UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Time-Varying Topological Descriptors for Cardiac Disease Diagnosis

---

Author:
Jon FERRERAS

Supervisors:
Dr. Carles CASACUBERTA
Dr. Laura IGUAL

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

January 17, 2025

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Time-Varying Topological Descriptors for Cardiac Disease Diagnosis

by Jon FERRERAS

Cardiac diseases are among the most common illnesses in the world, and data scientists have created a wide range of tools to contribute to their detection and diagnosis. In particular, topological data analysis has been used to work with medical imaging and specifically with cardiac magnetic resonance images.

This project introduces the use of time-varying topological descriptors along a cardiac cycle and applies them for disease diagnosis. The methods used aim to develop the relationship between topological data analysis and temporal data. We also intend to contribute to the simplification, interpretability and improvement of a computational approach to cardiac disease diagnosis, which usually involves costly calculations of radiomics or potential black boxes.

Acknowledgements

I want to express my gratitude to everyone that contributed to the development and completion of this work.

First and foremost, I would like to thank my tutors Dr. Carles Casacuberta and Dr. Laura Igual for their excellent advice and guidance.

I would also like to thank Dr. Polyxeni Gkontra, Dr. Víctor Campello and Grzegorz Skorupko from the BCN-AIM Artificial Intelligence in Medicine Lab for their advice, support and expertise, and for providing tools that were crucial for the completion of this project.

I would like to thank Aina Ferrà as well, for her guidance in the practical application of topological data analysis, and Marina Anguas, because without her project this work would not have existed.

Finally, I would like to thank my family and my friends for their support throughout this project, particularly Xisco, Sergio, Ana and Jokin, who have made this last stretch of the master a lot more enjoyable.

Contents

Chapter 1

Introduction

1.1 Machine learning in medicine

The astonishing development of machine learning in recent years has reached all fields in science, economy and many other disciplines, and medicine is not an exception. Machine learning and deep learning models have been successfully used for a wide range of problems in the field (there is a good summary in Shehab et al., 2022). Due to the nature of the medical sciences, though, there are some specific conditions and requirements that need to be taken into account for machine learning models before they can be implemented in clinical practice.

The first important condition that data scientists in the field face is difficulties with accessing data. Medical data is much more delicate, hard to obtain and tough to label than data in other fields. For medical data to be made public, it first needs to go through a privatization filter, and even when it is accessible, it usually needs cleaning to be functional. Moreover, annotating data requires work from experts, so the amount of available data is usually small. An overview of this long pipeline of medical data preparation can be found in Diaz et al., 2021. Private datasets with better conditions usually require some layers of bureaucracy to access. This situation greatly conditions the options for data scientists, who usually have to resort to techniques that learn from limited data or work with non-expert annotations, and ask for feedback after development.

Medical data also tends to suffer from imbalance, particularly for rare diseases where the possible pool of cases is very reduced. This can then turn into a lack of generalizability due to overfitting in the training set. In this regard, the diversity of medical data (different scanner models, different criteria of annotation between countries or institutions, unintended particularities of a certain dataset...) can generate biased models that work very well for the dataset they were trained on but quickly deteriorate for other cases (this phenomenon is known as domain shift).

Finally, machine learning still needs to gain the trust of clinical experts. While the acceptance of artificial intelligence in medicine is growing, experts have basic and reasonable concerns, mainly regarding the process by which the models take decisions. Neural networks, for example, tend to achieve the best performance in terms of accuracy, but they are considered black boxes because they do not provide explanations for model outcomes, and hence cannot be trusted, because mistakes in medicine are much more serious than in other fields. Moreover, incorrect artificial intelligence results affect the performance of medical experts (Scheikl, Grüber, Glatz-Krieger, et al., 2023). In this context, the need for explainable artificial intelligence becomes more prominent, as well as studying the trade-off between accuracy and explainability and designing additional models that can explain the basic models (see Salih et al., 2023 for a study about explainability in cardiac imaging).

1.2 Cardiac imaging

Cardiovascular diseases are the leading cause of death in the world (Vaduganathan et al., 2022), even though scientific investigation in this field has greatly evolved in the past decades. Any progress in improving cardiac disease diagnosis and detection is, hence, potentially saving many lives around the world. One of the most remarkable advancements in this field is the ability to produce cardiac images to assess the heart's function and disease.

One way to acquire these images is magnetic resonance imaging (MRI), which is a non-invasive technique without radiation that produces images with a good temporal resolution (Campello, 2023). MRI generates two-dimensional slices that are stacked to form a 3D volume, and it yields images at different time points to obtain information about the whole cardiac cycle. Additionally, four different views are generated, corresponding to the heart's planes: the short axis, which consists of slices that are orthogonal to the heart's longitudinal direction, and three different views for the long axis, where the different chambers of the heart are visualized in the heart's longitudinal direction. The short axis view and one of the long axis views are shown in Figure 1.1.

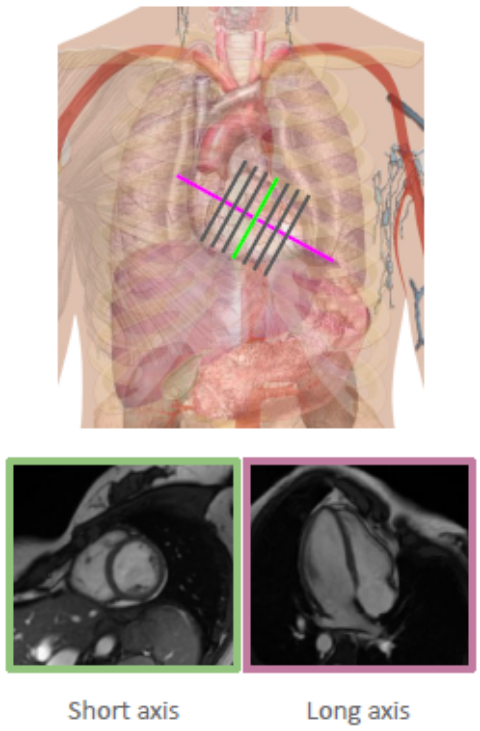


FIGURE 1.1: Short axis and one of the long axis views of a cardiac MRI. The image is taken from Campello, 2023.

When the quality of the images is acceptable, the experts analyse them visually and quantitatively. This, as in many other fields in medicine, implies a long annotation process consisting of detecting specific time points, such as the end-diastole frame, which is the frame corresponding to the maximal cavity area achieved at the expansion phase of the heart, and the end-systole frame, that is, the frame corresponding to the minimal cavity area achieved at the contraction phase of the heart (Darvishi et al., 2013); and also delineating important areas of the heart, such as

the left ventricle, the left ventricular myocardium and the right ventricle. This last process is also known as segmentation.

As expected, machine learning has also made its way into cardiac imaging, with the goal of simplifying the tedious process of acquisition, annotation and diagnosis. For example, convolutional neural networks have been used to infer a full-coverage stack from some slices (Upendra, Simon, and Linte, 2021) to avoid the many long breath-holds that patients are usually required to do for MRIs, to automatically detect the end-diastole and end-systole (Pérez-Pelegrí et al., 2022) or to perform the segmentation (Tran, 2017). For diagnosis, a common approach is the extraction of radiomics from the regions of interest with posterior feature selection and evaluation of the model, which have proved to be useful to diagnose certain diseases, like myocardial infarction and hypertrophic cardiomyopathy (Martin-Isla et al., 2020; Izquierdo et al., 2021). Radiomics are features extracted from images that can describe a wide range of properties: shape, intensity distribution, texture... A complete list of common radiomics can be found in the PyRadiomics library (`https://pyradiomics.readthedocs.io/en/latest/features.html`).

Recently, topological data analysis (TDA) has also been used as a way to tackle medical images (Singh, Farrelly, Hathaway, et al., 2023), presenting an alternative or add-on to the often costly to extract and redundant radiomics, while also being a quite interpretable model that can solve dimensionality issues and not requiring as many samples as other machine learning models or neural networks to perform correctly.

1.3 Problem description and objectives

This project was devised with the purpose of expanding on a Bachelor's thesis (Anguas, 2023). In this thesis, TDA was used to extract features from the end-diastole and end-systole frames of a dataset (see Chapter 2) and used them for disease classification, proving that TDA achieves comparable results to radiomics and that a combination of both achieves better results, that is, TDA enhances radiomics.

However, the above project did not use all the information available, but only used the end-diastole and end-systole frames, in spite of the fact that the dataset provides all the frames of the cardiac cycle. Our hypothesis is that introducing temporal information improves the performance of machine learning classifiers, i.e., that the temporal evolution of the images contains information that is lost when only using the first and last frame of the cycle.

Introducing every frame, though, multiplies the amount of information that the machine learning classifiers need to analyse, because it generates several time series. The main goal of the present project is, hence, finding out how to take advantage of the temporal information available in a cardiac cycle while correctly managing the dimensionality, explainability and simplicity of the problem.

The code used for this work can be found in the following GitHub repository: `https://github.com/jonferreras/MasterThesis`.

Chapter 2

Dataset

In this chapter we go over the data used for this project, outlining its characteristics to understand why TDA is able to extract information from it.

2.1 Overview

The Automated Cardiac Diagnosis Challenge Dataset (ACDC dataset from now on, (Bernard et al., 2018)) is a dataset consisting of 150 three-dimensional short axis cardiac MRI recordings with reference measurements and classification from medical experts. It was created from real clinical exams carried out at the University Hospital of Dijon. The 150 patients are divided into 5 subgroups: 1 for healthy patients and 4 for sick patients with different diseases:

Subgroup	Label	No. of patients
Normal (healthy)	NOR	30
Previous myocardial infarction	MINF	30
Dilated cardiomyopathy	DCM	30
Hypertrophic cardiomyopathy	HCM	30
Abnormal right ventricle	RV	30

TABLE 2.1: Distribution of patients in the ACDC dataset

In addition to the recordings, the dataset provides the weight, height, and the end-diastole and end-systole phase instants (from now on, diastolic and systolic) for each patient. There are also segmentation masks for these instants, which allow the division of the heart MRIs into three regions of interest: left ventricle, right ventricle and myocardium.

The challenge was created for two purposes: checking performance of segmentation methods and checking performance of classification of the five subgroups. The dataset is divided into 100 train subjects (20 from each subgroup) and 50 test subjects (10 from each subgroup), and we use this division for our experiments as well.

2.2 Use of the ACDC dataset for our project

This data is interesting for our project for many reasons. For starters, it is one of the largest fully-annotated public MRI cardiac datasets ever assembled. Moreover, the pathologies of the patients are related to the shape and texture of the three parts of the provided segmentations (left ventricle, right ventricle and myocardium), which allows us to apply TDA methods that can detect differences in the shape or texture and therewith classify. We can translate the pathologies into understandable differences as follows:

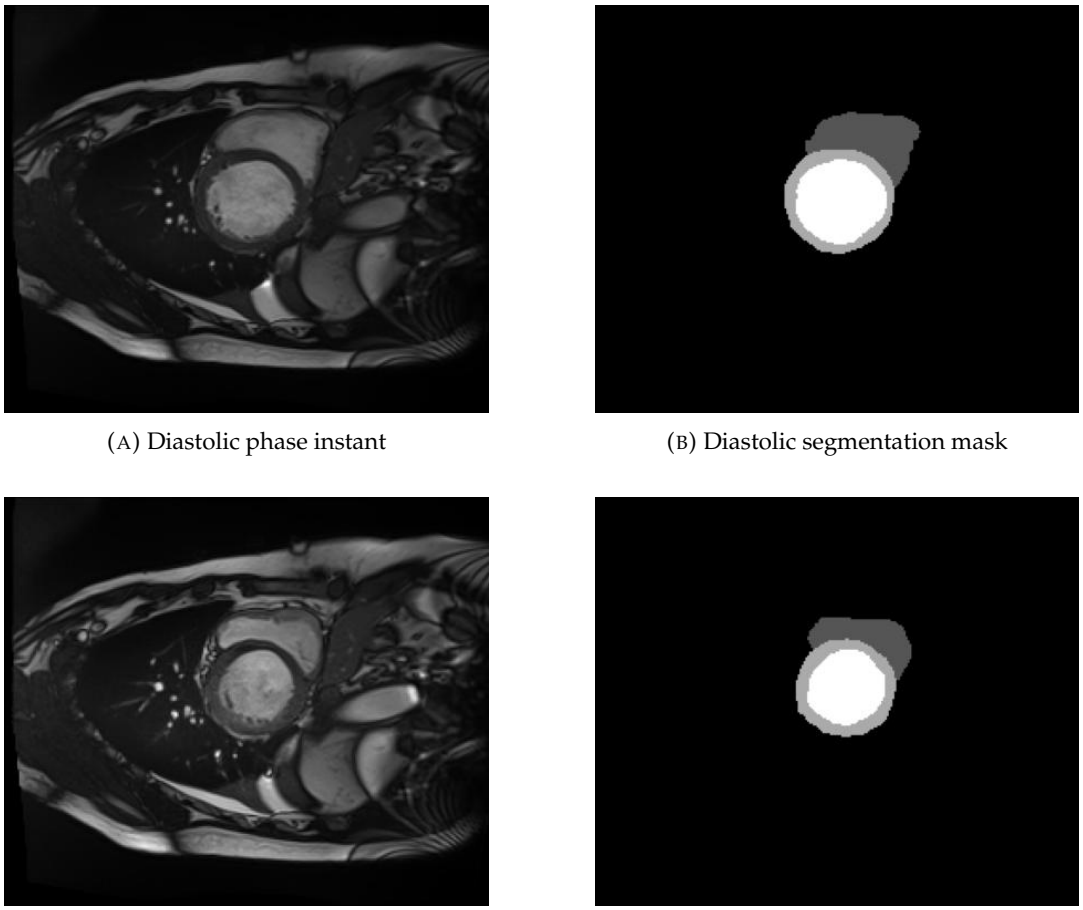
- Patients with previous myocardial infarction could present differences in the left ventricle and myocardium.
- Patients with dilated cardiomyopathy present differences in the left ventricle.
- Patients with hypertrophic cardiomyopathy could present differences in the left ventricle and myocardium.
- Patients with abnormal right ventricle present differences in the right ventricle.

