution is based on only using the segment of the image that corresponds to the delineation of the ground truth and generate an image by repeating this segment. It could be called tiling strategy.

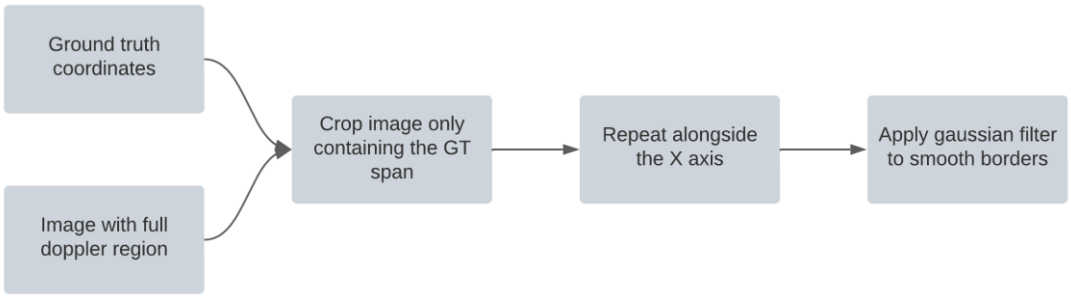


Figure 22: Scheme of the tiling strategy.

The first step of the tiling strategy is to crop the image containing the full doppler spectrum so that it only represents the ground truth span. The current image now only contains a full cardiac cycle that corresponds to the ground truth curve. Then, the image is repeated alongside the x-axis as many times as needed so that the x dimension has a length of 512 pixels. The same repetitions are performed for the ground truth curve. The hard cut was smoothed using a small Gaussian filter in the borders of each cycle.

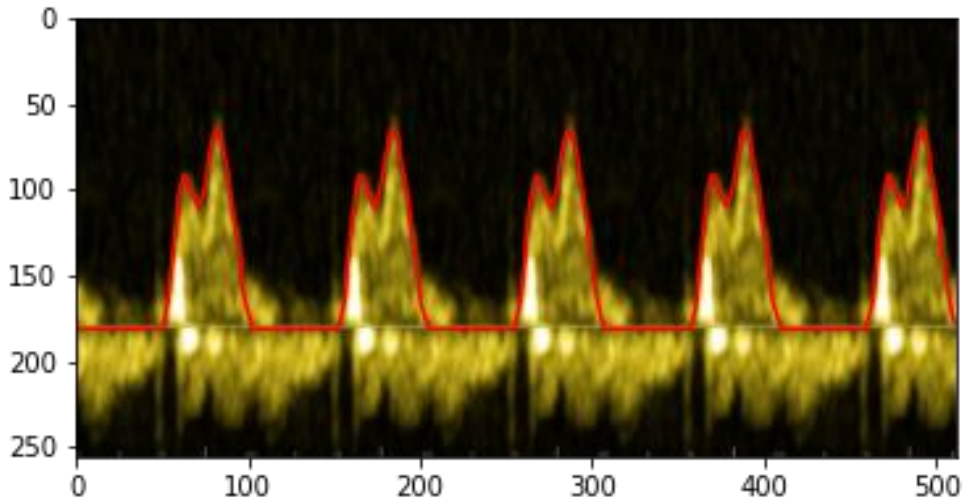


Figure 23: Image generated by the repetition of the ground truth segment.

In *Figure 23*, it is shown the result of applying the tiling strategy. These images correspond to the repetition of the same exact cardiac cycle.

- **Baseline solution, from the reference line to the curve position:**

This is the existing data representation used in the previous models by the research group and it is the way is usually done for segmentation problems (MRI, CT, etc). This is the baseline where this project intends to test other data representations and try to generate better results.

A binary mask is created from the ground truth curve. Values between the reference line and the curve position receive a value of 1, whereas values outside this region receive a value of 0.

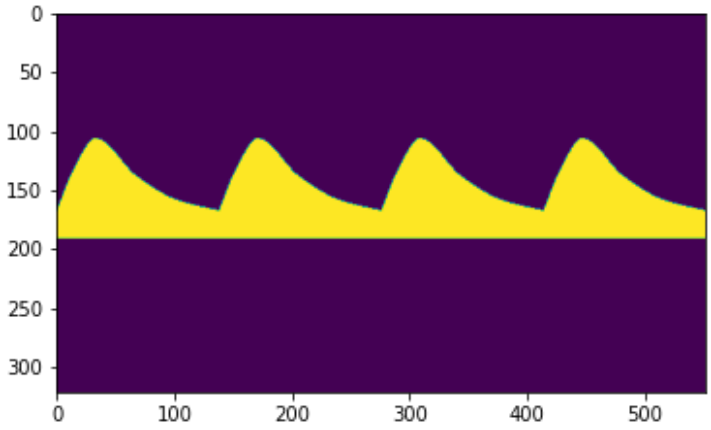


Figure 24: Binary mask from the reference line until the curve.

- **Binary mask around the curve position**:

Based on the existing solution, a different approach using a binary mask is proposed.

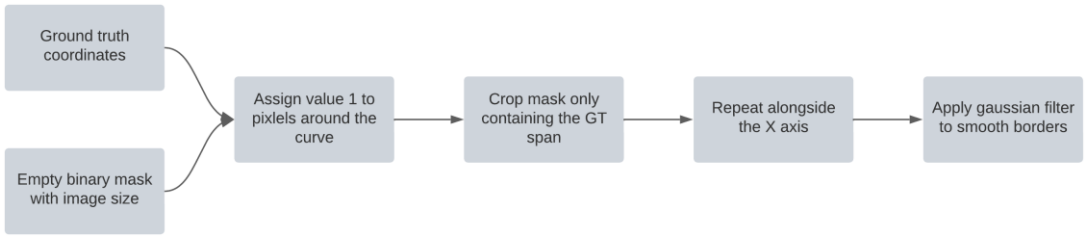


Figure 25: Scheme for the generation of the binary mask.

Initially, an empty mask is initialized with the same size as the doppler image. Using the coordinates of the ground truth, a value of 1 is assigned to those pixels that find themselves close to the position curve. A width value predefined determines which is the width wanted of the curve in the mask.

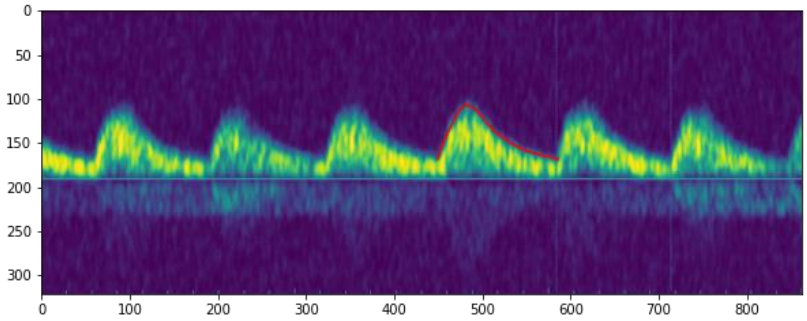


Figure 26: Pre-processed image and ground truth withoud tiling.

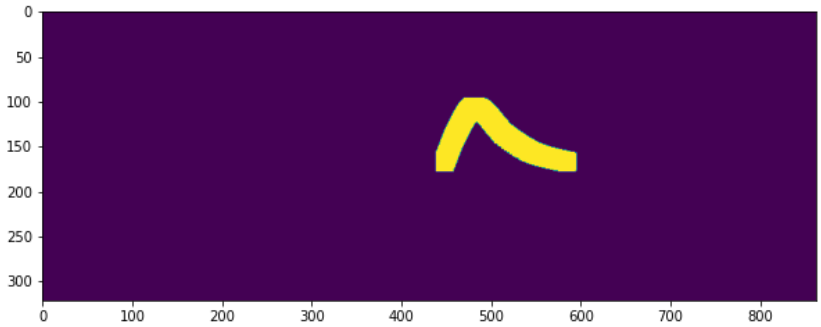


Figure 27: Binary mask with 20 pixel width.

In *Figure 26*, it is observed the initial image with the ground truth. In *Figure 27*, it is shown the binary mask after applying a width value of 20 pixels around the position curve.

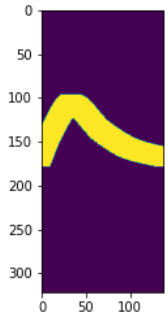
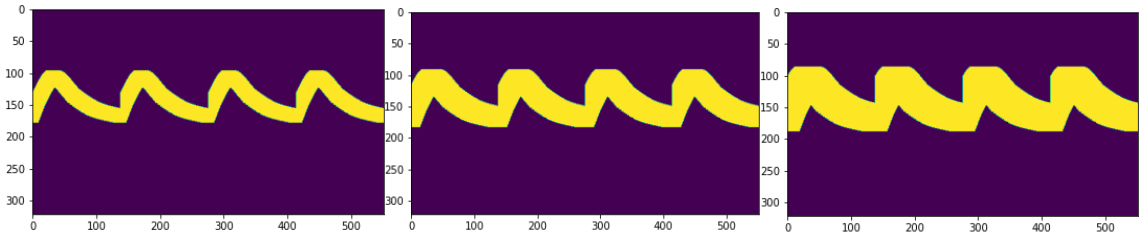


Figure 28: Segment cropped

Next, the mask is cropped so that it only contains the segment with the ground truth, as it is shown in *Figure 28*.



Figures 29, 30 and 31: Binary mask with width values of 20, 30 and 40 pixels respectively

Finally, the segment from *Figure 28* is repeated to obtain the final mask. These final steps are the same performed on the images as shown in *Figure 22*.

In *Figures 29, 30 and 31*, it is shown the final mask with different width values. Each model will be trained using three different width values: 20, 30 and 40 pixels.

- **Linear regression**:

Instead of introducing the ground truth as a modified binary mask, this approach is based on introducing the ground truth as a curve. In order to do this, the model needs to receive a 2D input image and generate a 1D output prediction.

For this case, a similar approach as [28] is implemented. The Soft-argmax function along the width axis allows to convert feature maps directly to joint coordinates, resulting in a fully differentiable framework. To do so a layer is added at the end of the network that operates through the previous 2D output generating a unidimensional vector prediction. While for the other cases the predicted output presents a shape of 256x512, in this case the predicted output will be 1x512 and thus obtaining a 1D vector.

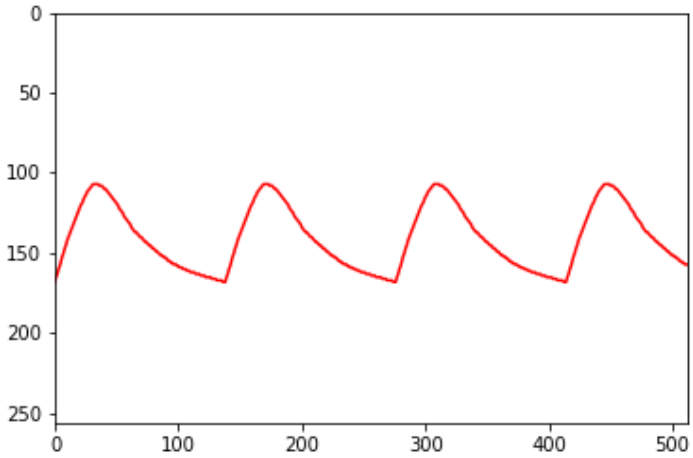


Figure 32: Ground truth tiled.

For this approach, the tiled ground truth curve (*Figure 32*) that is given to the model is obtained following the same last steps as in *Figure 22*.

4.1.2 Dataset splitting

In order to perform a good evaluation of the models, the dataset must be divided in 3 sets.

- **Training set**: the images in this set are used for training the model. This images are introduced to the model while training in order to fit the model.
- **Validation set**: the images in this set are used to provide an unbiased evaluation of the model fitted on the training set while tunning the model hyperparameters.
- **Test set**: the images in this set are used for the performance evaluation of the final model. These images are completely unknown for the model. After training, the model predicts the output for all the test set and the result is compared to the ground truths using different metrics for evaluation.

4.2 Network architectures

In this section, the possible network architectures that can be used for the segmentation of the images will be discussed. Most of them involve FCNs, and particularly U-Nets and W-nets.

4.2.1 U-net

The U-net implemented in this project will be based on the model proposed by [16]. In this paper the model implemented presents good performance using very few annotated images and does not demand a time-expensive training.