Aside from these differences directly related to the pathology, some particular patients could present other distinctions. For example, some patients with dilated cardiomyopathy (of the left ventricle, which is the case in the dataset) can also present differences in the right ventricle because their right ventricle adjusts to the dilation of the left ventricle.

Furthermore, the dataset consists of temporal data (recordings of a cycle), so it meets all the conditions to test our hypotheses. An important detail is that the recordings have a different number of frames for each patient, and the amount of frames between the diastolic and systolic instants also varies greatly between patients.

Figure 2.1 shows MRIs of the diastolic and systolic phase instants of the first patient of the dataset, together with their corresponding segmentation masks. In these masks, the white colour corresponds to the left ventricle, the light gray to the myocardium, and the dark gray to the right ventricle. Since the images are 3D, for visualization purposes here we are just showing the 2D images corresponding to a slice.

As stated above, one key part of this dataset for our project is that it does not only provide frames for the diastolic and systolic phase instants, but also all the frames in between. In Figure 2.2 we can observe slices of the whole cycle for the same patient as in Figure 2.1, which consists of 12 frames in this case. The dataset does not provide segmentation masks for each frame.



(A) Diastolic phase instant

(B) Diastolic segmentation mask

(C) Systolic phase instant

(D) Systolic segmentation mask

FIGURE 2.1: Slices of the diastolic and systolic MRIs and their corresponding segmentation masks

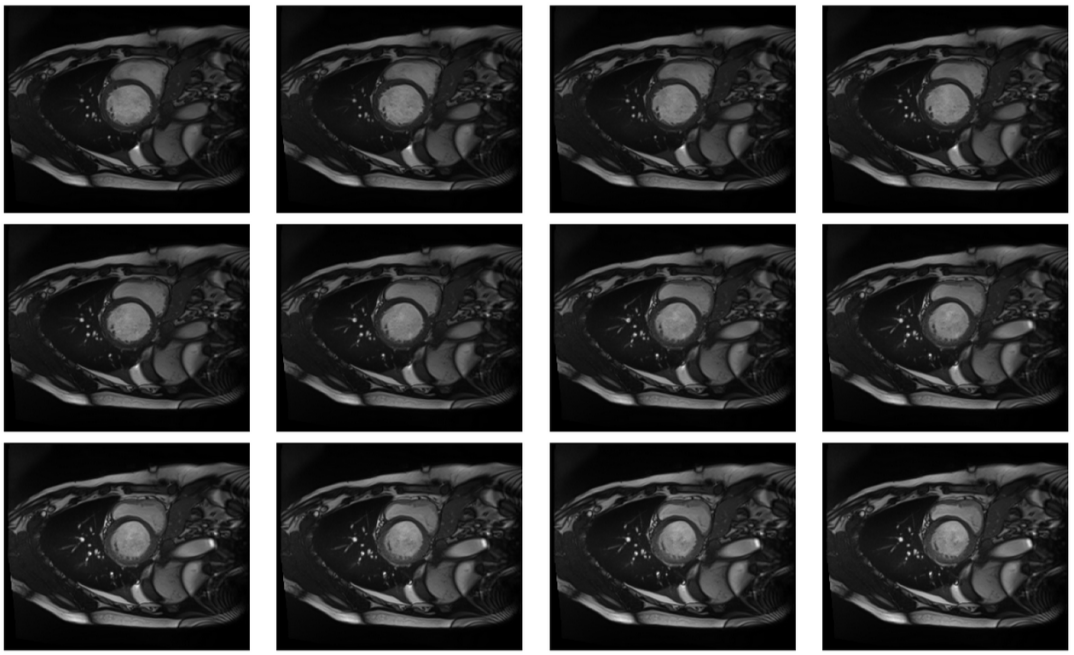


FIGURE 2.2: Slices of all the frames of a cardiac cycle ordered from left to right and from top to bottom

Chapter 3

Image processing

The ACDC dataset contains a 3D cardiac MRI recording for every patient, as well as the diastolic and systolic phase instants. The processing pipeline is the following:

1. Extract the frames between the diastolic and systolic instants (both included). This ensures that we are working with a single cardiac cycle.
2. Apply segmentation masks to the frames of the cycle.
3. Apply additional processing to the segmented images.

The first step is performed as follows: for each patient's cardiac MRI recording, we read their diastolic and systolic phase instants (ACDC provides this information) and we take the frames in between, including both ends.

3.1 Segmentation

As we said in Chapter 2, the ACDC dataset does not provide a segmentation mask for every frame; there are only segmentation masks for the diastolic and systolic frames. This led us to evaluate different options.

- We developed a basic segmentation model for the 2D slices of the dataset, consisting essentially of a convolutional neural network with a weighted loss that forced the network to generate white pixels, as initially it was generating masks consisting of all black pixels. We obtained some useful results and were considering different ideas to improve it further. Ultimately we discarded the use of a 2D approach because we applied a 3D model.
- The BCN-AIM Artificial Intelligence in Medicine Lab has developed a Virtual Research Environment (VRE) for the euCanSHare project (http://www.eucanshare.eu/). The VRE contains many tools for heart research, and they had developed a cardiac image segmentation tool trained precisely with the ACDC dataset. However, the tool was no longer available.

In the end, we used a nnUNet (Isensee et al., 2021) model trained by the BCN-AIM Lab with the M&Ms1 and M&Ms2 datasets (https://www.ub.edu/mnms/ and https://www.ub.edu/mnms-2/), which were precisely designed for cardiac segmentation. We applied this model to every frame (1600+ images in 3D) of the cardiac cycles (including the diastolic and systolic ones, for consistence) to obtain 3D segmentation masks. In Figure 3.1 we can see the masks that the model generates for the frames in Figure 2.2.

The main problem with this generated segmentation is that it is not checked by a medical expert, unlike the ones in ACDC. This led us to consider two different approaches: one with the nnUNet model segmentation, which we call **NETSEG**, and

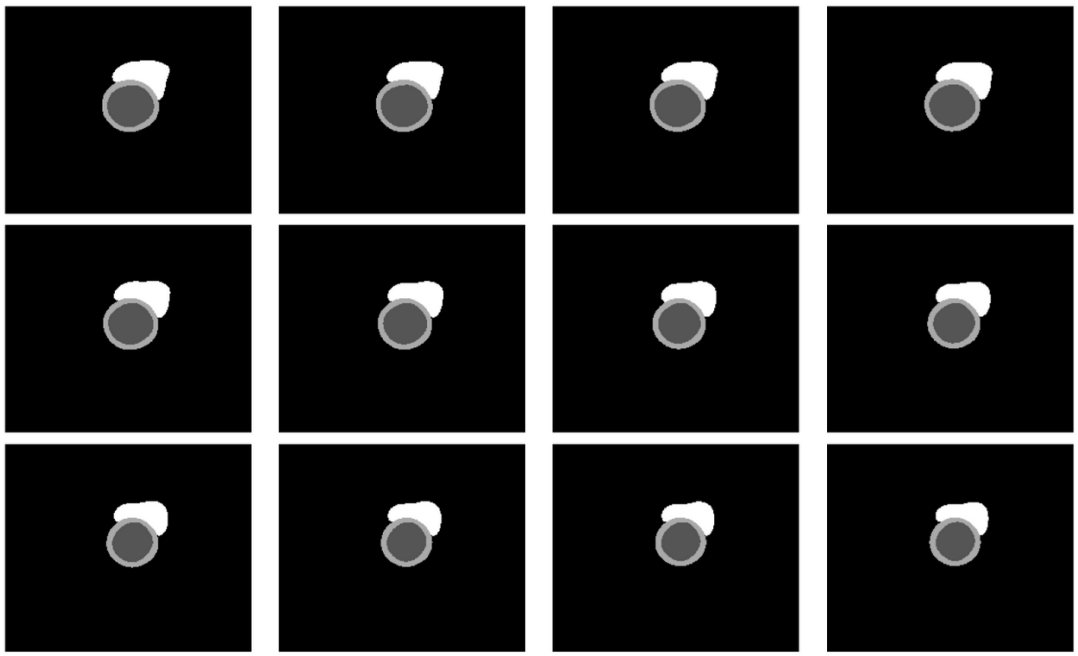


FIGURE 3.1: Slices of some masks generated by the nnUNet model

one with an approximate, rough segmentation, using only the elements in ACDC called **ROUSEG**. It is approximate because we considered that the best option within the ACDC dataset was to use the systolic phase mask to segment all the frames, that is, extending the segmentation of the final instant of the cycle to every image in it. We also considered using the diastolic mask, but this could lead to parts of the image outside of the regions of interest being considered for classification. Using the systolic mask made us lose some information about the shape, but it kept most of the image in the regions of interest. Since almost all the methodology is the same for both approaches, from now on everything that we say is applied to both, unless otherwise specified.

The segmentation mask is applied to an image using the Hadamard product:

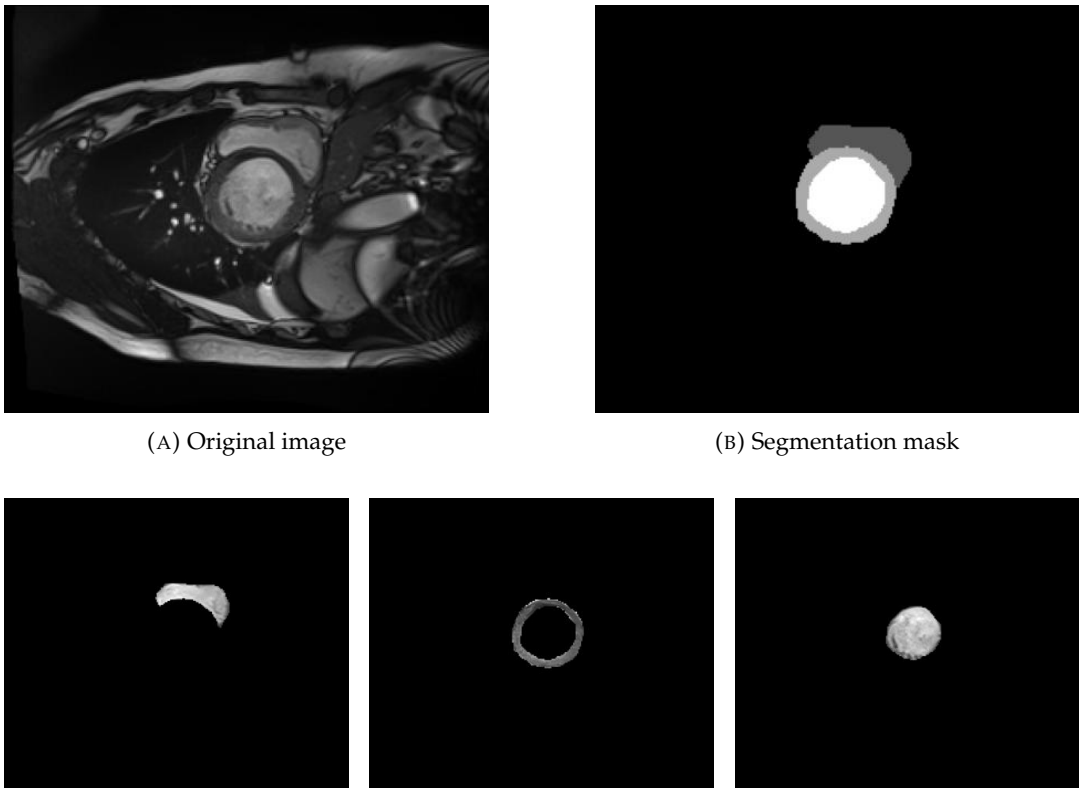
Definition 3.1.1. For two matrices A and B of the same dimension $m \times n$, with elements a_{ij} and b_{ij} respectively, the *Hadamard product* is a matrix C of dimension $m \times n$ with elements c_{ij} such that

$$c_{ij} = a_{ij} \cdot b_{ij}$$

for all $1 \leq i \leq m$, $1 \leq j \leq n$. We write

$$C = A \circ B.$$

Hence, the Hadamard product applied to 3D images consists of multiplying each voxel of an image with the voxel on the same position. In particular, multiplying a voxel by a black voxel generates a black voxel in the new image, as seen in Figures 3.2c, 3.2d and 3.2e. Notice that performing the segmentation yields three images from each frame: one for the left ventricle, one for the myocardium and one for the right ventricle. The following steps in this chapter and the next one are applied to each of these images.