The main structure of this network consists of a contracting path or encoder and an expansive path or decoder. The symmetry between both paths yields a u-shaped model. The contracting part follows the typical architecture of an FCN. It consists of the repeated application of two 3x3 convolutions each followed by a rectified linear unit and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step the number of feature channels is doubled. Every step in the expansive path consists of an upsampling of the feature map or a 2x2 up-convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. At the final layer a 1x1 convolution is used to map each component feature vector to the desired number of classes. It is important to select the input image size such that all the 2x2 max-pooling operations are applied to a layer with size values of powers of 2. Also, the initial channel number must be a power of 2 in order to avoid problems with the max-pooling operations [16].

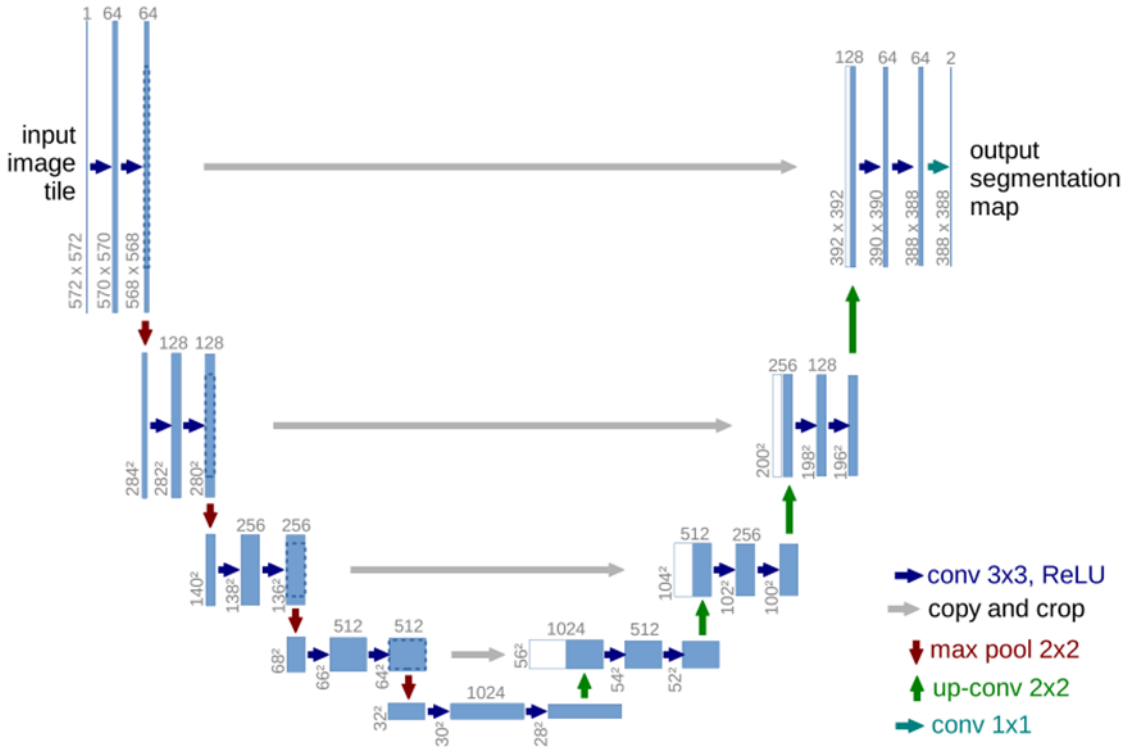


Figure 33: U-net architecture [16].

Medical images present a series of characteristics that are challenging for segmentation problems. Different from common image segmentation, medical images usually contain noise and show blurred boundaries. On the one hand, it is very difficult to detect or recognize objects only depending on image low-level features. On the other hand, it is impossible to obtain accurate boundaries depending only on image high-level features due to lack of image detail information.

The encoder – decoder system of the U-net is a possible solution to this problem. The encoder path performs convolutions such as a CNN extracting image features from high to low levels according to the network's depth. Then, the decoder path is used to restore extracted features to

the original image resolution and output the final segmentation results. To avoid losing information from the initial features extracted, skip connections are performed from the encoder path to the decoder path at the same level to fuse low- and high-resolution information. At this point a problem might be present due to the large semantic gap between features but can be solved by performing some additional convolution operations after fusing. Currently, the U-Net has become the benchmark for most medical image segmentation tasks and has inspired a lot of meaningful improvements [14].

The regularization techniques implemented seek to improve results. In [29] it is shown that, after testing many configurations, the use of batch normalization and spatial drop-out provides with the best results.

- **Batch normalization**: this step is necessary after each convolution operation in order to standardise the inputs, which means that inputs should have approximately zero mean and unit variance. Mathematically, it transforms each input in the current mini-batch by subtracting the input mean in the current mini-batch and dividing it by the standard deviation. This operation helps the network training by restricting the distribution of the input data to any particular layer, which helps the network to produce better gradients for weights update. Hence batch normalization often provides a much stable and accelerated training regime [30].
- **Spatial drop-out**: this operation is performed after each convolution operation. It works by probabilistically removing inputs to a layer, which may be input variables in the data sample or activations from a previous layer. It has the effect of simulating a large number of networks with very different network structure and, in turn, making nodes in the network generally more robust to the inputs. It helps the network generalise better during training and can drastically reduce the chance of overfitting. In this case, instead of removing individual units, spatial dropout drops entire 2D feature maps. If adjacent pixels within feature maps are strongly correlated it will help promote independence between feature maps.
- **Padding**: the padding operation allows to preserve the original input size of an image after a convolution operation. It consists of adding a border of pixels with value zero around the edges of the image after the convolution. The number of pixels that need to be added to the edges depends on the kernel size used on the convolution. As the kernel used in the model is 3x3, the number of pixels added must be 1 to preserve the size of the image. In *Figure 33*, it is observed a cropping step before the skip connections, however by adding a padding of 1 on each convolution there is no need for cropping as size is preserved through all the network.

4.2.2 W-net

The W-net implemented in this project was based on the structure proposed by [31], which is based on concatenating together two fully convolutional networks into an encoder-decoder framework, where each of the FCNs are variants of the U- Net architecture. This decision is based on the outperformance of this model over a number of existing classical and recent techniques. Also, W-net architecture has proven to be working well on several segmentation problems applied to biomedical images.

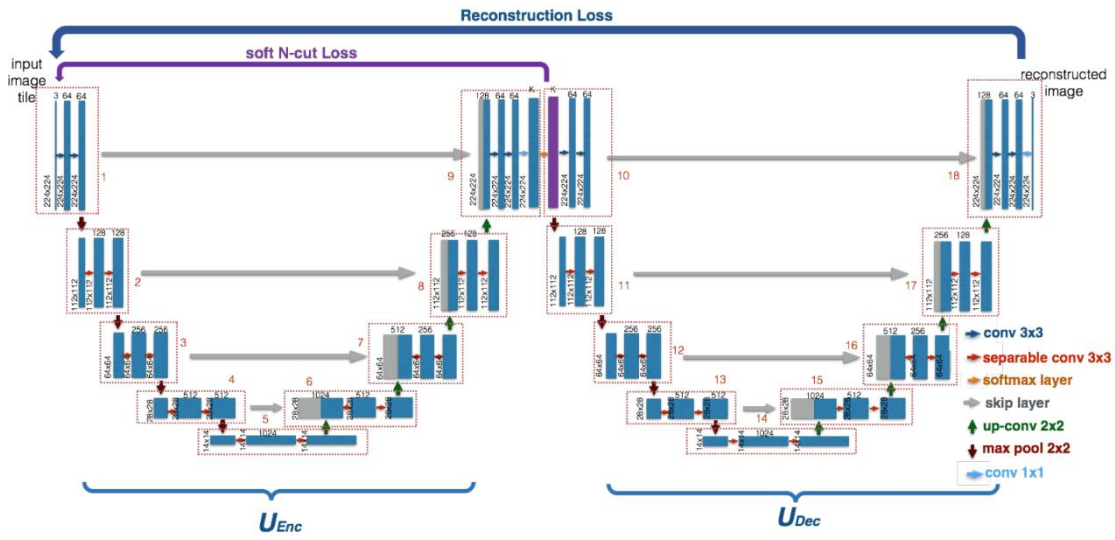


Figure 34: W-net architecture [31].

The W-net structure (*Figure 34*) is divided into two U-shaped paths which correspond to the encoder (U_{Enc}) on the left side and the decoder (U_{Dec}) on the right side. The working principle of each convolutional block is the same as the previous U-net architecture, as well as the downsampling, upsampling and skip connections. The main difference would be the transition between the encoder and the decoder, where a softmax layer is applied. This layer rescales the output of the encoder so that the elements lie in the range (0,1) and sum to 1. The decoder receives this output. The final convolutional layer is a 1x1 convolution to map the feature vector back to a reconstruction of original input. For the implementation of this architecture, the loss function used will be different from the ones appearing in *Figure 34*, as only a Dice loss function will be computed as a reconstruction loss from the end of the decoder path to the start of the encoder path.

4.3 Training

For the training part the coding language used will be *Python* and the main library used will be *PyTorch*. Even though there are other DL libraries such as *Keras* or *TensorFlow*, *PyTorch* has a very similar structure as *Numpy*, which makes it easier to use, and also the research group had already been working on this project using *PyTorch*.

The work on the training of the models is going to be based on a previous code repository generated by the research group which had already been tested. The modifications regarding the data representation will be performed on the python file for training and the modification of the model parameters for training will be performed on the configuration file (*json* format).

The training of the models will be performed by the high performance computing cluster (HPC) and so the jobs will be sent via VPN connection.

4.4 Performance evaluation metrics

In this section several evaluation metrics are going to be analysed. The main goal is to measure the performance of the different models trained. This evaluation will be applied to the model when predicting the images from the test dataset, where the prediction will be compared to the ground truth of the image.

Depending on the model's data representation, the metrics are divided into imaging-based and regression.

4.4.1 Imaging based metrics

4.4.1.1 Hausdorff distance

The Hausdorff distance measures how far two subsets from the same metric space are from each other. In other words, measures how similar two objects are (images, etc). Informally, the Hausdorff distance is the greatest of all distances from a point in one set to the closest point in the other set. Mathematically, X and Y are non-empty subsets of a metric space and the Hasudorff distance is defined to be [32]

$$d_{\mathrm{H}}(X,Y) = \max\left\{ \sup_{x\in X} d(x,Y),\ \sup_{y\in Y} d(X,y) \right\}.$$

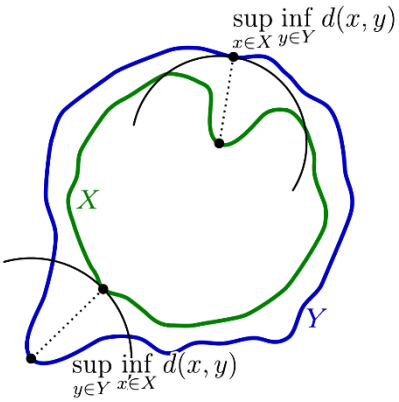


Figure 35: Computation of the Hausdorff distance between 2 images.

This distance is a widely used as a model performance measure for segmentation problems. It allows to compare binary images, such as the ground truth with the prediction generated by the trained model. In our case, this evaluation method would allow to measure the distances using the binary masks directly instead of having to transform them into unidimensional curves. However, it needs to be pointed out that the presence of outliers might lead to a bad evaluation as the final measure corresponds to the maximum distance which could correspond to an outlier.

4.4.1.2 Dice Coefficient

The Dice Coefficient is a highly used evaluation metric for semantic segmentation. It is computed as two times the area of overlap between the two images divided by the total number of pixels in both images [33]. This metric would also avoid the process of transforming the binary images into curves as it directly compares the images

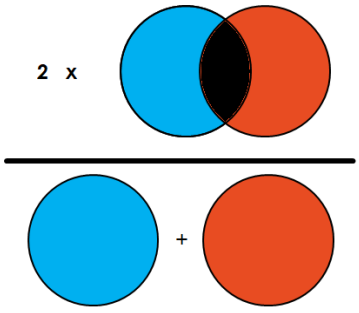


Figure 36: Graphic explanation of the computation of the Dice coefficient.

4.4.2 Regression metrics

4.4.2.1 Root Mean Square Error

The root mean square error (RMSE) is the square of the mean of the square of all the error. The use of RMSE is very common and it is considered an excellent general-purpose error metric for numerical predictions. Formally it is defined as follows, where $\hat{y}$ are the predicted values, *y* the observed values and *n* the number of observations.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

4.4.2.2 Mean Absolute Percentage Error

The mean absolute percentage error (MAPE) is one of the most popular metrics for evaluating forecasting performance.

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y - \hat{y}}{y} \right|$$

It computes the mean absolute error and converts it to a percentage. The use of the absolute value provides this metric with robustness regarding the effects of outliers. However, it needs to be taken into account that the division operation forces the result to be undefined for data point of value 0 [34].

4.4.2.3 Cross-correlation

The cross-correlation metric tracks the movement of two sets of data relative to one another. It compares those sets and objectively determines how well they match up. This metric does not consider how the sets are scaled, only their evolution. The possible range for the correlation coefficient is from -1 to 1, where 1 reflects identical data sets [35].

5. Detailed engineering

5.1 Implemented architectures

In this section all the architectures implemented in this project are developed. The following models have been designed from a common python file which allows to modify several parameters to train different models and test their performance. These parameters are: number of profundity levels of the network, initial number of channels, number of channels at the output and number of convolutions per level.

The feature number on the first convolution for each network has been chosen according to the training capacity of the computer as well as the batch size (explained in 5.2). By using padding the initial size of the images is conserved through all the steps.

According to the different data representations, when models use the linear regression an additional layer is implemented. A Soft Argmax operation layer is added at the end of the network generating an unidimensional vector output.

5.1.1 U-net with 5 levels

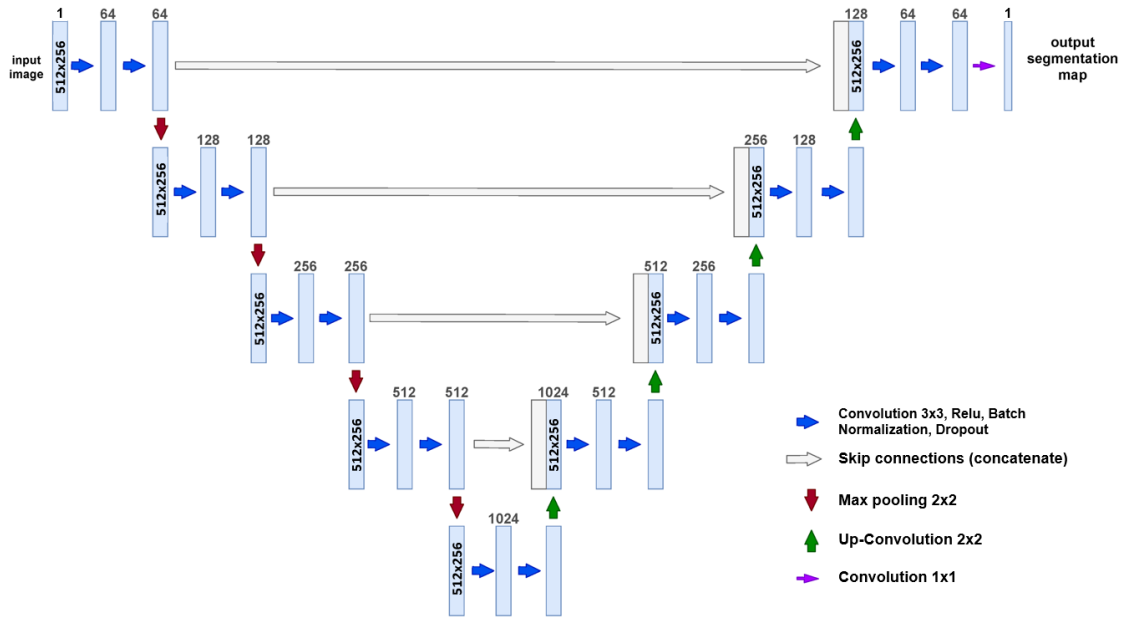


Figure 37: U-net architecture with 5 levels.

5.1.2 U-net with 6 levels

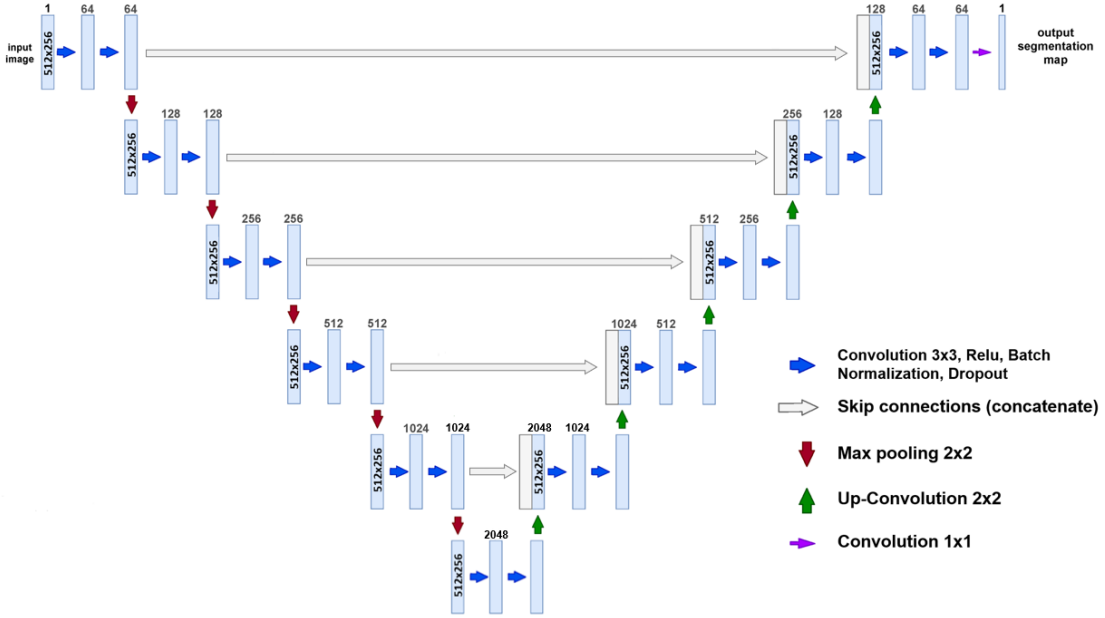


Figure 38: U-net architecture with 6 levels.

5.1.3 U-net with 7 levels

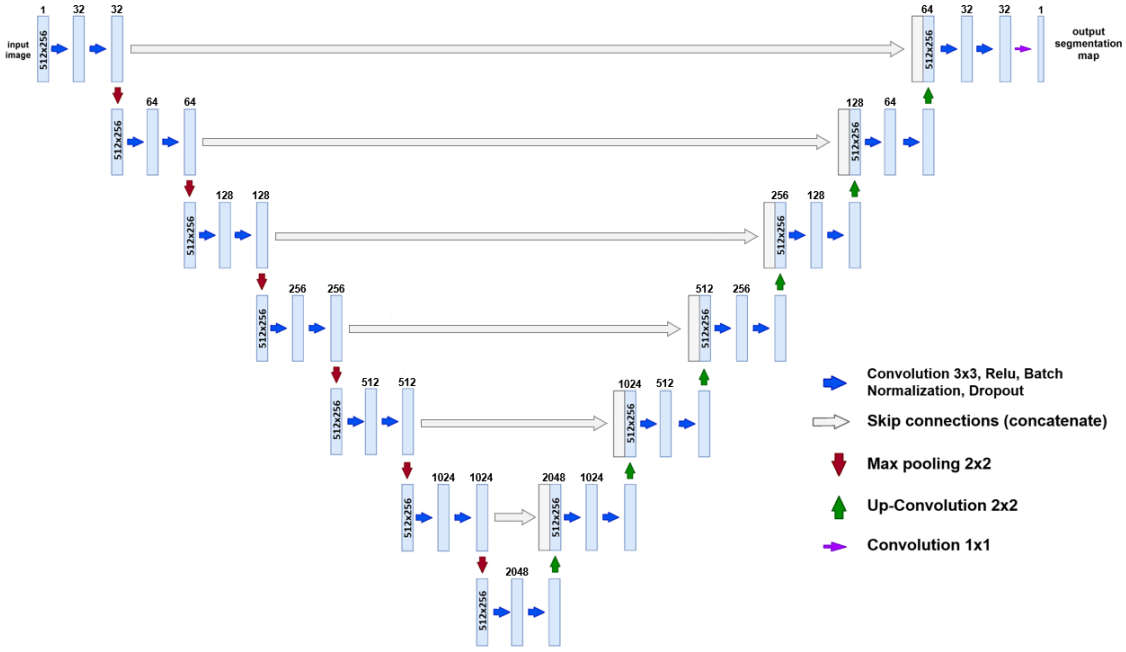


Figure 39: U-net architecture with 7 levels.

5.1.4 W-net with 5 levels

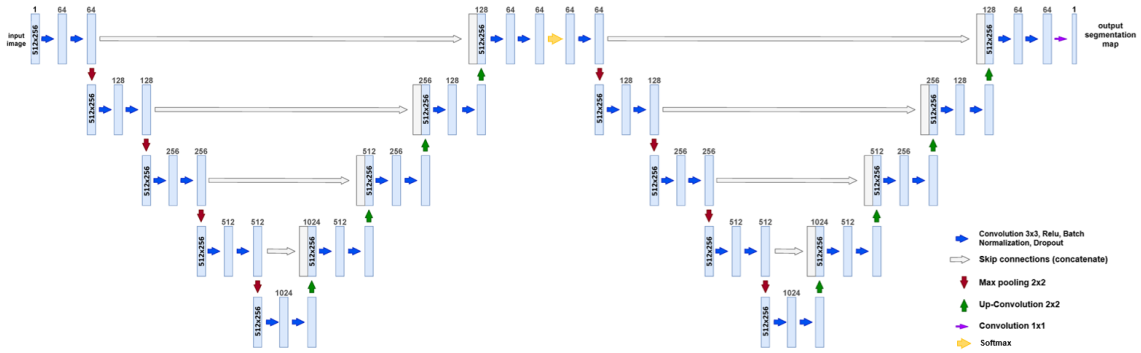


Figure 40: W-net architecture with 5 levels.

5.1.5 W-net with 6 levels

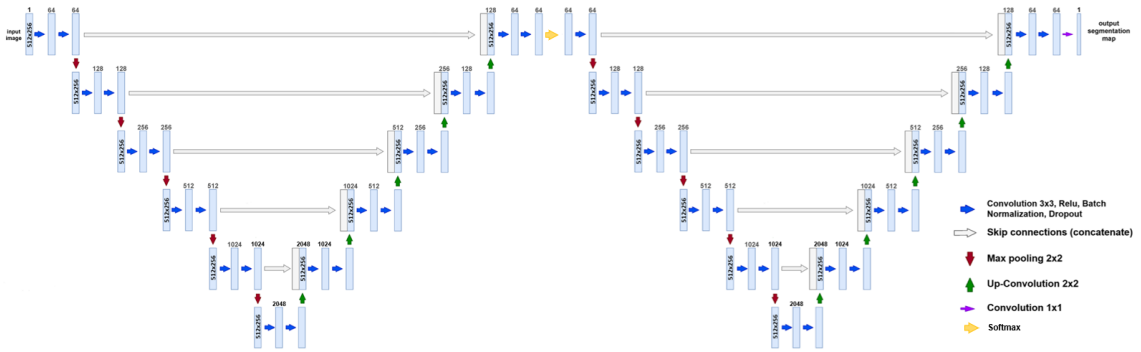


Figure 41: W-net architecture with 6 levels.

5.1.6 W-net with 7 levels

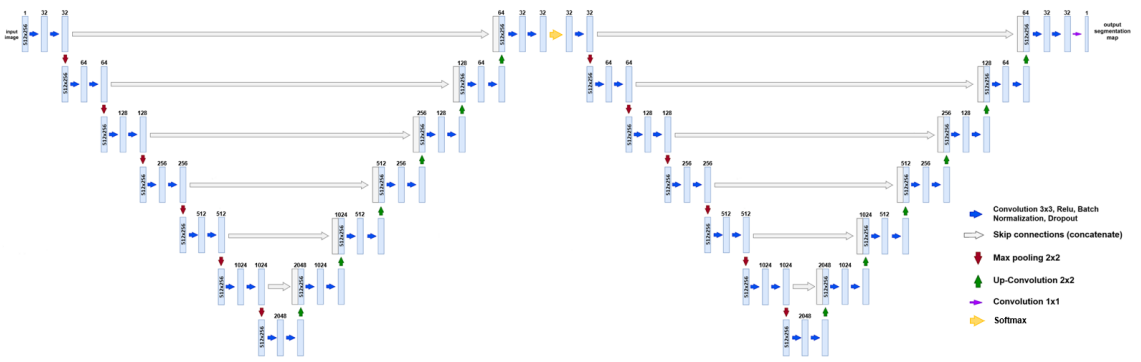


Figure 42: W-net architecture with 7 levels.