(A) Original image (B) Segmentation mask

(C) Right ventricle (D) Myocardium (E) Left ventricle

FIGURE 3.2: Example of application of a segmentation mask

3.2 Additional processing

Since the ACDC images are quite diverse, as is common in medical imaging due to the variabilities in the acquisition process, we applied a standard normalization between 0 and 1: if the image has M and m as maximum and minimum values respectively, then, for each value v of a voxel, the normalized value v' is

$$v' = \frac{v - m}{M - m}.$$

The images in the ACDC dataset have very different sizes in the three axes, ranging from 154 to 256 in the x and y axes and from 6 to 21 in the z axis, so we also considered padding or cropping the images to a standard size, but in the end we considered that it was not necessary. For TDA, what is important is the generated cubical complex (see Chapter 4), not the size of the image, and adding or cropping black pixels makes no difference. Resizing the segmented regions of interests was outside of the question, because the size of a region relative to other patients is a significant metric for the dataset (patients with abnormal right ventricle, for example, present bigger right ventricles than the other patients).

Chapter 4

Topological data analysis

Topological data analysis (TDA) is a topology-based approach to the study of data. It is relatively recent, having emerged around the year 2000, and it is based on combining algebraic topology concepts with statistical and algorithmic methods to describe the "shape of data", that is, the topological and geometrical structures underlying data. It detects connectivity of the data (connected components), loops and cavities of higher dimension and clusters, among other things.

TDA has found many applications in machine learning, and it has been particularly useful in studies of biomedical sciences. This is due to many factors, but we would like to highlight two: firstly, the shape (and texture) of the data is generally more relevant than in other fields, mainly in medical image; and secondly, TDA is easy to explain, which is a key factor in medical use of machine learning.

This chapter is divided into a theoretical introduction to TDA and its application to our images. The first part is mainly based on Chazal and Michel, 2021 and Kaczynski, Mischaikow, and Mrozek, 2004.

4.1 Background

4.1.1 Complexes

Simplicial complexes

The first thing that TDA needs is to convert the data to a topological space so that it can extract topological features. Simplicial complexes, which are higher-dimensional analogues of graphs, are the basic structures we can build from point clouds. Since point clouds and images are finite, every complex we discuss in this work is finite.

Definition 4.1.1. Given a set $P = \{p_0, \dots, p_n\}$ of $n+1$ points in $\mathbb{R}^d$, a point $\sum_{i=0}^n \lambda_i p_i$ is an *affine combination* of P if $\sum_{i=0}^n \lambda_i = 1$. If $\lambda_i > 0$ for all i, then $\sum_{i=0}^n \lambda_i p_i$ is called a *convex combination*. The *convex hull* of a set is the collection of all its convex combinations.

Definition 4.1.2. An *n-simplex* in $\mathbb{R}^d$, where $0 \leq n \leq d$, is the convex hull of a collection of $n+1$ points $p_0, \dots, p_n$ such that the vectors $p_0p_1, \dots, p_0p_n$ are linearly independent. We denote it as $\Delta(p_0, \dots, p_n)$, and we say that it has *dimension n*. Every subset $\{p_{i_0}, \dots, p_{i_k}\} \subseteq \{p_0, \dots, p_n\}$ with $0 \leq k \leq n$ spans a k-simplex, which is called a *k-face* of $\Delta(p_0, \dots, p_n)$.

Definition 4.1.3. The *standard n-simplex* Δ^n is the convex hull of the coordinate unit points $e_0, \dots, e_n$ in $\mathbb{R}^{n+1}$ where $e_i = (0, \dots, 1, \dots 0)$ with 1 is in the i-th position. Hence,

$$\Delta^n = \Delta(e_0, \dots, e_n) = \{(x_0, \dots, x_n) \in \mathbb{R}^{n+1} \mid x_0 + \cdots + x_n = 1,\ x_j \geq 0 \text{ for all } j\}.$$

Definition 4.1.4. A *geometric simplicial complex* in $\mathbb{R}^d$ is a set X of simplices in $\mathbb{R}^d$ such that:

1. Every k-face of a simplex of X is a simplex of X.
2. Any two simplices of X are either disjoint or intersect along one common face.

The *dimension* of a geometric simplicial complex is the maximum of the dimensions of its simplices.

With this definition, we can understand why these are higher-dimensional analogues of graphs; they encode relationships between points but are not confined to the 1-dimensional structure of graphs, since higher-dimensional faces also contain information.

Notice that geometric simplicial complexes are fully characterized by their vertices and a combinatorial description of its simplices (such that they follow the two rules above). Geometric simplicial complexes are, hence, combinatorial objects, well-suited for computations. But we need more: to make use of algebraic topology, we need to be able to consider them as topological spaces.

Definition 4.1.5. Every geometric simplicial complex X has an *underlying* topological space $|X|$, which is the union of all the simplices in X, endowed with the Euclidean topology. $|X|$ is called a *polyhedron* and X is called a *triangulation* of $|X|$.

Definition 4.1.6. An *abstract simplicial complex* with vertex set $V = \{v_i\}_{i\in I}$ is a collection K of nonempty finite subsets $\{v_{i_0}, \dots, v_{i_k}\} \subseteq V$ such that:

1. $\{v\} \in K$ for all $v \in V$.
2. If $F \in K$ and $G \subseteq F$ is nonempty, then $G \in K$.

Notice that the second condition here is analogous to the first one in the definition of a geometric simplicial complex.

Definition 4.1.7. The elements of V are called *vertices* of K and the elements of K are called *faces* of K (a face $\{v_{i_0}, \dots, v_{i_k}\}$ of cardinality $k+1$ is called a *k-face*, or an *edge* if it is a 1-face). With this definition, $\{v\}$ is a 0-face of K for every $v \in V$. The collection of all k-faces of K for all $0 \leq k \leq m$ is an abstract simplicial complex in itself, called the *m-skeleton* of K.

The relation between geometric and abstract simplicial complex is then as follows: if K is an abstract simplicial complex and we define an order on its vertices, then we can assign each k-face to a k-simplex in $\mathbb{R}^d$, formed by the coordinate unit points corresponding to the vertices of the face, and let X_K be the set of all such simplices associated with the faces of K. The set X_K is called the *geometric realization* of K and this allows us to define an underlying topological space for K as $|K| := |X_K|$. Conversely, if X is a geometric complex, we take V to be the set of 0-faces of X and then X yields an abstract complex K_X with vertex set V and faces given by the simplices of X. In this situation, $|X|$ is homeomorphic to $|K_X|$.

In the end, we get that simplicial complexes can be considered as both combinatorial objects and topological spaces, which is what makes them the basic constructs for TDA.

There are two main ways of building a simplicial complex from a *point cloud*, which we understand as an unordered finite collection of points $X = \{x_i\}_{i\in I}$ in $\mathbb{R}^d$, for $d \geq 2$.

Definition 4.1.8. For each real number $\epsilon > 0$, the *Čech complex* $C_\epsilon(X)$ of X is the abstract simplicial complex with vertex set X whose k-faces are collections of points $\{x_{i_0}, \dots, x_{i_k}\}$ such that the closed balls $\overline{B}_{\epsilon/2}(x_{i_0}), \cdots, \overline{B}_{\epsilon/2}(x_{i_k})$ have at least one common point.

An intuitive way of understanding Čech complexes is imagining balls of diameter ϵ around each of the points, and edges and higher dimensional faces generating where the balls meet[1].

Definition 4.1.9. For each real number $\epsilon > 0$, the *Vietoris-Rips complex* $R_\epsilon(X)$ of X is the abstract simplicial complex with vertex set X whose k-faces are collections of points $\{x_{i_0}, \dots, x_{i_k}\}$ such that $d(x_{i_r}, x_{i_s}) \leq \epsilon$ for all $r, s \in \{0, \dots, k\}$.

Remark 4.1.10. The Čech and Vietoris-Rips complexes are not equivalent in general, but they are related. For every X and every ϵ, there are inclusions $C_\epsilon(X) \subseteq R_\epsilon(X)$ and $R_\epsilon(X) \subseteq C_{\sqrt{2}\epsilon}(X)$.

Cubical complexes

Cubical complexes are analogous to simplicial complexes, but more restrictive, because they can only be built from points that are placed on a grid. As we will see, these will be very useful for 2D and 3D images, which are essentially grids of pixels or voxels.

Definition 4.1.11. An *elementary interval* I is a closed interval of the form $I = [a, a+1]$ (*non-degenerate)* or $I = [a, a]$ (*degenerate*) for some $a \in \mathbb{Z}$. We write $[a] = [a, a]$ for the intervals with just one point.

Definition 4.1.12. An *elementary cube* Q is a finite product of elementary intervals:

$$Q = I_1 \times I_2 \times \cdots \times I_d \subset \mathbb{R}^d$$

where each I_i is an elementary interval. The *dimension* of Q is the number of non-degenerate intervals in the product.

Definition 4.1.13. A *cubical complex* C is a collection of elementary cubes such that if $Q \in C$ and $Q' \subseteq Q$ then $Q' \in C$.

This definition is clearly analogous to that of simplicial complexes; elementary cubes are to cubical complexes what simplices are to simplicial complexes. The vertices are the endpoints, the edges are the intervals, 2-faces correspond to squares, and so on. In fact, if $Q' \subseteq Q$ we say that Q' is a *face* of Q. As one can imagine, the faces correspond to the "boundaries" of the cubes. For example, for the elementary cube $[1,2] \times [1,2] \subset \mathbb{R}^2$ (a square), the faces of dimension 1 are $[1] \times [1,2]$, $[2] \times [1,2]$ (the vertical sides), $[1,2] \times [1]$ and $[1,2] \times [2]$ (the horizontal sides).

Remark 4.1.14. Cubes in a cubical complex are placed in a grid, and hence they are always either disjoint or intersecting along a common face; that is why we do not need to add anything like the second condition in Definition 4.1.4.

Now that we have the tools to build topological objects from data, we will see how we can obtain their topological features.

[1]There is a good visualization of this for a 2D point cloud in https://sauln.github.io/blog/nerve-playground/.

4.1.2 Persistent homology

As we have stated, TDA aims to capture the "shape" of data. To do this, it builds complexes from data (we can say "on top of data" as well), but usually it is more interesting to build a nested family of simplicial complexes that reflects the structure of the data at different scales. These families are called filtrations, and we describe below how to build them. By building filtrations, we can study changes in the topological features along the family, and for this we use persistent homology.

Filtrations

Definition 4.1.15. A *filtration* of a topological space M is a nested family of subspaces $(M_i)_{i\in I}$ where $I \subseteq \mathbb{R}$ such that for any $i, j \in I$, if $i \leq j$ then $M_i \subseteq M_j$, and $M = \cup_{i\in I} M_i$. In particular, a filtration of a simplicial complex K is a collection $\{K_i\}$ of subcomplexes of K such that $K_i \subseteq K_j$ if $i \leq j$ and $K = \cup_{i\in I} K_i$.

Example 4.1.16. If $K^{(m)}$ is the m-skeleton of a simplicial complex K, then $\{K^{(m)}\}_{0\leq m\leq n}$ is a filtration, where n is the dimension of K.