5.2 Training strategy

5.2.1 Validation test

Firstly, the correct functionality of the network was tested. In order to do so, the *5.1.4* model was chosen. As all the networks are based on the same structure testing the performance of one is enough. This validation test was based on training the model with the mask that is currently being

implemented by the research group. This type of data representation was already validated previously to this project.

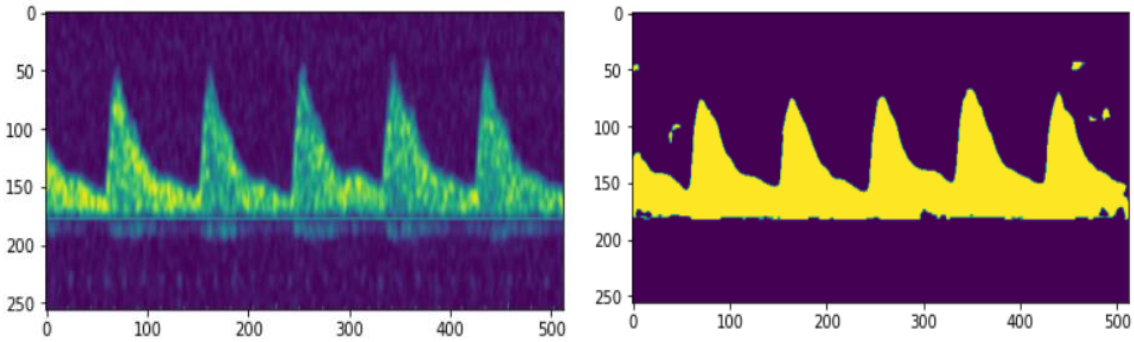


Figure 43 and 44: First image is the curve predicted by the model and second image is the prediction.

As it can be observed the prediction is good. Which means that the network working adequately.

5.2.2 Calibration

Due to the computational capacity limitations of the HPC, each model had to be trained with several parameters in order to choose the optimal ones. These parameters to choose were: initial number of channels, batch size and RAM memory used.

After several trials, the parameters appeared to be mainly dependent on the profundity level.

Table 2: Training parameters chosen.

Model	Number of channels	Batch size	RAM memory
5 level network	64	4	64 G
6 level network	64	4	64 G
7 level network	32	4	64 G

5.2.3 Job strategy

Table 3: Job strategy.

Model	Data representation	Job order / Model ID
U-net with 5 levels	Binary mask with 20 pixel width	1
U-net with 5 levels	Binary mask with 30 pixel width	2
U-net with 5 levels	Binary mask with 40 pixel width	3
U-net with 6 levels	Binary mask with 20 pixel width	4
U-net with 6 levels	Binary mask with 30 pixel width	5
U-net with 6 levels	Binary mask with 40 pixel width	6
U-net with 7 levels	Binary mask with 20 pixel width	7
U-net with 7 levels	Binary mask with 30 pixel width	8
U-net with 7 levels	Binary mask with 40 pixel width	9
W-net with 5 levels	Binary mask with 20 pixel width	10
W-net with 5 levels	Binary mask with 30 pixel width	11
W-net with 5 levels	Binary mask with 40 pixel width	12
W-net with 6 levels	Binary mask with 20 pixel width	13
W-net with 6 levels	Binary mask with 30 pixel width	14
W-net with 6 levels	Binary mask with 40 pixel width	15

W-net with 7 levels	Binary mask with 20 pixel width	16
W-net with 7 levels	Binary mask with 30 pixel width	17
W-net with 7 levels	Binary mask with 40 pixel width	18
U-net with 5 levels	Linear regression	19
U-net with 6 levels	Linear regression	20
U-net with 7 levels	Linear regression	21
W-net with 5 levels	Linear regression	22
W-net with 6 levels	Linear regression	23
W-net with 7 levels	Linear regression	24

5.3 Implemented metrics

The performance evaluation of the models is performed using the metrics described in *4.4*. This evaluation is based on using the images from the test set and comparing the predictions of the model with the ground truths. For the images, the same pre-processing steps must be performed as in *4.1.1*.

For the computation of the metrics, the dice coefficient function has been defined in python, the hausdorff distance function used is from the skimage library [36], the root mean square function used if from the torch library [37], the MAPE function is used from the sklearn library [38] and the cross correlation function has been defined in python.

5.4 Model results

In this section it is shown the segmentation performance of the models trained. This performance is measured by the metrics proposed. Also, each model performance is illustrated observing the segmentation of a randomly selected image from the test dataset:

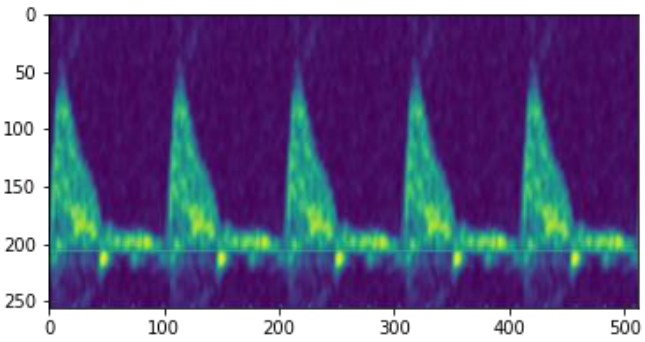


Figure 45: Random selected image.

The results obtained are being compared to the performance evaluation of the baseline solution. This model is a W-net network and uses the baseline data representation that uses a binary mask from the reference line to the curve position. The metric values are the following:

Table 4: Baseline solution metrics.

Dice coefficient
0.9037749568759134
Hausdorff distance
42.27339812812656

RMSE
13.079955740447383
MAPE
0.06595673562824161
Cross correlation
0.9941020288719871

5.4.1 Model ID: 1

Table 5: Metric values for Model 1.

Dice coefficient
0.2888593779634916
Hausdorff distance
112.77724301708936

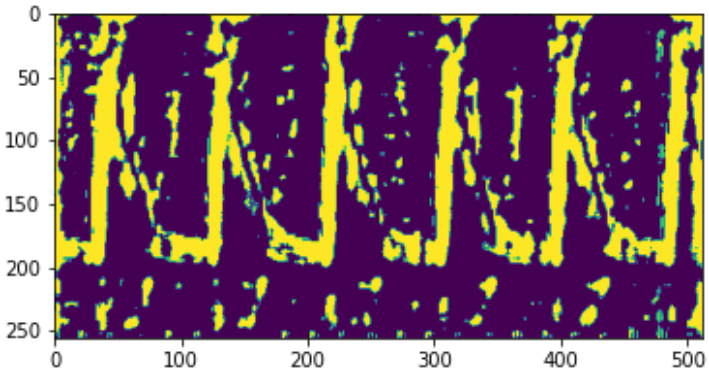


Figure 46: Prediction of Model 1.

5.4.2 Model ID: 2

Table 6: Metric values for Model 2.

Dice coefficient
0.0290460034543923
Hausdorff distance
108.39900992723237

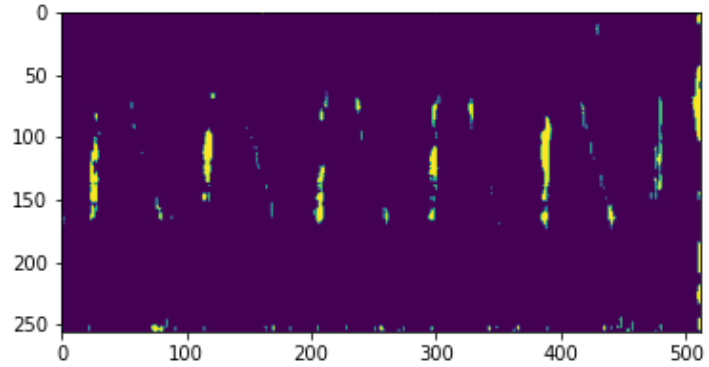


Figure 47: Prediction of Model 2.

5.4.3 Model ID: 3

Table 7: Metric values for Model 3.

Dice coefficient
0.029562358012144515
Hausdorff distance
103.45635826660497

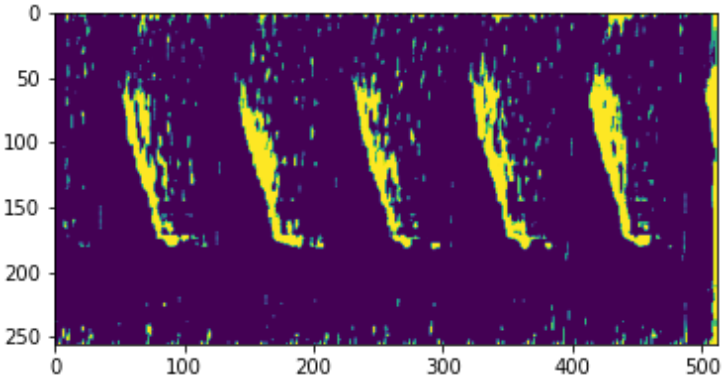


Figure 48: Prediction of Model 3.

5.4.4 Model ID: 4

Table 8: Metric values for Model 4.

Dice coefficient
0.055413129533081605
Hausdorff distance
112.96051780423169

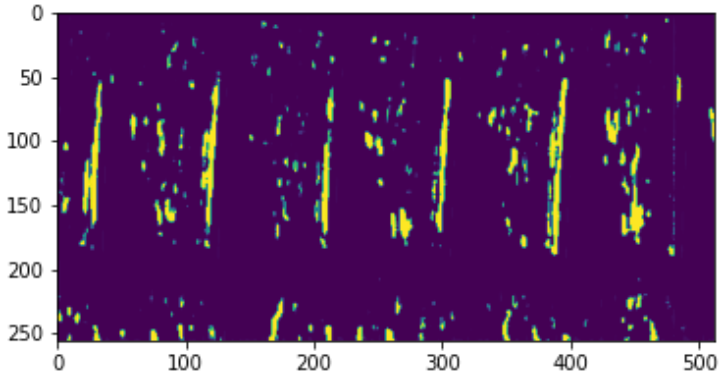


Figure 49: Prediction of Model 4.

5.4.5 Model ID: 5

Table 9: Metric values for Model 5.

Dice coefficient
0.2688098856245765
Hausdorff distance
107.38066416850435

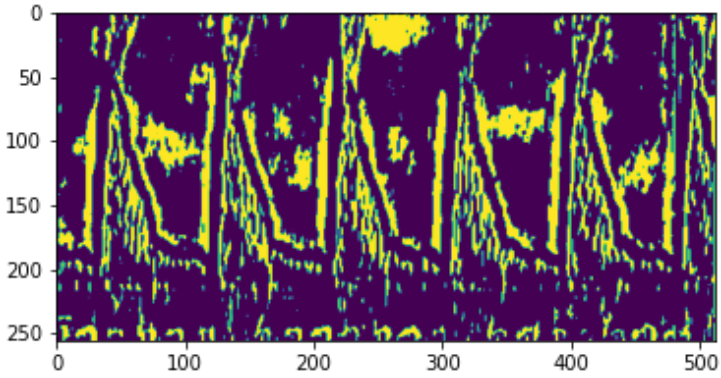


Figure 50: Prediction of Model 5.

5.4.6 Model ID: 6

Table 10: Metric values for Model 6.

Dice coefficient
0.4808080053428538
Hausdorff distance
101.89651283079445

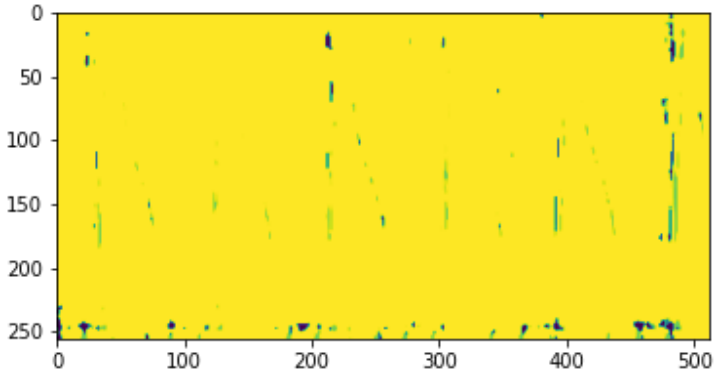


Figure 51: Prediction of Model 6.

5.4.7 Model ID: 7

Table 11: Metric values for Model 7.