Definition 4.1.17. A *filtering function* of a simplicial complex K is a function $f : K \to \mathbb{R}$ such that $f(\tau) \leq f(\sigma)$ when $\tau \subseteq \sigma$, where τ and σ are faces. When K is finite, a filtering function f on K determines a finite filtration

$$K_0 \subseteq K_1 \subseteq \cdots \subseteq K_{m-1} \subseteq K_m = K$$

where f takes the values $\{\epsilon_0, \ldots, \epsilon_m\}$ with $\epsilon_0 < \cdots < \epsilon_m$ and

$$K_i = \{\sigma \in K \,|\, f(\sigma) \leq \epsilon_i\} = f^{-1}((-\infty, \epsilon_i])$$

for every $i \in \{0, \ldots, m\}$. The complexes K_i are called *sublevel complexes* of f.

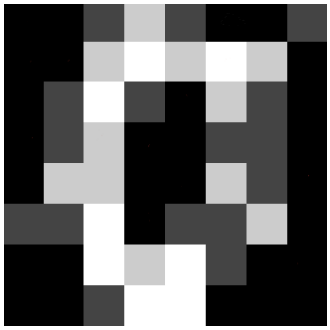
Looking at the definition of filtration, we see that the index set I could be infinite, which would be problematic from a computational point of view. But since we will always be working with finite data, we can use filtering functions to build filtrations. The most common approach for a point cloud X is to use the structure of Vietoris-Rips complexes as follows: take the set $\{\epsilon_0, \ldots, \epsilon_m\}$ of values of ϵ for which a new face appears precisely at $R_\epsilon(X)$, and the filtering function sends every face of $R_D(X)$ to the ϵ value at which the face is born, where D is the diameter of the point cloud (the longest distance between two points). A similar approach works for Čech complexes. This generates a family of complexes which contain all the topological information of the point cloud, because even with different values of ϵ, if there are no new faces, the complexes are topologically the same.

Regarding cubical complexes, we can build them from a d-dimensional image in two main ways (Solomon and Bendich, 2024):

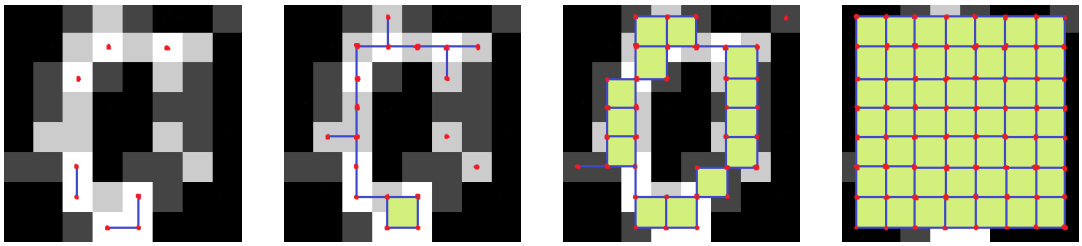
- A *lower-star filtration* starts with considering the voxels as vertices and faces as coming from voxel adjacencies; that is, a pair of adjacent pixels form an edge, squares come from four voxels in a square position, and so on. Now, given a real-valued function defined on the vertices (for example, assigning a number between 0 and 255 to each voxel of a gray-scale image depending on its brightness), we extend it by assigning to each cube the value of its maximum vertex. In this way, the faces "appear" in a subcomplex only when all of its vertices are in it.

- An *upper-star filtration* considers the voxels as d-dimensional cubes, and the lower-dimensional cubes are the faces of these voxels. In this case, the real-valued function is defined on the voxels and the vertices appear in the adjacencies between voxels, and they are assigned the minimum of the values of the surrounding voxels.

In this project we use lower-star filtration, as it is the standard way that GUDHI (the TDA library for Python that we use, `https://gudhi.inria.fr/`) builds cubical complexes. In Figure 4.1 we can see an example of a cubical complex filtration for a 2D gray-scale image.



(A) Original gray-scale image



(B) K0 (C) K1 (D) K2 (E) K3

FIGURE 4.1: A cubical complex filtration of a 2D image, where white pixels have value 0, light-gray pixels have value 1, dark gray pixels have value 2 and black pixels have value 3. Vertices are colored in red, edges in blue and squares in green.

Homology

Homology is one of the main concepts in algebraic topology, as it is an algebraic method to extract topological features of topological spaces (such as simplicial complexes). There are many books to consult for a complete explanation of homology, such as Hatcher, 2002.

For any dimension $k \geq 1$, a topological space can have k-dimensional cavities. One-dimensional cavities are called *loops*, and we can see one in Figure 4.1d around the black pixels in the middle. The most obvious example is the space in the middle of a circle. Two-dimensional cavities are "volume" cavities, such as the inside of an empty tetrahedron. For a topological space, k-dimensional cavities are represented by a vector space H_k the dimension of which is the number of these cavities. Moreover, there is a vector space H_0 which represents the set of path-connected components of the space.

For a complex K, the vector spaces H_k, which are called *homology vector spaces*, are calculated with *chains* and *boundaries* of simplices.

Definition 4.1.18. If $\{\sigma_1, \dots, \sigma_r\}$ is the set of k-simplices of K, then a *k-chain* is

$$c = \sum_{i=1}^{p} \mu_i \sigma_i$$

where the coefficients μ_i can be taken in any field. We take coefficients in $\mathbb{Z}_2$, since it is the standard choice for TDA libraries such as GUDHI.

The set of all k-chains is called the *space of k-chains on K* and we denote it $C_k(K)$.

Definition 4.1.19. The *boundary* of a k-simplex σ, which we denote using its vertices as $\sigma = [v_0, \dots, v_k]$, is the $(k-1)-$chain

$$\partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i [v_0, \dots, \hat{v_i}, \dots, v_k]$$

where $[v_0, \dots, \hat{v_i}, \dots, v_k]$ is the $(k-1)$-simplex spanned by all the vertices in the list except v_i.

The boundary is hence the chain generated by the alternated sum of every $(k-1)$-face of the k-simplex. This boundary operation can be extended to be a linear map from $C_k(K)$ to $C_{k-1}(K)$, which has a kernel $Z_k(K) = \{c \in C_k(K) \,|\, \partial_k(c) = 0\}$ called the *space of k-cycles of K*, and there is also an operator ∂_{k+1} from $C_{k+1}(K)$ to $C_k(K)$ which has image $B_k(K) = \{c \in C_k(K) \,|\, \exists c' \in C_{k+1}(K),\ \partial_{k+1}(c') = c\}$ called the *space of k-boundaries of K*. One can easily prove that $\partial_{k+1} \circ \partial_k \equiv 0$ for every k, so any boundary is a cycle, which means that $B_k(K) \subseteq Z_k(K) \subseteq C_k(K)$.

Definition 4.1.20. The *k-th (simplicial) homology vector space of K* is the quotient vector space

$$H_k(K) = Z_k(K)/B_k(K).$$

An element of this space is referred as a *homology class*.

In Figure 4.1d we can see an example of a cycle that is not a boundary, around the black pixels. This generates a cavity of dimension one.

Due to the combinatorial nature of complexes, the computation of their cycles and boundaries is neither difficult nor costly, which is why they are convenient structures to build on top of data.

Persistence diagrams

Now that we have homology at our disposal, we can apply it to families of complexes built on top of data. First, we need to understand how homology behaves through filtrations.

Given a filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_{m-1} \subseteq K_m = K$ of a complex K, for $i \leq j$ and for every k, the inclusion $K_i \hookrightarrow K_j$ induces a linear map

$$\varphi_k^{i,j} : H_k(K_i) \to H_k(K_j)$$

between the homology vector spaces of the subcomplexes.

Definition 4.1.21. A homology class $\alpha \in H_k(K_j)$ is *born* at K_j if it does not belong to the image of $\varphi_k^{i,j}$ for any $i < j$, and a homology class $\alpha \in H_k(K_i)$ *dies* at K_j for $j > i$ if $\varphi_k^{i,j}$ for $j > i$ if $\varphi_k^{i,j}(\alpha) = 0$ but $\varphi_k^{i,j-1}(\alpha) \neq 0$.

The notion behind homology classes "being born" and "dying" for the filtrations that we have discussed is the following: when new faces appear, cavities can appear (such as in the transition between Figure 4.1c and Figure 4.1d) and disappear (such as in the transition between Figure 4.1d and Figure 4.1e), so homology classes are born and die respectively in these transitions. Knowing when the cavities appear and disappear is the core of persistent homology, which leads to the following definition:

Definition 4.1.22. If a filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_{m-1} \subseteq K_m = K$ of a complex K comes from a filtering function $f : K \to \mathbb{R}$ for which $K_i = \{\sigma \in K \,|\, f(\sigma) \leq \epsilon_i\}$ and $K_j = \{\sigma \in K \,|\, f(\sigma) \leq \epsilon_j\}$, then if a homology class α is born at K_i and dies at K_j we say that the *persistence* of α is $\epsilon_j - \epsilon_i$.

The persistence of homology classes measures the lifetime of the cavities that appear along the filtration, but it does not say at which point they appeared or disappeared. This is solved by combining both in the representation of persistence, which starts with *persistence barcodes*. A persistence barcode is a collection of horizontal line segments in a plane coordinate system whose x-axis has marks at $\{\epsilon_0, \ldots, \epsilon_m\}$ (the values of the filtering function) and whose y-axis represents classes of the homology vector spaces H_0, H_1, and so on. If a homology class α is born at K_i and dies at K_j, we draw a horizontal segment $[\epsilon_i, \epsilon_j)$. If a class never dies, the segment is instead a ray $[\epsilon_i, \infty)$. We order them by length, hence giving more importance to homology classes with higher persistence.

The *persistence diagram* associated with a barcode has a point (b_i, d_i) in a coordinate plane for each segment in the barcode starting at b_i and ending at d_i. Thus, a point (b_i, d_i) in a persistence diagram represents the persistence of a homology class from the complex that originated the barcode. The rays $[b_i, \infty)$ are represented as points (b_i, ∞) where ∞ is a fixed arbitrary value larger than ϵ_m. The diagonal $b = d$ is added in the persistence diagram as well, as it is useful to check the importance of the points (points near the diagonal have very small persistence, which can mean that they are just noise). In both barcodes and diagrams, we represent classes of different homology vector spaces with different colors.

Persistence diagrams are the standard way of working with persistent homology of filtrations. Since we will be working with several images of a cardiac cycle, which means several cubical complexes and hence several persistence diagrams, it will be useful to introduce a notion of distance or similarity between persistence diagrams.

Definition 4.1.23. Let D and D' be two persistence diagrams, and suppose they have the same number of points with infinite death coordinate. A *matching* between D and D' is a bijective function $\phi : D \to D'$ where diagonal points are counted with infinite multiplicity.

Definition 4.1.24. The family of *Wasserstein distances* between two persistence diagrams is defined as

$$W_p[q](D, D') = \min_{\varphi : D \to D'} \left(\sum d_q((x, y), \varphi(x, y))^p \right)^{1/p},$$

where $d_q((x, y), (x', y')) = (|x - x'|^q + |y - y'|^q)^{\frac{1}{q}}$. We say that p is the *order* of the distance.

In other words, the distance between diagrams is obtained by pairing each point on diagram 1 with a point on diagram 2 (pairing the infinite points with each other and without mixing different homology vector spaces) or with a point in the diagonal. Then, out of all possible pairings (matchings), we take the one that minimizes the sum of distances, weighted by the order $p \in [1, \infty]$ (a higher p gives more importance to points that are further away from the diagonal). The parameter q controls the choice of the distance d.

Since the matchings are made for each homology vector space, we actually obtain as many Wasserstein distances between two diagrams as homology vector spaces are represented in those diagrams. A common approach (which we use) is to add every distance through all dimensions to get a single distance between two diagrams.

Persistence descriptors

For TDA, we need some way to retrieve information, in numerical or vector form, from persistence diagrams. Here we introduce the two metrics used in this project. For this section, we let $\{(b_i, d_i)\}_{i \in I}$ be the set of points of a certain homology dimension outside of the diagonal, with $d_i \neq \infty$, in a given persistence diagram.

Definition 4.1.25. The *total persistence* of $\{(b_i, d_i)\}_{i \in I}$ is defined as

$$P = \sum_{i \in I} (d_i - b_i)$$

which is the accumulated persistence of all the given points. Points are counted with their corresponding multiplicities.

Definition 4.1.26. The *entropy* of $\{(b_i, d_i)\}_{i \in I}$ is defined as

$$E = -\sum_{i \in I} \frac{d_i - b_i}{P} \log_2 \frac{d_i - b_i}{P}.$$

It can be proved that the entropy of a persistence diagram takes values in the interval $[0, \log_2 n]$, where n is the number of points in the set $\{(b_i, d_i)\}_{i \in I}$. A high entropy value means that the points in the diagram have similar persistences, and $\log_2 n$ is reached if all points have the same persistence.

Persistence diagrams in general, and entropy in particular, are stable (Cohen-Steiner, Edelsbrunner, and Harer, 2007; Atienza and Rucco, 2020), in the sense that small perturbations on the data generate small perturbations in the diagrams and in the entropy. This makes them useful tools to track the variation between similar but evolving images.

Total persistence and entropy are both used as ways to retrieve information from a persistence diagram, and there is no general preference to use one or the other. As it is logarithmic, entropy may be more adequate in cases where there are outliers or when comparing persistence diagrams with very different amounts of points.

Note that these descriptors are calculated for each homology dimension, so for example a 3D image will provide three values of entropy: one for H_0, one for H_1 and one for H_2.

TDA and temporal data

Topological data analysis has not been widely used for temporal data. One approach to use TDA for time series is to turn them into point clouds and calculating the

persistence diagram of those clouds (Wu and Hargreaves, 2020), but this was too intricate for our project (starting with images, then obtaining time series, then transforming the series into point clouds would make us lose much information about the important features). Another approach uses persistence vineyards (introduced in Cohen-Steiner, Edelsbrunner, and Morozov, 2006; more recently used in Bubenik and Bush, 2023), which studies the evolution of points in a temporal stack of persistence diagrams. This seemed suitable for our project, but we dismissed it because it is a very recent technology and it lacks implementation in the TDA libraries, as it is still developing and not precisely defined.

Hence, we had to find other ways to describe the variation of topological descriptors, adapted to the nature of our data. We go over these ways in 4.2.2, after a brief explanation of the calculation of persistence diagrams and their descriptors.

4.2 Time-varying persistence features for cardiac data

In our data we have three images for each frame, and several frames for each patient's cycle. Hence, each patient will be described by several persistence diagrams and their descriptors.

4.2.1 Persistence diagrams and descriptors of a patient's cycle

We start by computing the persistence diagram of every frame of the cycle. Then, for each persistence diagram that we have generated, we have to compute its persistence descriptor. As we said earlier, there is not a clear advantage of entropy over total persistence, and it generally depends on the addressed problem. We were more inclined to use entropy, because the diagrams had a wide range of amounts of points (see Figure 4.2). This is due to the very diverse nature of the ACDC dataset. Hence, we considered that the logarithmic character of entropy could help us with classification, so we decided to use it instead of total persistence.

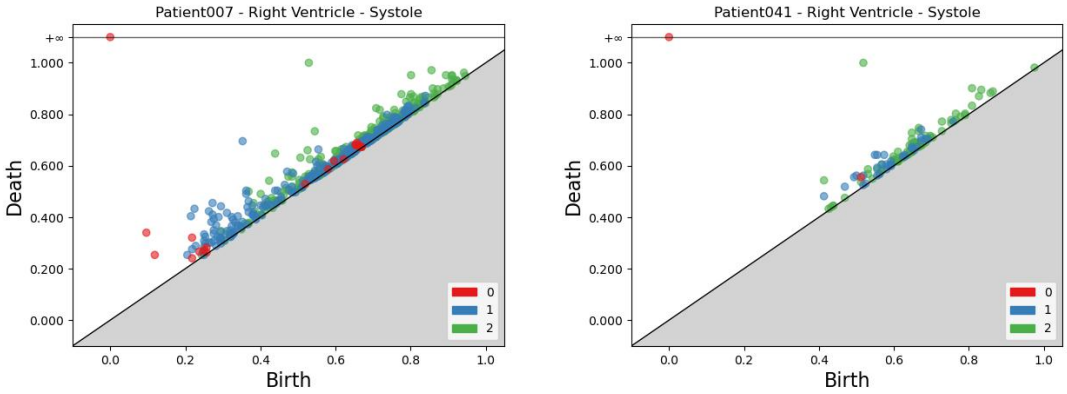
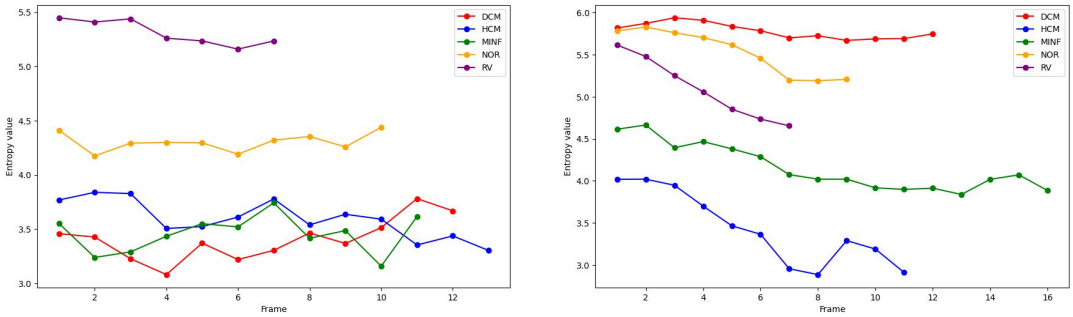


FIGURE 4.2: Two persistence diagrams of the right ventricle of two different patients

After having computed a persistence diagram of every frame of the cycle, we computed the Wasserstein distance (with $p = q = 2$, see Definition 4.1.24) between the persistence diagram of a frame and the persistence diagram of the next frame for all three regions of interest. So for the first patient, whose cycle consists of 12 frames (see Figure 2.2), we obtain 11 distances for each region. The idea behind calculating the distances between diagrams is to grasp the changes in the evolution of the cycle.

This distance time series measures how different are the topologies of consecutive images in the cycle, which is precisely the main goal of our project: to use all the topological information available in the cycle and not just the endpoints.

In Figure 4.3 we can see some examples of time series of topological descriptors. These are series of dimension 1 entropy of the right and left ventricle of the heart. The five colours correspond to five random[2] patients of each of the groups (see Table 2.1). The series show that the right ventricle entropy clearly distinguishes the patient with abnormal right ventricle, and the left ventricle entropy presents significant differences for the 5 classes.



(A) Right Ventricle - Dimension 1 entropy series from 5 patients of the 5 subgroups

(B) Left Ventricle - Dimension 1 entropy series from 5 patients of the 5 subgroups

FIGURE 4.3: Examples of entropy time series

4.2.2 Extraction of time-varying features

After processing and applying TDA tools, we obtain 12 time series that describe each patient: for each of the 3 parts of the heart, we have the distance time series and the entropy time series in dimensions 0, 1 and 2. Our hypothesis is that these time series, that encode every frame of the cardiac cycle and its evolution, contain useful information about the patient's subgroup that is lost if we only use the initial and final state of the cycle.

One important factor about these time series is that they are of different lengths between patients (as we can see in Figure 4.3), ranging from 7 to 16 frames, 6 to 15 in the case of the distance time series. This immediately dismissed the option of using them directly as the features for a machine learning algorithm, and we wanted to avoid putting the time series directly in a neural network. TDA is quite clear with the process it follows, so using a black box algorithm at this point is risky because we might lose explainability. Moreover, the dataset is very small, which makes it more difficult to train a neural network, even with data augmentation.

The next step, then, was to study how to extract features from this short and uneven time series. This proved to be quite difficult, because usual time series methods such as ARIMA or calculating the peaks of the Fourier transform were not applicable to time series of this length. ARIMA in particular, or any other measure of autocorrelation, was also pointless, not only because of the length but also because we are working with time series that represent a single cycle. In this context, autocorrelation or forecasting has no meaning.

Even with these unusual conditions, it was necessary for our project to find features that reflected the temporal nature of the data and its evolution, so we considered the following measures.

[2]The 5 patients in Figure 4.3a are not the same as the 5 patients in 4.3b.

Mean, variance and skewness

The *mean* was an obvious choice for extraction of features, as well as the *variance*. These are the first and second standardized moments, and we also wanted to add the third one, also known as *skewness*, which is defined as

$$\mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$

for a random variable X with mean μ and standard deviation σ. Skewness measures the relative size of the two tails of the distribution. Thus, a time series with many small values and few large values will be right-tailed, which means positive skewness, while a time series with many large values and few small values will be left-tailed, i.e., have negative skewness. Symmetric distributions will have zero skewness (the other way around is not always true, see Bastianin, 2020). Skewness is significant for us because it contains information about significant changes in the cardiac cycle. From this perspective, for an entropy time series a skewness close to zero means that the topology of the heart has not made very significant changes during the cycle or that the changes have been regular, while big (positive or negative) skewness means that the topology of the heart has changed significantly at some point.

We also considered kurtosis, which is the fourth standardized moment and measures the behaviour of the tail. However, in the end we dismissed it because for short time series as ours, the behaviour of the tail was not noticeable, and some early tests generated meaningless values.

Normalized jumps

We define the *normalized jumps* of a time series as the sum in absolute value of the differences of the series divided by its length. That is, if we have a time series $X = (X_1, \ldots, X_n)$, the jumps are defined as

$$\frac{1}{n}\sum_{i=1}^{n-1}|X_{i+1}-X_i|.$$

Normalization is necessary because of the diverse length of the time series along the dataset.

This is, again, a measure of how large the changes are in the topology of the heart during a cycle. While in some cases the distance between diagrams was regular, in other cases there were very big jumps. The main problem with this metric is that it does not contain information about when the changes happen.

Normalized argmin and argmax

The *argmax* and *argmin* functions, applied to a time series, return the position where the maximum and minimum are located respectively. We normalize them by dividing that position by the length of the time series, again because of the diverse length of the time series along the dataset. These features contain more temporal information than the previous ones, as they provide the relative position in the series of its extrema, so it can distinguish between early or late maxima and early or late minima.

Permutation entropy

Permutation entropy (introduced in Bandt and Pompe, 2002) is a measure of the complexity or randomness of a time series. Unlike other entropies, such as Shannon entropy, it considers the order of the values in the time series rather than just their distribution, which for series as ours is very important. Permutation entropy divides the time series into overlapping windows of a certain dimension d and performs permutations over this window. Then it computes the probability of each permutation and calculates the entropy of these probabilities, that is,

$$H(d) = -\sum p(\pi) \log p(\pi),$$

where the sum runs over all permutations π of order d. A high entropy means randomness and noise, while a low entropy indicates more structure and predictability.

One important choice was the dimension d of the windows. For comparison purposes, we had to choose the same d for every time series, as generating a different d depending on the length of the series would generate inconsistent results. A general recommendation for d is that $d! << n$, where n is the length of the time series, so given that our shortest time series is just 6 steps long, our only option seemed to be $d = 2$. However, in Cuesta-Frau et al., 2019 it is argued that the recommendation $d! << n$ is very restrictive for real-world time series and that a higher d tends to reach better classification even if it does not fulfill the inequality. Hence, we decided to choose $d = 3$.

Even though these features are already normalized with respect to the length of the time series, the length itself could also be considered as a feature. The experts that work with the BCN-AIM Lab, however, have never considered that the duration of a cycle in an MRI is significant for diagnosis, so we decided to discard that option.

Chapter 5

Experiments and results

In this chapter we go over the results of all the experiments performed with the ROUSEG and NETSEG approaches. For both, the experiments consist of multi-class classification (the five classes are in Table 2.1) using the features described in the previous chapter. We used five machine learning models: Random Forest (RF), Gradient Boosting (GB) eXtreme Gradient Boosting (XGB), Support Vector Machine (SVM) and K-Nearest Neighbours (KNN), each of them finetuned with 5-fold cross-validation GridSearchCV.

The experiments were conducted on a computer equipped with an AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz, 16 GB of RAM (3200 MHz), and a 512 GB SSD, running Windows 11. Computational models were implemented using Python 3.11.10 and GUDHI 3.10.1.

5.1 Evaluation metrics for multi-class classification

Since we are dealing with a multi-class classification problem, we will use the usual metrics: accuracy and macro-averaged precision, recall and f1-score. We define the last three as follows:

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{F1-Score} = 2 \cdot \frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}$$

where TP stands for the True Positive cases, FP for False Positive and FN for False Negative. Macro-averaging means calculating the metric for each class and then taking the arithmetic mean across all classes. Since there is no class imbalance in our data, the macro-average coincides with the weighted-average.

Precision and recall are particularly important in medicine (Hicks, Strümke, Thambawita, et al., 2022), as wrong diagnoses have far more serious consequences than in other fields. For our experiments, a low precision or low recall would mean incorrect diagnoses, either between sick and healthy patients or between different diseases. For a binary sick-healthy classification, recall would be the most important one, because it grows as false negatives decrease and false negatives (i.e., determining that a sick person is healthy) are critical mistakes.

We compute the confusion matrix as well, which is particularly useful in a problem like ours because it allows us to detect if the classifier is having problems distinguishing between two particular diseases.