Dice coefficient
0.1515928791189047
Hausdorff distance
112.80008559554199

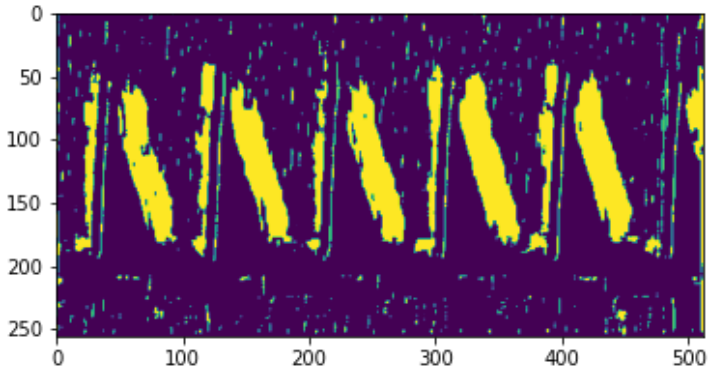


Figure 52: Prediction of Model 7.

5.4.8 Model ID: 8

Table 12: Metric values for Model 8.

Dice coefficient
0.3906598849433161
Hausdorff distance
107.3517679103469

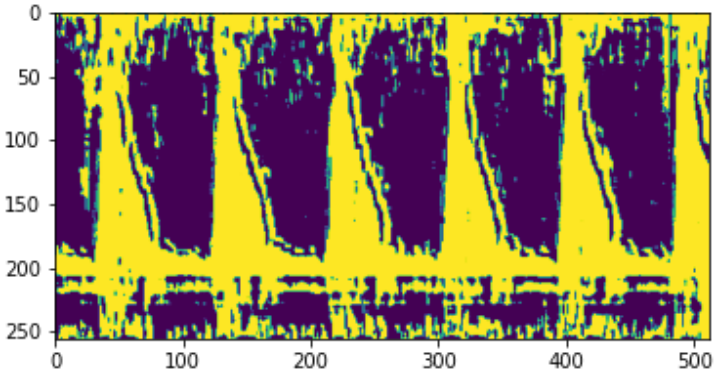


Figure 53: Prediction of Model 8.

5.4.9 Model ID: 9

Table 13: Metric values for Model 9.

Dice coefficient
0.38333541319025305
Hausdorff distance
101.77876045003754

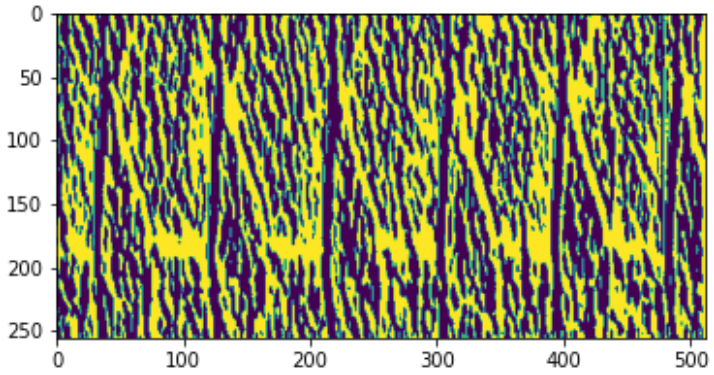


Figure 54: Prediction of Model 9.

5.4.10 Model ID: 10

Table 14: Metric values for Model 10.

Dice coefficient
0.17122470264677275
Hausdorff distance
112.9839369615932

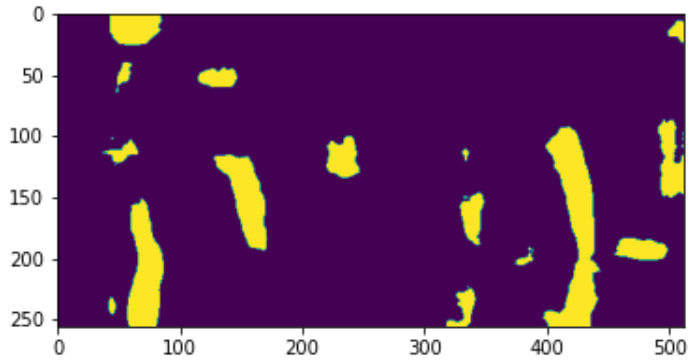


Figure 55: Prediction of Model 10.

5.4.11 Model ID: 11

Table 15: Metric values for Model 11.

Dice coefficient
0.1936857640737967
Hausdorff distance
107.43865191959758

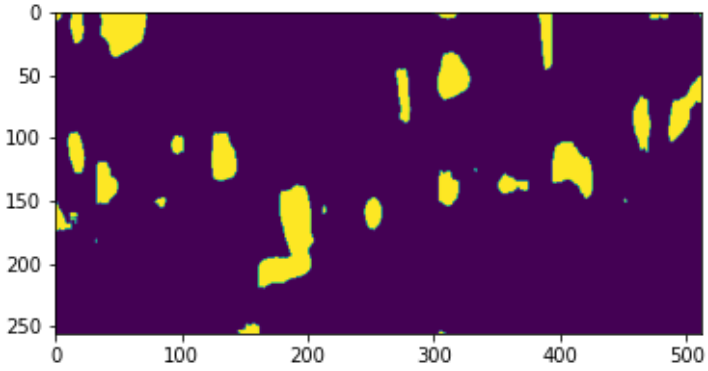


Figure 56: Prediction of Model 11.

5.4.12 Model ID: 12

Table 16: Metric values for Model 12.

Dice coefficient
0.2669584902466908
Hausdorff distance
101.89413559433615

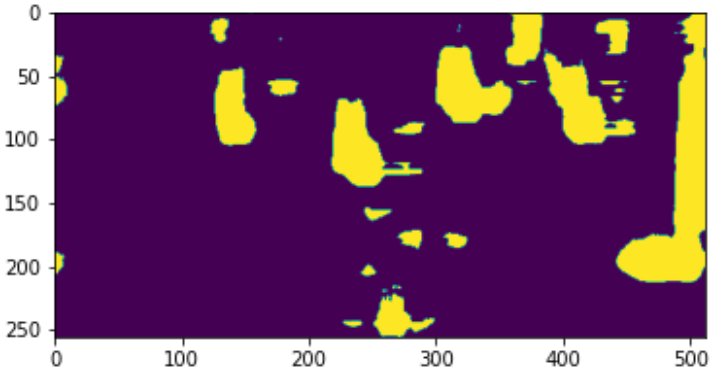


Figure 57: Prediction of Model 12.

5.4.13 Model ID: 13

Table 17: Metric values for Model 13.

Dice coefficient
0.14464378810129014
Hausdorff distance
112.9839369615932

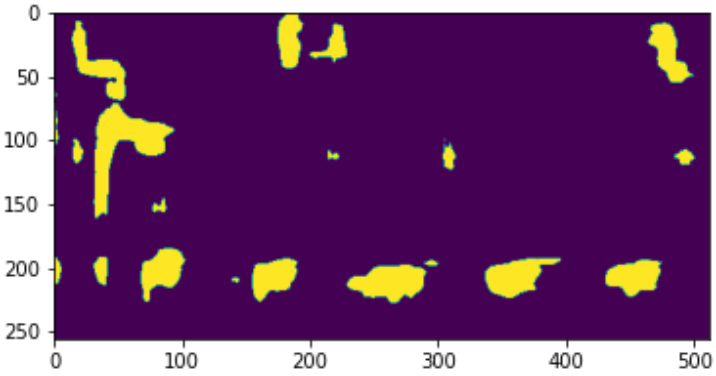


Figure 58: Prediction of Model 13.

5.4.14 Model ID: 14

Table 18: Metric values for Model 14.

Dice coefficient
0.2904967162514867
Hausdorff distance
107.43865191959758

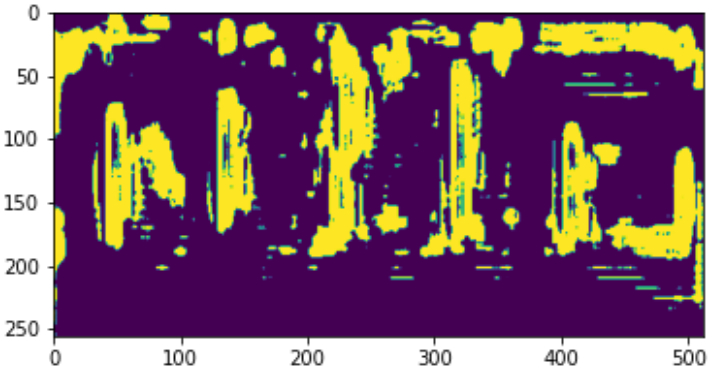


Figure 59: Prediction of Model 14.

5.4.15 Model ID: 15

Table 19: Metric values for Model 15.

Dice coefficient
0.02896816555805188
Hausdorff distance
101.89651283079445

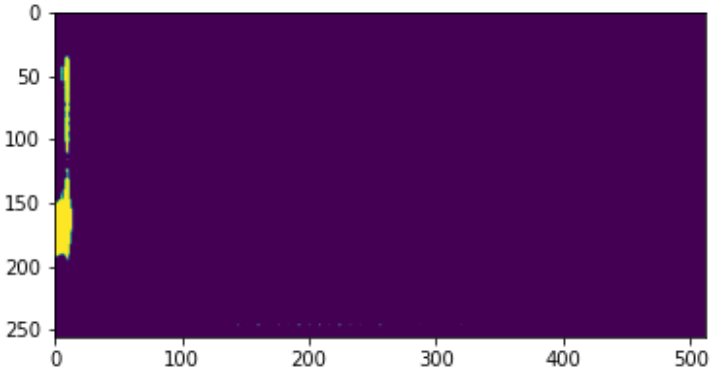


Figure 60: Prediction of Model 15.

5.4.16 Model ID: 16

Table 20: Metric values for Model 16.

Dice coefficient
0.006557661217977295
Hausdorff distance
112.9839369615932

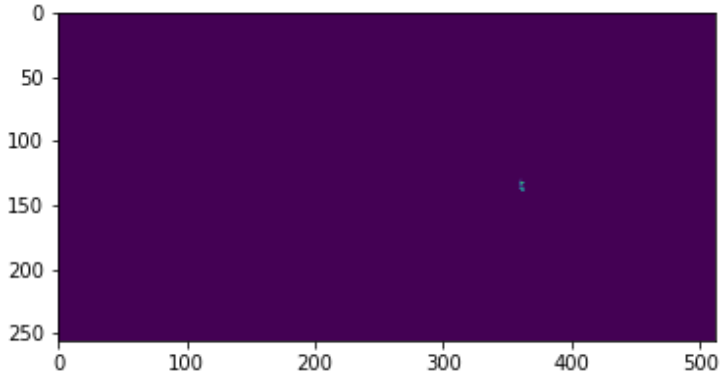


Figure 61: Prediction of Model 16.

5.4.17 Model ID: 17

Table 21: Metric values for Model 17.

Dice coefficient
0.2334054929548093
Hausdorff distance
107.43865191959758

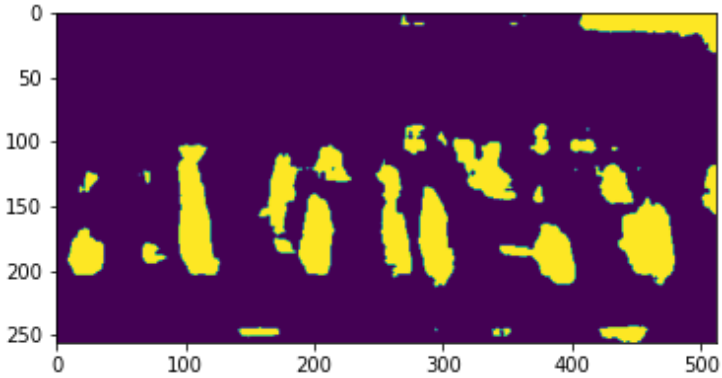


Figure 62: Prediction of Model 17.

5.4.18 Model ID: 18

Table 22: Metric values for Model 18.

Dice coefficient
0.5579057875787617
Hausdorff distance
101.89651283079445

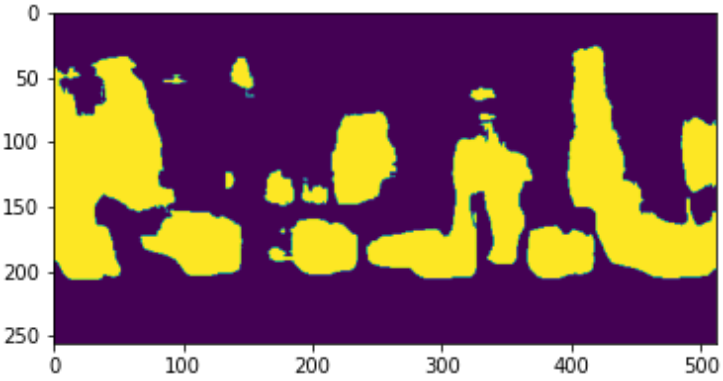


Figure 63: Prediction of Model 18.