Finally, we also compute the ROC-AUC score. While this metric is generally used for binary classification, it can be extended to multi-class using a one-vs-rest

approach, where the ROC-AUC is calculated for each class treated as the "positive" class, then macro-averaged.

5.2 Feature selection

After extracting the 7 features from the 12 time series, we were left with 84 features for each patient. TDA is well known for being able to describe things with few variables, but the temporal nature of our data made the number of features grow very quickly. To avoid dimensionality issues (especially with a small dataset like ours) and keep the model simple, we implemented a process of feature selection.

We first wanted to find a way to check the relevance of the features within the training test. We considered using the ANOVA test (a generalization of the t-test for more than two variables) to discard statistically irrelevant features before classifying, but in the end we chose mutual information as the way to select features.

Mutual information is a measure of the mutual dependence between two variables, that is, it quantifies the amount of information obtained about one random variable through another. It is widely used for feature selection in machine learning (e.g., Beraha et al., 2019; Huang et al., 2024) and for finding statistical dependencies in general.

Following MacKay, 2003, for two random variables X and Y that take values in the sets $\mathcal{X}$ and $\mathcal{Y}$ respectively, their mutual information $I(X;Y)$ is calculated as

$$I(X;Y) = H(X) - H(X|Y)$$

where $H(X)$ is the (marginal) entropy

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x)$$

and $H(X|Y)$ is the conditional entropy

$$H(X|Y) = -\sum_{x\in\mathcal{X},\, y\in\mathcal{Y}} p(x,y)\log p(x|y).$$

With these formulas, we can understand what mutual information is quantifying: entropy is a measure of the average level of information associated with the variable's potential states or outcomes, and the conditional entropy $H(X|Y)$ is a measure of the average level of information remaining in x when y is known. Hence, mutual information measures the average reduction in information about x that results from learning the value of y. Mutual information satisfies $I(X;Y) = I(Y;X)$ and $I(X;Y) \geq 0$, and it is equal to zero when X and Y are independent (because then knowing y gives no information about x). So if $I(X;Y)$ is big, knowing y represents a big part of the information contained in x, and this is how mutual information operates for feature selection; if the labels are X and Y is a feature, it computes $I(X;Y)$ for every feature, which allows us to rank them from biggest to smallest MI score, i.e., by how much they are related to the labels.

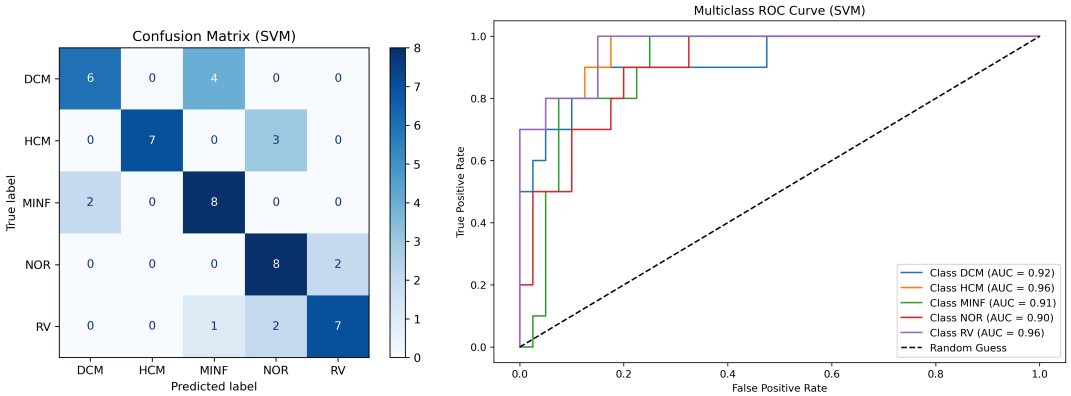
We computed the rankings for ROUSEG and NETSEG's train data (see Appendix A), and we use them in the following sections. With these rankings, we perform three classifications: one with every feature, one with the features that have an MI score bigger than a fixed threshold (0.1), and one with the optimal number of features for each model, checked by tracking the evolution of the classification as we add features along the ranking. Optimality is measured by accuracy and F1-score.

5.3 Multi-class classification results with ROUSEG

In this section we present the results of the multi-class classification experiments using the ROUSEG segmentation approach. Table 5.1 shows the results using all 84 features (without feature selection):

TABLE 5.1: ROUSEG best results for different models without selection of features

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC Score
RF	0.64	0.65	0.64	0.64	0.92
GB	0.60	0.67	0.60	0.59	0.84
XGB	0.66	0.70	0.66	0.67	0.87
SVM	**0.72**	**0.75**	**0.72**	**0.72**	**0.93**
KNN	0.70	0.74	0.70	0.70	0.93



(A) Confusion matrix (B) ROC curve

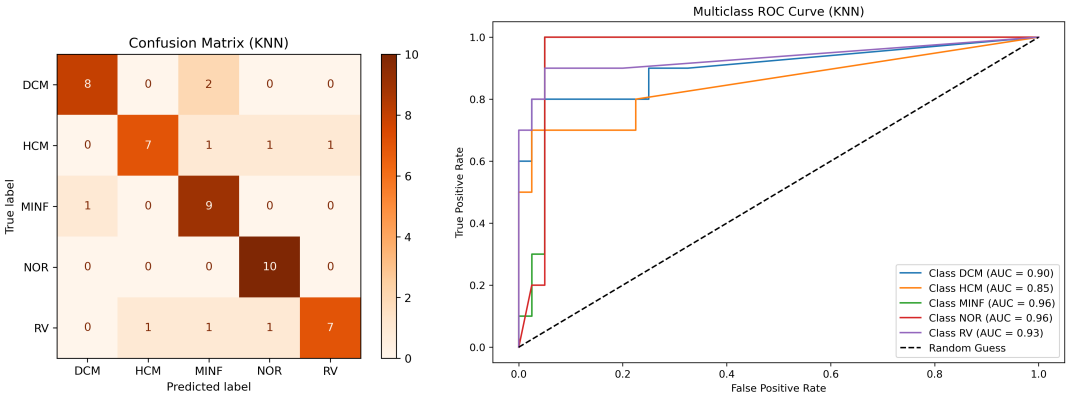
FIGURE 5.1: Confusion matrix and ROC curve for ROUSEG - All features - SVM

Figure 5.1 shows the performance of the best model in this experiment, SVM. We can observe that it mixes DCM and MINF predictions up (which will be a recurrent occurrence in this chapter) and that it tends to overpredict that a patient is healthy, which leads to a big false positive rate for the NOR subgroup.

Table 5.2 shows the results using the threshold selection of 42 features:

TABLE 5.2: ROUSEG best results for different models with threshold selection of features (42)

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC Score
RF	0.62	0.64	0.62	0.62	0.93
GB	0.62	0.68	0.62	0.61	0.84
XGB	0.68	0.71	0.68	0.69	0.87
SVM	0.82	0.86	0.82	0.81	0.94
KNN	**0.82**	**0.83**	**0.82**	**0.82**	**0.92**



(A) Confusion matrix (B) ROC curve

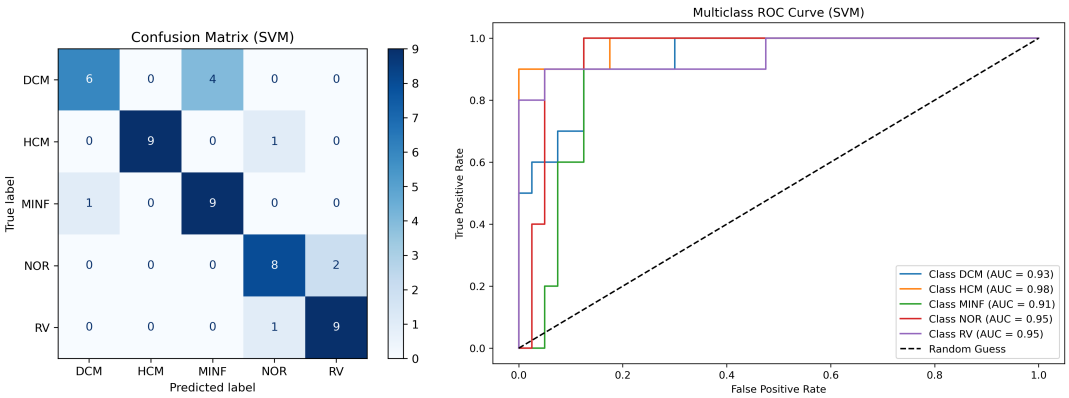
FIGURE 5.2: Confusion matrix and ROC curve for ROUSEG - Threshold selection - KNN

The KNN and SVM models achieve remarkably higher metrics with this selection of features than without it, while RF, GB and XGB do not present significant changes. Figure 5.2 shows the performance of the KNN model, which attains much better classification along the classes. There is a slight tendency to overpredict MINF and NOR.

Table 5.3 shows the results using the optimal number of features for each model:

TABLE 5.3: ROUSEG best results for different models with optimal selection of features for each model

Model	Nº Features	Accuracy	Precision	Recall	F1-Score	ROC-AUC Score
RF	59	0.68	0.72	0.68	0.69	0.92
GB	23	0.70	0.72	0.70	0.70	0.88
XGB	56	0.72	0.76	0.72	0.73	0.88
SVM	**39**	**0.82**	**0.83**	**0.82**	**0.82**	**0.95**
KNN	42	0.82	0.83	0.82	0.82	0.92



(A) Confusion matrix (B) ROC curve

FIGURE 5.3: Confusion matrix and ROC curve for ROUSEG - Optimal selection - SVM

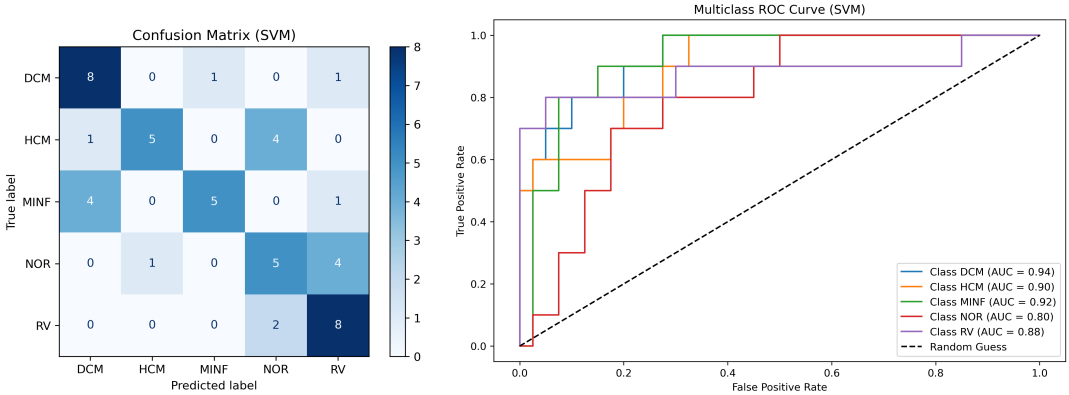
With optimal selection of features, RF, GB and XGB improve slightly, but are still far behind SVM and KNN. Figure 5.3 shows the performance of the SVM model, which presents a very good classification for every class except DCM. The confusion between DCM and MINF that we observed in Figure 5.1 is still present.

5.4 Multi-class classification results with NETSEG

In this section we present the results of the multi-class classification experiments using the NETSEG segmentation approach. Table 5.4 shows the results using all 84 features (without feature selection):

TABLE 5.4: NETSEG best results for different models without selection of features

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC Score
RF	0.62	0.63	0.62	0.62	0.89
GB	0.58	0.58	0.58	0.58	0.82
XGB	0.60	0.60	0.60	0.60	0.87
SVM	**0.62**	**0.66**	**0.62**	**0.62**	**0.89**
KNN	0.60	0.65	0.60	0.60	0.87



(A) Confusion matrix (B) ROC curve

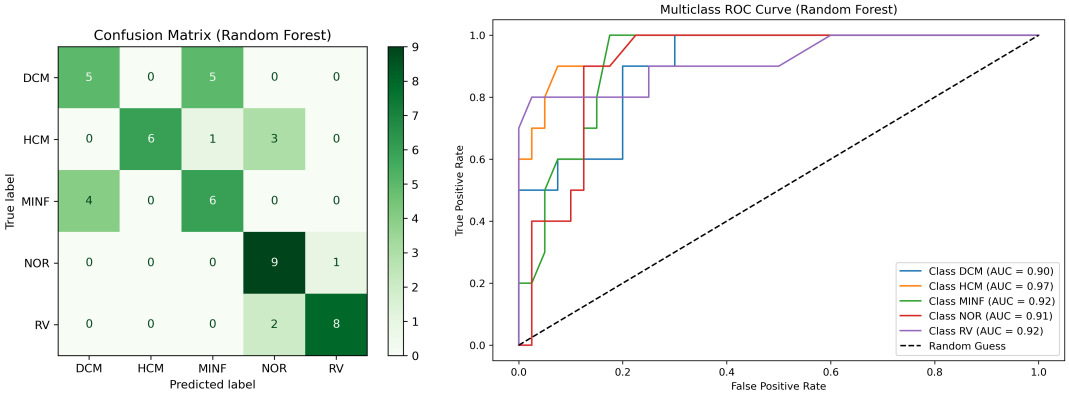
FIGURE 5.4: Confusion matrix and ROC curve for NETSEG - All features - SVM

The models for this experiment achieve similar metrics. SVM, shown in Figure 5.4, presents the usual mixing between DCM and MINF (with overprediction tending to DCM in this case) and some significant wrong predictions of HCM as NOR and NOR as RV.

Table 5.5 shows the results using the threshold selection of 37 features:

TABLE 5.5: NETSEG best results for different models with threshold selection of features (37)

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC Score
RF	**0.68**	**0.72**	**0.68**	**0.68**	**0.92**
GB	0.58	0.59	0.58	0.58	0.83
XGB	0.60	0.61	0.60	0.59	0.87
SVM	0.60	0.62	0.60	0.60	0.91
KNN	0.68	0.69	0.68	0.68	0.91



(A) Confusion matrix (B) ROC curve

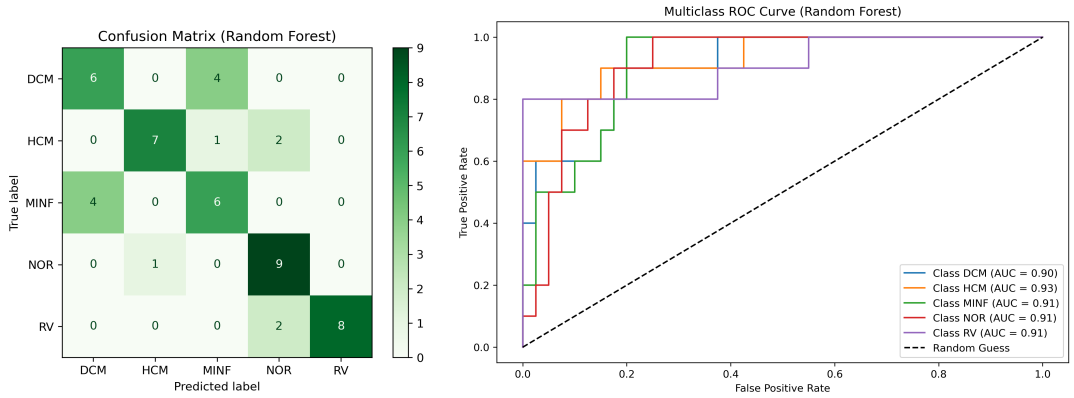
FIGURE 5.5: Confusion matrix and ROC curve for NETSEG - Threshold selection - RF

The only models that improve with the threshold selection are RF and KNN. RF, shown in Figure 5.5, presents similar problems to those of the previous experiment, with the wrong NOR as RV predictions corrected but an almost 50/50 right-wrong classification between DCM and MINF patients

Table 5.6 shows the results using the optimal number of features for each model:

TABLE 5.6: NETSEG best results for different models with optimal selection of features for each model

Model	Nº Features	Accuracy	Precision	Recall	F1-Score	ROC-AUC Score
RF	**12**	**0.72**	**0.74**	**0.72**	**0.72**	**0.91**
GB	22	0.70	0.70	0.70	0.70	0.87
XGB	12	0.68	0.69	0.68	0.68	0.89
SVM	20	0.70	0.71	0.70	0.70	0.89
KNN	71	0.68	0.73	0.68	0.69	0.90



(A) Confusion matrix (B) ROC curve

FIGURE 5.6: Confusion matrix and ROC curve for NETSEG - Optimal selection - RF

This experiment generates significantly better results for all the models except KNN, but RF is still the best with just 12 features. Figure 5.6 shows very similar results to those in Figure 5.5.

5.5 Discussion and comparison with previous results

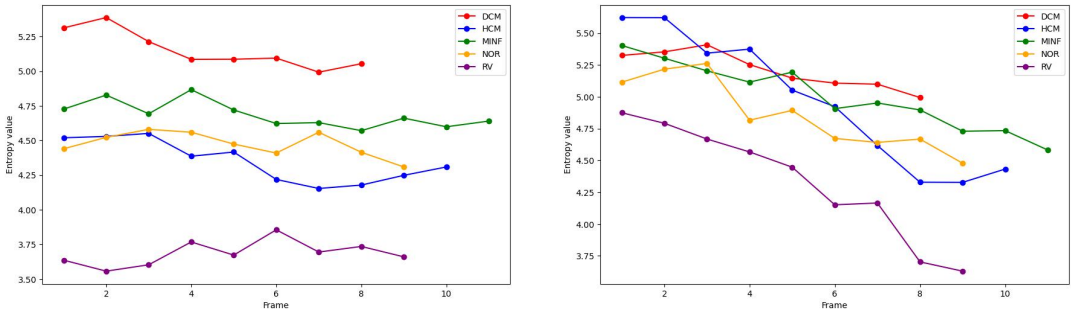
In this section, we compare our results with those obtained in Anguas, 2023, shown in Table 5.7. As stated in Chapter 1, these results were obtained using only the diastolic and systolic frames.

TABLE 5.7: Best results in Anguas, 2023

Method	Accuracy	Precision	Recall
Radiomics	0.76	0.77	0.76
TDA	0.74	0.74	0.73
Radiomics+TDA	0.79	0.79	0.78

On one hand, the best results for the ROUSEG experiments with either type of feature selection improved the classification obtained with only the diastolic and systolic frames, even with radiomics+TDA. On the other hand, the NETSEG experiments achieved worse results, even with optimal feature selection. In Figure 5.7 we can see a ROUSEG-NETSEG comparison of the same time series (left ventricle, dimension 1 entropy) for five selected patients (the same in both graphs), which suggests why NETSEG is working worse. The series in ROUSEG present just small variations, because we are using the same mask for every frame, and this clearly distinguishes the classes from one another. In contrast, the series in NETSEG present high variation, which makes them more faithful to reality, but they get mixed up and are only distinguishable in the final points.

If we check the MI scores in Appendix A, we see that for ROUSEG the mean features were more important, while in NETSEG the variance and jump features were more important. The MI scores also show that the use of Wasserstein distances between diagrams did not prove very useful for classification for both ROUSEG and NETSEG. This could be a consequence of a wrong approach to its calculation, since we simply added the distances for each homology dimension instead of designing a weighted distance.



(A) Left Ventricle - Dimension 1 entropy series from 5 patients of the 5 subgroups (ROUSEG)

(B) Left Ventricle - Dimension 1 entropy series from 5 patients of the 5 subgroups (NETSEG)

FIGURE 5.7: Comparison between ROUSEG and NETSEG Left Ventricle - Dimension 1 entropy series for 5 patients of the 5 subgroups

In general, the most important features for both approaches were the means, the variances and the jumps. A few argmin and argmax had some degree of importance, and most of the skewness and permutation entropies where on the lower half of the tables. In fact, some of them provided zero information about the diseases. We believe that skewness had very small significance because our time series were very short, and permutation entropy was not important because there was no difference in the "randomness" of the series between patient groups.

Regarding the importance of the different regions of the heart, in both approaches we obtained expected results; the left ventricle is the most important region for classification, because three diseases present differences there, followed by the right ventricle for the patients with abnormal right ventricle (RV). This last group, as well as the healthy patients, seem to be easily distinguished by topological descriptors, while patients with previous myocardial infarction (MINF) and dilated cardiomyopathy (DCM) are often mistaken for each other.

The SVM and KNN models were the best or among the best for the two approaches. Both are among the most explainable models (Salih et al., 2023), so the results are transparent and we believe that the whole classification pipeline could be understood by a medical expert.

Finally, we remark that we are using a single train-test split (the default ACDC dataset division) for the sake of comparison with these previous results. With such a small dataset, particular cases in the test set can generate metrics that are not representative. Hence, it is necessary to check the generalizability of our work, as we discuss in the last section.

Chapter 6

Conclusions and future work

6.1 Conclusions

The results presented in the previous chapter yield two conclusions. First and foremost, we proved the relevance of TDA for cardiac disease diagnosis. Introducing more topological details improved classification without using radiomics (which contain a large number of features, risking overfitting, and are costly to calculate). However, it remains unclear why the ROUSEG approach, which used less accurate segmentation masks than NETSEG, achieved higher classification accuracies. One possible explanation could be that the variation of topology along a cardiac cycle is less determinant than the information retrieved from the most relevant time instants.

We conclude that TDA is a useful tool for cardiac imaging and it can achieve comparable or even better results than radiomics. Moreover, our findings suggest that introduction of temporal information can yield improved classification results under suitable settings.

6.2 Future work

As we mentioned in Chapter 1, a big challenge for data science in medicine is domain shift, so after this project we would like to check if the methodology is generalizable to other datasets. The confusion matrices in Chapter 5 already give hints about what diseases seem to be clearly identifiable using our methods, so we could start with that, and check for other heart diseases as well. We have the opportunity to do so in the near future, because after introducing my project to people from the BCN-AIM Artificial Intelligence in Medicine Lab, I gained access to the UK Biobank (https://www.ukbiobank.ac.uk/) database, which contains thousands of medical data, including hundreds of cardiac MRI recordings as those in the ACDC dataset, so we will be able to check the generalizability of our work.

Regarding other possible methodologies, one that we considered but we did not implement was describing each patient's cardiac cycle with a graph and using graph neural networks to classify. The idea behind this approach was to assign a node to each diagram and have each edge between diagrams be weighted with the Wasserstein distance between them. The nodes would contain the nine entropy descriptors of the three diagrams of the three regions of interest. This idea occurred to us when we realized that we could not use the series directly as features due to the varying length of the cardiac cycles. We dismissed it because we wanted to avoid neural networks and because it needed more development; the resulting "graphs" were just a branch of weighted nodes and edges, so they had very little graph structure. Nevertheless, we think that this idea can be developed further and has the potential to be a promising approach.

Appendix A

Mutual information scores

In this appendix we display the MI scores of the features for both experiments. "L", "M" and "R" stand for Left Ventricle, Myocardium and Right Ventricle respectively, and "dist", "e0", "e1" and "e2" stand for Distance, Dimension 0 Entropy, Dimension 1 Entropy and Dimension 2 Entropy respectively.

A.1 ROUSEG MI scores

TABLE A.1: Mutual information scores for features in ROUSEG

Feature	**Mutual information score**
L_e0_mean	0.588736
L_e1_mean	0.470237
L_e2_mean	0.468011
R_e2_mean	0.408692
R_e0_mean	0.399721
L_e0_var	0.386066
R_e0_var	0.385742
L_e1_jumps	0.357864
R_e1_mean	0.336981
L_e1_var	0.313763
L_e2_var	0.308686
M_e1_jumps	0.295472
M_e0_mean	0.272543
L_e0_jumps	0.244934
R_e1_var	0.237003
M_dist_var	0.234757
M_dist_jumps	0.231593
L_e1_argmin	0.228720
M_e2_jumps	0.222533
R_e2_jumps	0.201239
M_e1_mean	0.197879
M_e1_var	0.195544
M_e1_argmin	0.194388

Feature	**Mutual information score**
L_e2_jumps	0.193756
R_e2_var	0.167690
M_dist_argmax	0.149258
R_e0_jumps	0.149031
R_e0_skew	0.148126
L_dist_argmin	0.143596
M_e2_mean	0.143047
M_e0_permentro	0.138783
R_e1_jumps	0.138210
M_e0_jumps	0.137812
L_dist_jumps	0.132169
M_dist_mean	0.130348
L_dist_mean	0.128668
M_e1_skew	0.128605
R_e1_skew	0.125063
R_e1_permentro	0.124660
L_e0_permentro	0.124461
R_dist_mean	0.119899
R_e0_permentro	0.113595
R_e2_argmax	0.098884
M_e1_permentro	0.094432
L_e2_skew	0.093256
L_e1_skew	0.091879
M_dist_argmin	0.088905
R_e2_permentro	0.086422
L_e1_argmax	0.084279
L_e0_argmax	0.080324
L_e2_permentro	0.077789
R_e2_argmin	0.073256
L_dist_argmax	0.068826
L_e2_argmax	0.063913
L_e2_argmin	0.060156
M_e1_argmax	0.053506
R_dist_jumps	0.053122
L_e0_skew	0.048180
L_dist_permentro	0.042864
M_e0_argmax	0.042548
M_e0_argmin	0.039794
L_dist_var	0.037712
M_dist_permentro	0.035264