5.4.19 Model ID: 19

Table 23: Metric values for Model 19.

MAPE
0.613529644502693
Cross correlation
0.475845873782959
RMSE
54.16038780523438

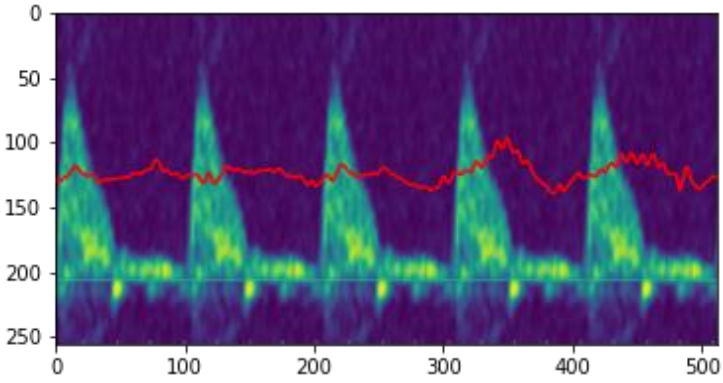


Figure 64: Prediction of Model 19.

5.4.20 Model ID: 20

Table 24: Metric values for Model 20.

MAPE
0.6733614099206194
Cross correlation
0.4451213431361262
RMSE
78.58893781959979

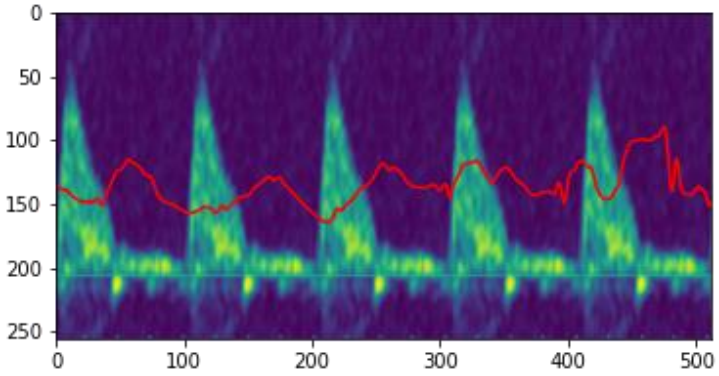


Figure 65: Prediction of Model 20.

5.4.21 Model ID: 21

Table 25: Metric values for Model 21.

MAPE
0.41635479331792047
Cross correlation
0.4688914738762607
RMSE
49.631254910544804

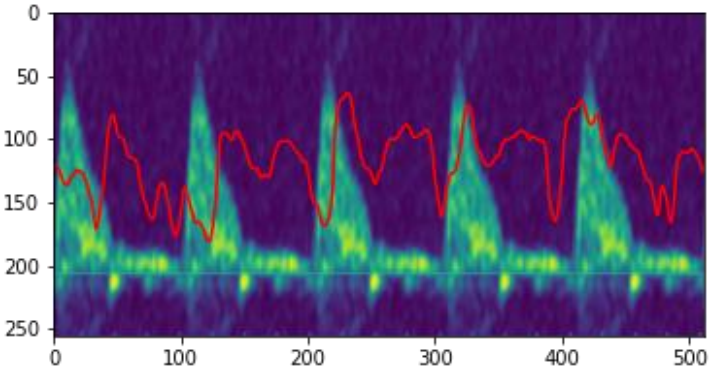


Figure 66: Prediction of Model 21.

5.4.22 Model ID: 22

Table 26: Metric values for Model 22.

MAPE
0.46045785743531203
Cross correlation
0.477294404839291
RMSE
46.20067670080675

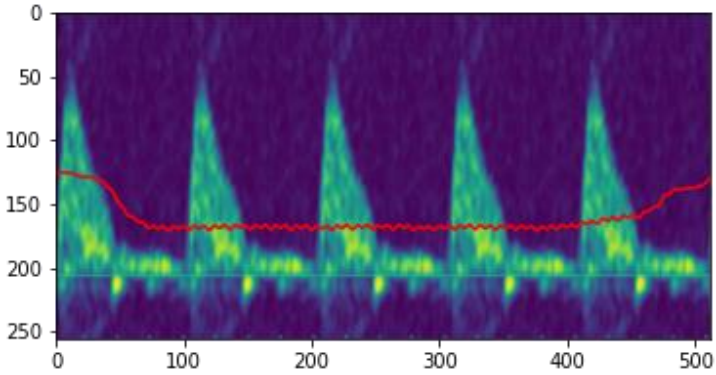


Figure 67: Prediction of Model 22.

5.4.23 Model ID: 23

Table 27: Metric values for Model 23.

MAPE
0.4353168228056141
Cross correlation
0.46989182126151186
RMSE
48.36919890598897

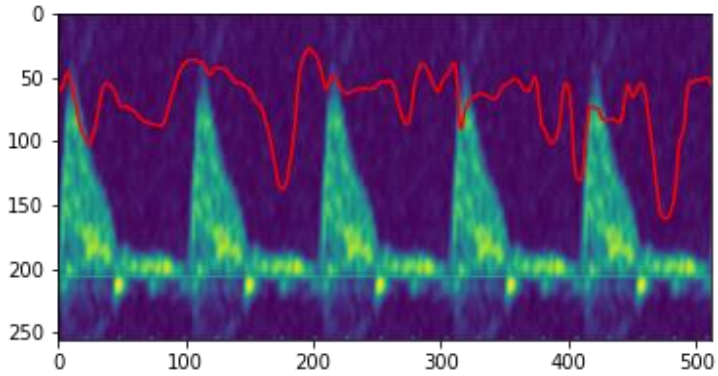


Figure 68: Prediction of Model 23.

5.4.24 Model ID: 24

Table 28: Metric values for Model 24.

MAPE
0.41409849265332593
Cross correlation
0.47048364551350375
RMSE
46.253778589329926

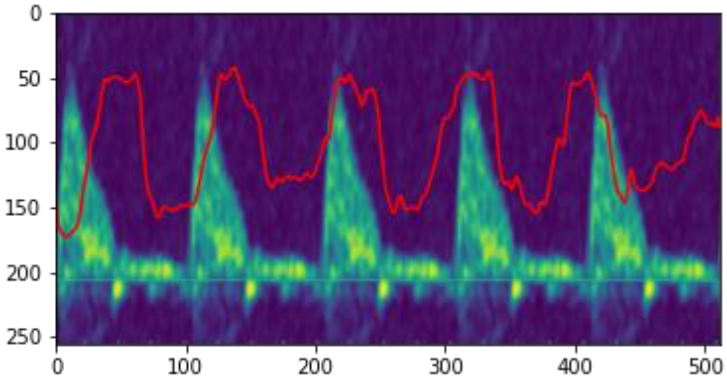


Figure 69: Prediction of Model 24.

5.4.25 Summary

Table 29: Summary of the performance evaluation (Models 1 to 18).

Model ID	Dice coefficient	Hausdorff distance
1	0.2888593779634916	112.77724301708936
2	0.0290460034543923	108.39900992723237
3	0.029562358012144515	103.45635826660497
4	0.055413129533081605	112.96051780423169
5	0.2688098856245765	107.38066416850435
6	0.4808080053428538	101.89651283079445
7	0.1515928791189047	112.80008559554199
8	0.3906598849433161	107.3517679103469
9	0.38333541319025305	101.77876045003754
10	0.17122470264677275	112.9839369615932
11	0.1936857640737967	107.43865191959758
12	0.2669584902466908	101.89413559433615
13	0.14464378810129014	112.9839369615932
14	0.2904967162514867	107.43865191959758
15	0.02896816555805188	101.89651283079445
16	0.006557661217977295	112.9839369615932
17	0.2334054929548093	107.43865191959758
18	0.5579057875787617	101.89651283079445

Table 30: Summary of the performance evaluation (Models 19 to 24).

Model ID	MAPE	Cross correlation	RMSE
19	0.613529644502693	0.475845873782959	54.16038780523438
20	0.6733614099206194	0.4451213431361262	78.58893781959979
21	0.41635479331792047	0.4688914738762607	49.631254910544804
22	0.46045785743531203	0.477294404839291	46.20067670080675
23	0.4353168228056141	0.4698918212615118	48.36919890598897
24	0.41409849265332593	0.4704836455135037	46.253778589329926

The results obtained for each model are illustrated with a representative example alongside the metrics. The dice coefficient shows better results for models 6, 8, 9 and 18. However, these results do not relate to the hausdorff distance values. The representative examples of these models do show a slight improvement over the other, although the segmentation is not optimal.

Regarding the linear regression, models 21 to 24 present better metric values for MAPE and RMSE than models 19 and 20. Although the best results are not optimal, these results may suggest a better performance of more complex architectures with larger depth when using linear regression as data representation.

The results obtained for the alternative models proposed can be compared to the metric values of the baseline solution in *Table 4*. It is clear that no model improves the performance of the baseline solution as the metric values are worse for the models proposed.

To analyse better the metric values, the mean and standard deviation (Std) of some aggrupation of models is calculated. The comparisons are made between network architectures (U-net and W-net), network depth (5, 6 and 7 levels) and the width of the binary mask tested (20, 30 and 40 pixels).

Table 31: Network architecture comparison table.

	U-net		**W-net**	
Metric	Mean	Std	Mean	Std
Dice	0.230898548575	0.161375125165	0.210427396514	0.153301092176
Hausdorff distance	107.6445466633	4.306281378336	107.4394364332	4.52674552818
MAPE	0.567748615913	0.109802936142	0.436624390964	0.01894870227
Cross correlation	0.463286230265	0.013154549601	0.472556623871	0.00335881828
RMSE	60.79352684512	12.71837908978	46.94121806537	1.009967627702

The comparison between network architectures shows better mean value of the metrics for the W-net, although the Dice coefficient points the other way. It is important to notice that the differences between both architectures are not significantly big. Also, most values show a poor performance of the models.

Table 32: Network depth comparison table.

	5 levels		**6 levels**		**7 levels**	
Metric	Mean	Std	Mean	Std	Mean	Std
Dice	0.16322278	0.1027993	0.21152328	0.15502818	0.2872428	0.17947647
Hausdorff distance	107.824889	4.2007593	107.426132	4.52169257	107.37495	4.51346204
MAPE	0.53699375	0.0765358	0.55433911	0.11902229	0.4152266	0.00112815
Cross correlation	0.47657013	0.0007242	0.45750658	0.01238523	0.4696875	0.00079608
RMSE	50.1805322	3.979855	63.4790683	15.1098694	47.942516	1.68873816

When comparing the mean values of the metrics between models using different depth of the network, it is shown that most metrics point out a better performance when using 7 level networks. However, the same behaviour as the previous comparison it is observed, where differences are not significantly big and metric values show a non-optimal performance.

Table 33: Mask width comparison table.

	20 pixels		**30 pixels**		**40 pixels**	
Metric	Mean	Std	Mean	Std	Mean	Std
Dice	0.13638192	1.6887381	0.23435062	0.10998251	0.2912563	0.20549920
Hausdorff distance	112.914942	0.0899176	107.57456	0.37021674	102.13646	0.59182281

This comparison table focuses on the binary mask proposed in this project as a data representation, which is based on creating a mask of different width around the curve position. The results show that better mean metric values are observed with models using masks with bigger width value, 40 pixels.

6. Execution chronogram

In this section it is defined the timing regarding tasks and the organization of the project. Initially, the project was meant to be developed in a time range of one semester, although due to access problems to a suitable computational environment (explained in *1.2*) delayed the start of the project. This force to modify the duration and dates for all the tasks. Finally, the project started in November 2021 and finished in June 2022.

6.1 Work breakdown structure (WBE)

Decomposition of the tasks involved in the project:

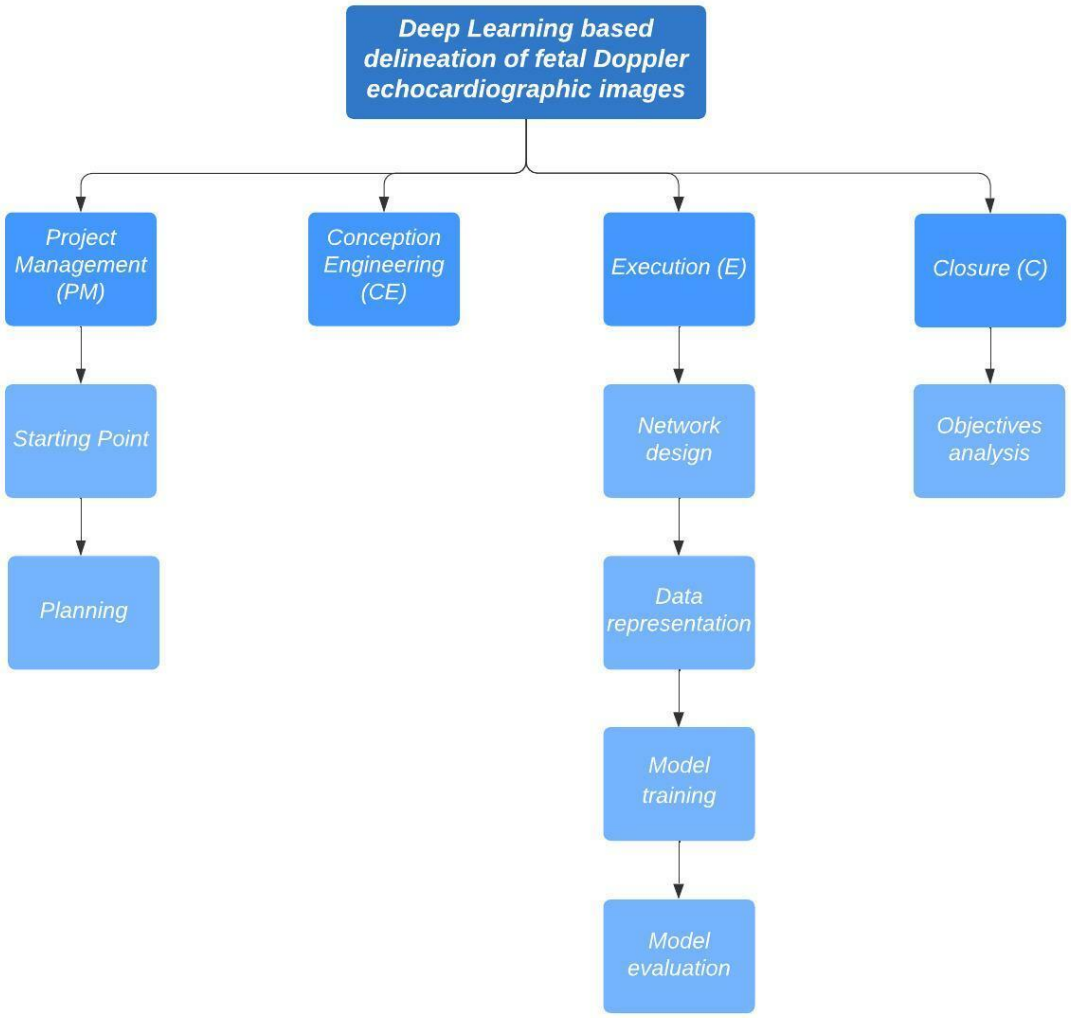


Figure 70: Work Breakdown Structure

6.2 WBE dictionary

Table 34: WBE dictionary.

Task	Description	Related activities
PM1 – Starting point	Creation of the project.	- Initial meeting with research group. - Definition of objectives.

PM2 – Planning	General planning of the project.	- Task definition. - Chronologic plan. - Programming of the meetings for monitoring and doubts. - Control checkpoints definition.
CE – Conception engineering	Analysis of all the previous information necessary to define and develop the project.	- State of the art and background analysis. - Market analysis. - Limitations. - Familiarization with Pytorch library - Network analysis. - Familiarization with the dataset. - Data representation analysis. - Familiarization with the use of the HPC cluster and the procedure for sending jobs. - Familiarization with the original code for training. - Evaluation metrics analysis.
E1 – Network design	Design of the chosen model architectures that will be trained.	- Development of a primary network. - Generalization of the network with several modifiable parameters. - Testing of the correct functioning of all the networks.
E2 – Data representation	Design of the chosen data representations.	- Data pre-processing. - Code development. - Testing of a good generalization for all images. - Error analysis and correction
E3 – Model training	Training of the different models chosen using the different data representations.	- Parameter modification of the original code according to the training designed for each model. - Error analysis and correction.
E4 – Model evaluation	Evaluation of the performance of each model trained using the metrics chosen.	- Code development for each metric. - Collection and analysis of the results.
C1 – Objective analysis	Checking the accomplishment of the goals defined at the beginning of the project.	

6.3 Precedence analysis

In this section it is presented the dependencies between the tasks mentioned above.

Table 35: Precedence analysis.

Task	Name	Antecedent	Durtion (weeks)
PM1	Starting point	-	1
PM2	Planning	PM1	8
CE	Conception engineering	PM2	11
E1	Network design	CE	10
E2	Data representation	CE	10
E3	Model training	E1/E2	5
E4	Model evaluation	E3	5
C1	Objective analysis	E4	4

6.4 GANTT chart

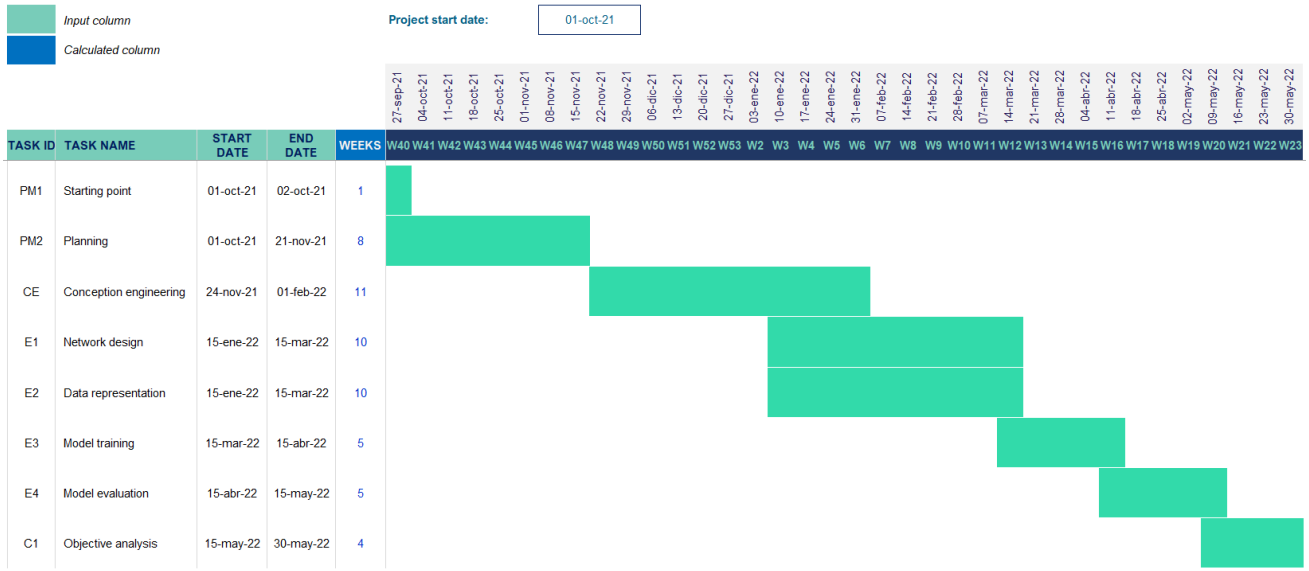


Figure 71: GANTT diagram.

7. Technical viability

7.1 Technical requirements

To perform this project, a high performance computation environment with several GPUs was needed, so that can the computational requirements regarding the training phase could be dealt with. This HPC is provided by the UPF and the connection is made via VPN in order to allow access from different locations.

Regarding the local server, it is needed to create a local virtual environment containing all the according libraries. This would allow to work and test code from the local server without having to connect to the HPC every time a code needs to be executed.

Furthermore, a pre-existing repository created by the research group needed to be downloaded. This repository contains all the files needed for training and it is downloaded via GitHub.

7.2 SWOT analysis

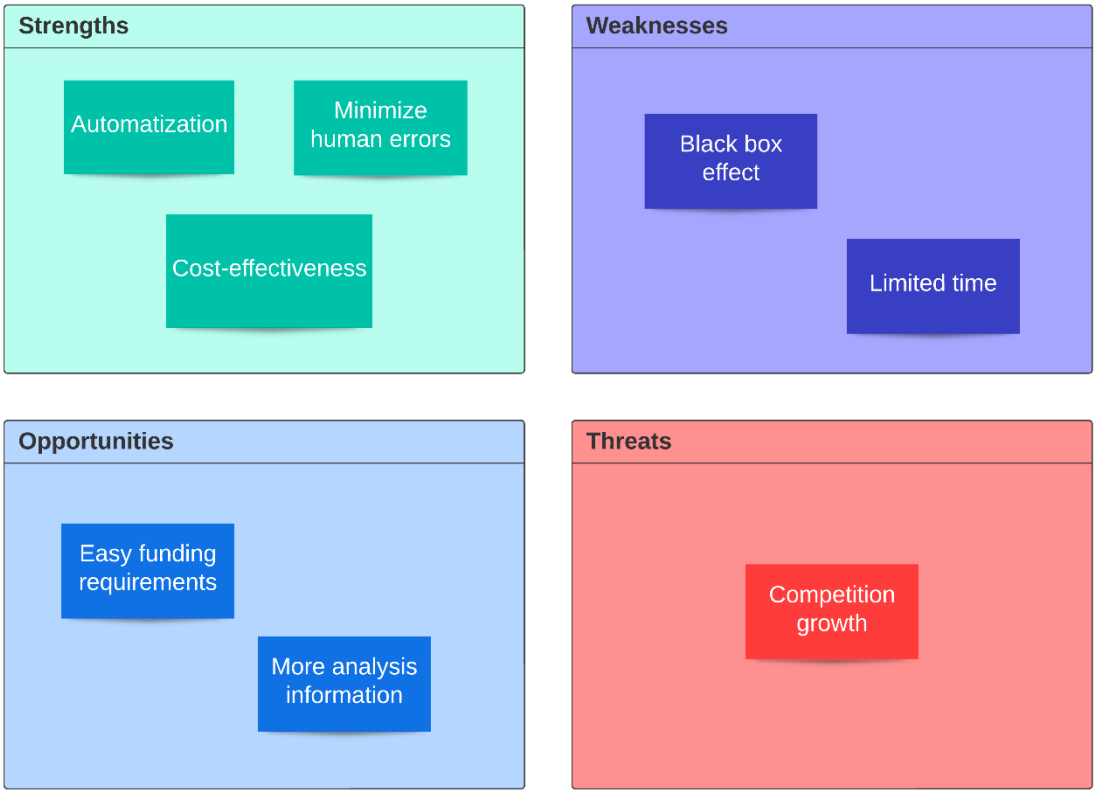


Figure 72: SWOT analysis.

7.2.1 Strengths

- **Less time consuming than current methods**: Current segmentation methods of doppler echocardiographic image involve manual delineation steps for the posterior analysis of the image. However, the proposed tool would provide with a pipeline for analysing these images much less time consuming. Furthermore, healthcare professionals would have less workload regarding automatic specialized tasks and their working time could be optimized.
- **Automatization**: Manual segmentation largely depends on the experience of the professional performing the task. Whereas the automatization of this task would

significantly reduce the influence of the individual experience and consequently decrease the rate of human errors.

- **Cost-effectiveness**: This tool would be extremely cost-effective as it would considerably reduce workforce and time, and at the same time improve performance.

7.2.2 Weaknesses

- **Black box effect**: DL models are provided with a sufficient complexity that they are not straightforwardly interpretable to humans. The lack of interpretability in predictive models can undermine trust in those models, especially in health care where decisions have a direct impact on the patient's health. Therefore, it is reasonable to think that healthcare professionals could be reluctant to such an opaque tool.
- **Limited time**: the structure and development of this project has been influenced by the limited timings. Further trials could have been made to reinforce results by testing the performance of other models based on different architectures and using different data representations.

7.2.3 Opportunities

- **Easy funding**: Echocardiographic images are one of the most commonly used imaging techniques, which an automatization of the segmentation for the analysis is a really interesting asset for many companies in the sector. Therefore, it would not be really hard to find funding.
- **More analysis information**: This segmentation tool not only would provide the information that is currently being analysed by the current method, but much more information of every image could be extracted and this would provide with more information for the posterior analysis that could be crucial.

7.2.4 Threats

- **Competition growth**: DL is rising extremely fast and the automatization of healthcare tasks is gaining a real interest. Therefore, everyday new studies and new techniques are being evaluated meaning that competition is rising.