Feature	Mutual information score
M_e0_var	0.033932
R_e0_argmin	0.029715
R_dist_var	0.028909
M_e2_var	0.022093
R_e1_argmax	0.019291
M_e2_permentro	0.016158
M_e2_argmax	0.014449
M_e2_argmin	0.007501
L_e1_permentro	0.000000
R_dist_skew	0.000000
R_dist_permentro	0.000000
R_dist_argmax	0.000000
R_dist_argmin	0.000000
R_e2_skew	0.000000
L_e0_argmin	0.000000
M_e0_skew	0.000000
R_e1_argmin	0.000000
L_dist_skew	0.000000
R_e0_argmax	0.000000
M_e2_skew	0.000000
M_dist_skew	0.000000

A.2 NETSEG MI scores

TABLE A.2: Mutual information scores for features in NETSEG

Feature	Mutual information score
L_e2_var	0.581166
L_e2_jumps	0.527956
L_e1_var	0.524543
L_e0_jumps	0.425281
R_e0_jumps	0.371193
L_e1_jumps	0.358846
R_e1_mean	0.350789
L_e2_mean	0.333611
R_e0_mean	0.332286
L_e0_var	0.328601
L_e0_mean	0.312192
M_e2_mean	0.310331
L_e1_mean	0.297011

Feature	**Mutual information score**
R_e1_jumps	0.267534
M_e2_var	0.259066
M_e1_var	0.251419
L_e1_argmin	0.228363
R_e2_mean	0.217956
M_e1_mean	0.216621
L_e2_permentro	0.198279
R_e2_var	0.193004
M_e2_jumps	0.183957
R_e0_var	0.170034
L_e0_argmax	0.164330
L_e1_argmax	0.160756
R_dist_argmin	0.141389
R_dist_argmax	0.138450
M_e0_jumps	0.137758
M_e0_mean	0.137166
M_e1_jumps	0.135194
M_e2_argmin	0.118715
L_e2_argmax	0.117473
M_dist_mean	0.115168
M_dist_permentro	0.109985
R_e0_skew	0.109025
R_e2_jumps	0.104083
M_e1_argmin	0.103279
M_dist_jumps	0.088953
M_e0_permentro	0.082252
L_dist_permentro	0.081790
M_dist_argmin	0.080616
L_dist_mean	0.075508
M_e1_argmax	0.074828
M_dist_var	0.068298
M_dist_argmax	0.066354
L_dist_var	0.062633
R_e0_argmin	0.059384
R_e1_var	0.059210
R_e2_permentro	0.058711
R_e1_argmin	0.058286
R_dist_skew	0.048245
M_e0_argmin	0.045844
M_e2_argmax	0.044273

Feature	Mutual information score
R_dist_mean	0.036925
M_dist_skew	0.033441
L_dist_jumps	0.032417
R_e0_permentro	0.031439
L_e0_argmin	0.025129
M_e0_var	0.019680
L_e1_permentro	0.016548
M_e0_argmax	0.011658
L_e2_argmin	0.011517
L_dist_argmin	0.010126
L_e0_permentro	0.006590
L_e0_skew	0.004357
R_e0_argmax	0.000000
R_dist_var	0.000000
L_e2_skew	0.000000
R_dist_jumps	0.000000
R_dist_permentro	0.000000
L_e1_skew	0.000000
R_e2_argmin	0.000000
R_e2_argmax	0.000000
L_dist_skew	0.000000
R_e1_skew	0.000000
M_e2_permentro	0.000000
R_e1_permentro	0.000000
R_e1_argmax	0.000000
M_e2_skew	0.000000
M_e1_permentro	0.000000
M_e1_skew	0.000000
R_e2_skew	0.000000
M_e0_skew	0.000000
L_dist_argmax	0.000000

Bibliography

Anguas, Marina (2023). *Integrating topological features to enhance cardiac disease diagnosis from 3D CMR images*. URL: https://hdl.handle.net/2445/202401.

Atienza, Rocío González-Díaz and Matteo Rucco (2020). "On the Stability of Persistent Entropy and New Summary Functions for Topological Data Analysis". In: *ArXiv* arXiv:1803.08304. URL: https://arxiv.org/abs/1803.08304.

Bandt, Christoph and Bernd Pompe (Apr. 2002). "Permutation Entropy: A Natural Complexity Measure for Time Series". In: *Phys. Rev. Lett.* 88 (17), p. 174102. DOI: 10.1103/PhysRevLett.88.174102. URL: https://link.aps.org/doi/10.1103/PhysRevLett.88.174102.

Bastianin, Andrea (2020). "Robust measures of skewness and kurtosis for macroeconomic and financial time series". In: *Applied Economics* 52.7, pp. 637–670.

Beraha, Mario et al. (2019). "Feature Selection via Mutual Information: New Theoretical Insights". In: *CoRR* abs/1907.07384. arXiv: 1907.07384. URL: http://arxiv.org/abs/1907.07384.

Bernard, O. et al. (Nov. 2018). ""Deep Learning Techniques for Automatic MRI Cardiac Multi-structures Segmentation and Diagnosis: Is the Problem Solved ?"" In: *IEEE Transactions on Medical Imaging* 37.11, pp. 2514–2525. DOI: doi:10.1109/TMI.2018.2837502.

Bubenik, Peter and Johnathan Bush (2023). *Topological feature selection for time series data*. arXiv: 2310.17494 [math.AT]. URL: https://arxiv.org/abs/2310.17494.

Campello, Víctor M. (2023). "Generalizability in multi-centre cardiac image analysis with machine learning". PhD thesis. Universitat de Barcelona. URL: https://diposit.ub.edu/dspace/handle/2445/206020.

Chazal, Frédéric and Bertrand Michel (2021). *An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists*. arXiv: 1710.04019 [math.ST]. URL: https://arxiv.org/abs/1710.04019.

Cohen-Steiner, David, Herbert Edelsbrunner, and John Harer (2007). "Stability of Persistence Diagrams". In: *Discrete & Computational Geometry* 37.1, pp. 103–120. DOI: 10.1007/s00454-006-1276-5.

Cohen-Steiner, David, Herbert Edelsbrunner, and Dmitriy Morozov (2006). "Vines and Vineyards by Updating Persistence in Linear Time". In: *Proceedings of the 22nd Annual Symposium on Computational Geometry*. ACM, pp. 119–126. DOI: 10.1145/1137856.1137877. URL: https://dl.acm.org/doi/10.1145/1137856.1137877.

Cuesta-Frau, David et al. (2019). "Embedded Dimension and Time Series Length. Practical Influence on Permutation Entropy and Its Applications". In: *Entropy* 21.4, p. 385. DOI: 10.3390/e21040385.

Darvishi, Saeed et al. (Feb. 2013). "Measuring Left Ventricular Volumes in Two-Dimensional Echocardiography Image Sequence Using Level-set Method for Automatic Detection of End-Diastole and End-systole Frames". In: *Research in Cardiovascular Medicine* 2.1. Epub 2013 Feb 24, pp. 39–45. DOI: 10.5812/cardiovascmed.6397.

Diaz, Oliver et al. (2021). "Data preparation for artificial intelligence in medical imaging: A comprehensive guide to open-access platforms and tools". In: *Physica Medica* 83, pp. 25–37. ISSN: 1120-1797. DOI: https://doi.org/10.1016/j.ejmp.2021.02.007. URL: https://www.sciencedirect.com/science/article/pii/S1120179721000958.

Hatcher, Allen (2002). *Algebraic Topology*. Cambridge: Cambridge University Press, pp. xii+544. ISBN: 0-521-79160-X; 0-521-79540-0.

Hicks, Steven A., Inga Strümke, Vajira Thambawita, et al. (2022). "On evaluation metrics for medical applications of artificial intelligence". In: *Scientific Reports* 12.1, p. 5979. DOI: 10.1038/s41598-022-09954-8.

Huang, Lin et al. (2024). "Time Series Feature Selection Method Based on Mutual Information". In: *Applied Sciences* 14.5. ISSN: 2076-3417. DOI: 10.3390/app14051960. URL: https://www.mdpi.com/2076-3417/14/5/1960.

Isensee, Fabian et al. (2021). "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation". In: *Nature Methods* 18, pp. 203–211. DOI: 10.1038/s41592-020-01008-z.

Izquierdo, Cristian et al. (2021). "Radiomics-Based Classification of Left Ventricular Non-compaction, Hypertrophic Cardiomyopathy, and Dilated Cardiomyopathy in Cardiovascular Magnetic Resonance". In: *Frontiers in Cardiovascular Medicine* 8. ISSN: 2297-055X. DOI: 10.3389/fcvm.2021.764312. URL: https://www.frontiersin.org/journals/cardiovascular-medicine/articles/10.3389/fcvm.2021.764312.

Kaczynski, T., K. Mischaikow, and M. Mrozek (2004). *Computational Homology*. Applied Mathematical Sciences. Springer New York. ISBN: 9780387408538. URL: https://books.google.es/books?id=V1kyZ1E7pLgC.

MacKay, David J.C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge, UK: Cambridge University Press. ISBN: 9780521642989.

Martin-Isla, Carlos et al. (2020). "Image-Based Cardiac Diagnosis With Machine Learning: A Review". In: *Frontiers in Cardiovascular Medicine* 7. ISSN: 2297-055X. DOI: 10.3389/fcvm.2020.00001. URL: https://www.frontiersin.org/journals/cardiovascular-medicine/articles/10.3389/fcvm.2020.00001.

Pérez-Pelegrí, Manuel et al. (2022). "End-systole and end-diastole detection in short axis cine MRI using a fully convolutional neural network with dilated convolutions". In: *Computerized Medical Imaging and Graphics* 99, p. 102085. ISSN: 0895-6111. DOI: https://doi.org/10.1016/j.compmedimag.2022.102085. URL: https://www.sciencedirect.com/science/article/pii/S0895611122000581.

Salih, Ahmed et al. (2023). "Explainable Artificial Intelligence and Cardiac Imaging: Toward More Interpretable Models". In: *Circulation: Cardiovascular Imaging* 16.4, e014519. DOI: 10.1161/CIRCIMAGING.122.014519. eprint: https://www.ahajournals.org/doi/pdf/10.1161/CIRCIMAGING.122.014519. URL: https://www.ahajournals.org/doi/abs/10.1161/CIRCIMAGING.122.014519.

Scheikl, Patrick M., David P. Grüber, Katharina Glatz-Krieger, et al. (2023). "Can incorrect artificial intelligence (AI) results impact radiologists, and if so, what can we do about it? A multi-reader pilot study of lung cancer detection with chest radiography". In: *European Radiology*. DOI: 10.1007/s00330-023-09600-9.

Shehab, Mohammad et al. (2022). "Machine learning in medical applications: A review of state-of-the-art methods". In: *Computers in Biology and Medicine* 145, p. 105458. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105458. URL: https://www.sciencedirect.com/science/article/pii/S0010482522002505.

Singh, Yashica, Catherine M. Farrelly, Quin A. Hathaway, et al. (2023). "Topological data analysis in medical imaging: current state of the art". In: *Insights into Imaging* 14.1, p. 58. DOI: 10.1186/s13244-023-01413-w.

Solomon, Elchanan and Paul Bendich (2024). *Convolutional Persistence Transforms*. arXiv: 2208.02107 [math.AT]. URL: https://arxiv.org/abs/2208.02107.

Tran, Phi Vu (2017). *A Fully Convolutional Neural Network for Cardiac Segmentation in Short-Axis MRI*. arXiv: 1604.00494 [cs.CV]. URL: https://arxiv.org/abs/1604.00494.

Upendra, R. R., R. Simon, and C. A. Linte (2021). "A Deep Learning Framework for Image Super-Resolution for Late Gadolinium Enhanced Cardiac MRI". In: *Computing in Cardiology* 48. DOI: 10.23919/cinc53138.2021.9662790.

Vaduganathan, Muthiah et al. (2022). "The Global Burden of Cardiovascular Diseases and Risk: A Compass for Future Health". In: *Journal of the American College of Cardiology* 80.25, pp. 2361–2371. ISSN: 0735-1097. DOI: https://doi.org/10.1016/j.jacc.2022.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0735109722073120.

Wu, Chengyuan and Carol Anne Hargreaves (2020). *Topological Machine Learning for Multivariate Time Series*. arXiv: 1911.12082 [math.AT]. URL: https://arxiv.org/abs/1911.12082.