8. Conclusions

This project's main objectives were to explore several alternative solutions to the existing pipeline for the generation of a DL-based algorithm that automatically delineates fetal Doppler echocardiographic images, in collaboration with the project awarded by the Bill and Melinda Gates Foundation. In order to accomplish this goal, several model architectures were design and alternative data representations were proposed. The literature did not present any DL-based Doppler segmentation algorithm. Therefore, the alternative solutions tested were not based on any specific algorithm previously implemented for this kind of problem.

Basing the analysis of the results to the performance evaluation metrics values, it is clear to conclude that the alternative data representations proposed do not show an improvement respect to the baseline solution. The Dice coefficient, which compares two images based on the area of overlap, does not present any result close to 1. Model 18 presents the best Dice coefficient with a value of 0.5579057875787617, whereas the baseline solution has a value of 0.9037749568759134. The Hausdorff distance measures the biggest distance between two points of the images, and all of them present values higher than 100, whereas the baseline solution has a value of 42.27339812812656. Regarding the mean absolute percentage error (MAPE), model 24 presents a value of 0.41409849265332593, however the baseline solution presents a much lower error with 0.06595673562824161. The cross correlation values from the models using linear regression are really similar, with the model 22 presenting the best number of 0.477294404839291. The baseline solution also outperforms the alternative solutions with a value of 0.9941020288719871. The RMSE presents a wider range of values where the minimum error is 46.20067670080675, but the baseline solution still seems to work better with an error of 13.079955740447383.

Analysing the tables with the different mean metric values of several model aggrupation, some tendencies can be observed. The network architecture comparison shows that W-net architecture presents a better performance than the U-net architecture. This result can be related to the findings observed when the baseline solution was tested, as the model that showed the best performance was also based on a W-net architecture. The network level comparison shows better performance in networks with larger number of levels. The baseline solution is based on a 6 level network. These findings might propose that performance could be improved using a deeper network with the baseline data representation. The mask width comparison shows that the proposed binary mask as data representation works better when the width number is higher. Although the results for this data representation are not optimal, it is observed that binary masks perform better when shapes are bigger. These findings support the fact that due to the use of bigger and more accurate shapes, the baseline solution performs much better.

These results need to be put into context in order to understand the problems faced for future research approaches. Initially, it needs to be said that this project is based on the basic knowledge of a final year student that just started to learn about DL. Thus, it needs to be taken into account that code implementations might have not been optimal when it comes to computational efficiency, and this has led to larger times of training and evaluation.

The lack of generalization of the models is hard to explain due to the black box effect, which is one of the main limitations when working with DL. The lack of interpretability of the models makes it much harder to understand what is going wrong and then be able to improve the generalization of the model to unseen inputs. However, the previous results show that the key must be the data representation to use. One important handicap is the presentation of the ground truth as the manual delineation of a single cardiac cycle, which leads to the tiling strategy. This forces the models to be trained by pre-processed images that contain identical cardiac cycles concatenated. However, when it comes to generalization to unseen data, the model predicts the segmentation of images containing different cardiac cycles. Following the results obtained, the models seem to not generalize when using the alternative data representation as it forms smaller shapes compared to the baseline solution.

With this project it has been proved that U-net and W-net based models of 5 to 7 profundity levels do not present good results for the delineation of Doppler echocardiographic fetal images when using the alternative data representations. For future research, we propose several approaches that might present better segmentation results for this problem. Firstly, an interesting line would be to test the linear regression data representation using other model architectures. Another line of research could be to use deeper W-net based models and explore the utilization other binary masks as data representation with bigger shapes. Also, U-net Transformer could be an architecture to look into which is starting to be applied in the medical imaging field [39].

Finally, it would be important to emphasize the impact this project has had at a personal level. Working alongside professionals of the field awards the opportunity to observe, learn and acquire huge amounts of knowledge. The project allowed for learning a new framework, PyTorch, and for researching state-of-the-art algorithms for image segmentation. Moreover, the achievement of goals through rigorous work provides with a wide range of skills incredibly valuable for the professional future.

9. Bibliography

[1] M. Zolgharni *et al.*, "Automated aortic doppler flow tracing for reproducible research and clinical measurements," *IEEE Transactions on Medical Imaging*, vol. 33, no. 5, pp. 1071–1082, 2014, doi: 10.1109/TMI.2014.2303782.

[2] "Deep Learning vs Machine Learning: What's the Difference." https://www.computer.org/publications/tech-news/trends/deep-learning-vs-machine-learning-whats-the-difference (accessed May 19, 2022).

[3] "Rocket: una plataforma que incide en la toma de decisiones clínicas de los profesionales de la salud - categorías - Archivo de noticias (UPF)." https://www.upf.edu/es/web/e-noticies/categorias/-/asset_publisher/wEpPxsVRD6Vt/content/id/202271829/maximized#.YotUv6hBxPa (accessed May 23, 2022).

[4] J. É. S. Kenny *et al.*, "A novel, hands-free ultrasound patch for continuous monitoring of quantitative Doppler in the carotid artery," *Scientific Reports 2021 11:1*, vol. 11, no. 1, pp. 1–11, Apr. 2021, doi: 10.1038/s41598-021-87116-y.

[5] "1.8.1.4 What information one can obtained with spectral Doppler | 123 Sonography." https://123sonography.com/book/395 (accessed Dec. 11, 2021).

[6] "Echocardiography | NHLBI, NIH." https://www.nhlbi.nih.gov/health-topics/echocardiography (accessed Dec. 16, 2021).

[7] N. S. Anavekar and J. K. Oh, "Doppler echocardiography: A contemporary review," *Journal of Cardiology*, vol. 54, no. 3, pp. 347–358, 2009, doi: 10.1016/j.jjcc.2009.10.001.

[8] J. Mccarthy, "WHAT IS ARTIFICIAL INTELLIGENCE?," 2004, Accessed: Nov. 17, 2021. [Online]. Available: http://www-formal.stanford.edu/jmc/

[9] "The Difference Between AI, Machine Learning, and Deep Learning? | NVIDIA Blog." https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/ (accessed Nov. 17, 2021).

[10] "Frameworks for Approaching the Machine Learning Process - KDnuggets." https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html (accessed Nov. 17, 2021).

[11] "What is Deep Learning? | IBM." https://www.ibm.com/cloud/learn/deep-learning (accessed Nov. 17, 2021).

[12] A. D. Dongare, R. R. Kharde, and A. D. Kachare, "Introduction to Artificial Neural Network," *Certified International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 9001, no. 1, pp. 2277–3754, 2008.

[13] T. Szandała, “Review and comparison of commonly used activation functions for deep neural networks,” *Studies in Computational Intelligence*, vol. 903, pp. 203–224, 2021, doi: 10.1007/978-981-15-5495-7_11.

[14] K. López-Linares Román, M. I. García Ocaña, N. Lete Urzelai, M. Á. González Ballester, and I. Macía Oliver, “Medical Image Segmentation Using Deep Learning,” *Intelligent Systems Reference Library*, vol. 171, pp. 17–31, 2020, doi: 10.1007/978-3-030-32606-7_2.

[15] C. Chen *et al.*, “Deep Learning for Cardiac Image Segmentation: A Review,” *Frontiers in Cardiovascular Medicine*, vol. 7, no. March, 2020, doi: 10.3389/fcvm.2020.00025.

[16] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Accessed: May 19, 2022. [Online]. Available: http://lmb.informatik.uni-freiburg.de/

[17] J. Brownlee, “Better Deep Learning. Train Faster, Reduce Overfitting, and Make Better Predictions,” *Machine Learning Mastery With Python*, vol. 1, no. 2, p. 539, 2018.

[18] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, pp. 177–186, 2010, doi: 10.1007/978-3-7908-2604-3_16.

[19] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[20] Y. H. Zweiri, J. F. Whidborne, and L. D. Seneviratne, “A three-term backpropagation algorithm,” *Neurocomputing*, vol. 50, pp. 305–318, 2003, doi: 10.1016/S0925-2312(02)00569-6.

[21] “Understanding Learning Rates and How It Improves Performance in Deep Learning | by Hafidz Zulkifli | Towards Data Science.” https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 (accessed Dec. 03, 2021).

[22] T. van der Bom, A. C. Zomer, A. H. Zwinderman, F. J. Meijboom, B. J. Bouma, and B. J. M. Mulder, “The changing epidemiology of congenital heart disease,” *Nature Reviews Cardiology*, vol. 8, no. 1, pp. 50–60, 2011, doi: 10.1038/nrcardio.2010.166.

[23] “Data and Statistics on Congenital Heart Defects | CDC.” https://www.cdc.gov/ncbddd/heartdefects/data.html (accessed Dec. 17, 2021).

[24] “Doppler Ultrasound Systems Market – Global Industry Trends and Forecast to 2028 | Data Bridge Market Research.”

https://www.databridgemarketresearch.com/reports/global-doppler-ultrasound-systems-market (accessed Dec. 17, 2021).

[25] E. Sulas, M. Urru, R. Tumbarello, L. Raffo, and D. Pani, “Automatic detection of complete and measurable cardiac cycles in antenatal pulsed-wave Doppler signals,” *Comput Methods Programs Biomed*, vol. 190, Jul. 2020, doi: 10.1016/J.CMPB.2020.105336.

[26] G. Zamzmi, L. Y. Hsu, W. Li, V. Sachdev, and S. Antani, “Harnessing Machine Intelligence in Automatic Echocardiogram Analysis: Current Status, Limitations, and Future Directions,” *IEEE Reviews in Biomedical Engineering*, vol. 14, pp. 181–203, 2021, doi: 10.1109/RBME.2020.2988295.

[27] E. Sulas *et al.*, “Impact of pulsed-wave-Doppler velocity-envelope tracing techniques on classification of complete fetal cardiac cycles,” *PLoS ONE*, vol. 16, no. 4 April, Apr. 2021, doi: 10.1371/journal.pone.0248114.

[28] D. C. Luvizon, H. Tabia, and D. Picard, “Human Pose Regression by Combining Indirect Part Detection and Contextual Information”, Accessed: May 06, 2022. [Online]. Available: https://github.com/dluvizon/pose-regression.

[29] G. Jimenez-Perez, A. Alcaine, and O. Camara, “Delineation of the electrocardiogram with a mixed-quality-annotations dataset using convolutional neural networks,” *Scientific Reports 2021 11:1*, vol. 11, no. 1, pp. 1–11, Jan. 2021, doi: 10.1038/s41598-020-79512-7.

[30] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, Feb. 2015, doi: 10.48550/arxiv.1502.03167.

[31] X. Xia and B. Kulis, “W-Net: A Deep Model for Fully Unsupervised Image Segmentation,” 2017, [Online]. Available: http://arxiv.org/abs/1711.08506

[32] “What is Hausdorff distance? | Statistical Odds & Ends.” https://statisticaloddsandends.wordpress.com/2019/10/02/what-is-hausdorff-distance/ (accessed Mar. 10, 2022).

[33] “Metrics to Evaluate your Semantic Segmentation Model | by Ekin Tiu | Towards Data Science.” https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2 (accessed Mar. 10, 2022).

[34] “Tutorial: Understanding Linear Regression and Regression Error Metrics.” https://www.dataquest.io/blog/understanding-regression-error-metrics/ (accessed May 21, 2022).

[35] “Cross-Correlation Definition.” https://www.investopedia.com/terms/c/crosscorrelation.asp (accessed May 21, 2022).

[36] “Hausdorff Distance — skimage v0.19.2 docs.” https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_hausdorff_distance.html (accessed Apr. 06, 2022).

[37] “MSELoss — PyTorch 1.11.0 documentation.” https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html (accessed Apr. 06, 2022).

[38] “sklearn.metrics.mean_absolute_percentage_error — scikit-learn 1.1.1 documentation.” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_percentage_error.html (accessed May 20, 2022).

[39] H. Wang *et al.*, “Mixed Transformer U-Net For Medical Image Segmentation,” Nov. 2021, doi: 10.48550/arxiv.2111.04734.

[40] C. Tissot, Y. Singh, and N. Sekarski, “Echocardiographic evaluation of ventricular function-for the neonatologist and pediatric intensivist,” *Frontiers in Pediatrics*, vol. 6, 2018, doi: 10.3389/FPED.2018.00079.

[41] M. A. Navarro, M. Kim, and E. E. Salcedo, “Real-Time 3D Echocardiography in Percutaneous Balloon Mitral Valvuloplasty,” *Hot Topics in Echocardiography*, Nov. 2013, doi: 10.5772/56433.

[42] K. A. Carpenter, D. S. Cohen, J. T. Jarrell, and X. Huang, “Deep learning and virtual drug screening,” *Future Medicinal Chemistry*, vol. 10, no. 21, pp. 2557–2567, Nov. 2018, doi: 10.4155/FMC-2018-0314.