UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

DOBLE GRAU DE MATEMÀTIQUES I ENGINYERIA INFORMÀTICA

Treball final de grau

ENHANCING CARDIAC IMAGE SEGMENTATION THROUGH PERSISTENT HOMOLOGY REGULARIZATION

Autor: Ignacio Javier Morera Barrios

Director Informàtica:	**Dr. Sergio Escalera**
Director Matemàtiques:	**Dr. Carles Casacuberta**
	Sr. Rubén Bautista Ballester
Realitzat a:	**Matemàtiques i Informàtica**
Barcelona,	**19 d'octubre de 2022**

Abstract

Cardiovascular diseases are a major cause of death and disability. Deep learning-based segmentation methods could help to reduce their severity by aiding in early diagnosing but high levels of accuracy are necessary. The vast majority of methods focus on correcting local errors and miss the global picture. To address this issue, researchers have developed techniques that incorporate global context and consider the relationships between pixels. Here, we apply persistent homology, a branch of topology that studies the topological structure of shapes, along with deep learning methods to improve the heart segmentation. We use multidimensional topological losses to avoid spurious components and holes and increase the total accuracy. We evaluate the performance of three different approaches: using the dice and pixel-wise losses with the sum of persistences of label diagrams as a regularizer, using the dice and pixel-wise losses with the bottleneck distance as a regularizer, and using both losses without any regularization. We find that, while more computationally demanding, the methods using topological regularizers outperform the other method in terms of accuracy.

2023 Mathematics Subject Classification. 55N31, 62R40, 68T09

Resumen

Las enfermedades cardiovasculares son la principal causa de muerte e incapacidad. Los métodos de segmentación basados en aprendizaje profundo podrían ayudar a reducir su gravedad mediante la ayuda en la detección temprana, pero para ello son necesarios altos niveles de precisión en la segmentatión. La gran mayoría de los métodos se centran en corregir errores locales y pierden la visión global. Para abordar este problema, los investigadores han desarrollado técnicas que incorporan el contexto global y consideran las relaciones entre los píxeles. En este trabajo aplicamos el concepto de homología persistente, una rama de la topología que estudia la estructura topológica de las formas, junto con los métodos de aprendizaje profundo para mejorar la segmentación del corazón. Utilizamos funciones de pérdida topológicas multidimensionales para evitar componentes y agujeros espurios y aumentar la precisión total. Evaluamos el rendimiento de tres enfoques diferentes: utilizando las funciones de pérdida *dice* y píxel a píxel, junto con la suma de persistencias de los diagramas de las etiquetas como un regularizador; utilizando las funciones de pérdida *dice* y píxel a píxel con la distancia *bottleneck* como un regularizador; y utilizando ambas funciones de pérdida sin ningún tipo de regularización. Encontramos que, aunque son más computacionalmente exigentes, los métodos que utilizan regularizadores topológicos superan al otro método en términos de precisión.

Resum

Les malalties cardiovasculars són la principal causa de mort i incapacitat. Els mètodes de segmentació basats en l'aprenentatge profund podrien ajudar a reduir la seva gravetat mitjançant l'ajuda de la detecció precoç, però per això, són necessaris que tinguin un alt nivell d'exactitud en la segmentació. La gran majoria dels mètodes es centren en corregir errors locals i perden la visió global. Per abordar aquest problema, els investigadors han desenvolupat tècniques que incorporen el context global i consideren les relacions entre els pixels. En aquest treball apliquem el concepte d'homologia persistent, una branca de la topologia que estudia l'estructura topològica de les formes, juntament amb els mètodes d'aprenentatge profund per millorar la segmentació del cor. Utilitzem funcions de pèrdua topològiques multidimensionals per evitar components i forats indesitjats i augmentar la precisió total. Evaluem el rendiment de tres enfocaments diferents: utilitzant les funcions de pèrdua *dice* i pixel a pixel, juntament amb la suma de persistències dels diagrames de les etiquetes com a regularitzador; utilitzant la funció de pèrdua *dice* i pixel a pixel amb la distància *bottleneck* com un regularitzador, i utilitzant ambdues funcions de pérdua sense cap tipus de regularització. Trobem que, encara que són més computacionalment exigents, els mètodes que utilitzen regularitzadors topològics superen l'altre mètode en termes de precisió.

Acknowledgements

I would like to express my appreciation to my thesis supervisors Sergio, Rubén and Carles for their invaluable aid and assistance throughout the process. A special thanks to Ruben, who went above and beyond in providing support.

I am also grateful for the unwavering support of my mother, as well as the invaluable last-minute corrections provided by my flatmates, whose support and encouragement have been instrumental in helping me reach this point.

I would like to extend my sincere appreciation to Víctor M. Campello, Carlos Martín-Isla, and Karim Lekadir for providing the dataset and for their patience in answering all of my questions. I am also grateful to the Topology UB team for their guidance and support throughout this process.

I also want to thank my classmates for being a constant source of support throughout my academic journey. Without them, I would not have made it this far.

Thank you all for your support and belief in me.

Contents

Chapter 1

Introduction

Cardiovascular diseases (CVDs) are a leading cause of death globally [4, 23]. According to the World Health Organization, 17.9 million die each year because of CVDs. Early diagnostics could prevent up to 90% of CVDs [6, 7]. Medical imaging has been transformative in healthcare and is essential for diagnosing and treating CVDs. Several technologies, including computed tomography and magnetic resonance imaging, have been particularly useful for providing detailed anatomical information of the heart.

In order to accurately measure morphological and pathological changes, it is often necessary to segment the important structures in cardiac medical images. The process of identifying and separating the various components of the heart, known as whole heart (substructure) segmentation (WHS), is important for analyzing and understanding the anatomy and function of the heart, which translates into a variety of clinical applications, including computer-aided diagnosis [9] and surgery.

Nowadays, machine learning models have achieved enough computational power to aid in cardiac magnetic resonance (CMR) image segmentation [23]. However, extreme accuracy is required. For example, the inability measuring the circumference or the width of the walls could misidentify heart conditions such as hypertrophic cardiomyopathy [8]. In response, some promising deep learning methods have emerged to increase the accuracy of the segmentation. Among them, incorporating prior knowledge about the topology of the segmented object have recently came into the spotlight.

1.1 Brief introduction to deep learning

Deep learning is a subfield of machine learning that involves using layered algorithms to automatically extract useful features from data, in which each layer

performs a specific function. Deep learning has a wide range of applications, such as facial recognition [2] or stock market predictions [3]. In contrast to traditional machine learning approaches, which rely on manually created features and representations of the data, deep learning involves the automatic feature learning through the use of these layered algorithms in order to improve its predictions.

The deep learning field was originated trying to reproduce a model that emulated intelligent behaviour [1], despite the reason of its current success is not its similarity with the human brain. Inspired by the function of the biological neurons, it started with a primitive model working on circuits and evolved to the creation of the first net and the later discovery of the *Perceptron*. The first neural networks were created stacking multiple perceptrons together. By placing perceptrons side by side we can create a single neural network's layer. By stacking multiple one-layer neural networks on top of each other, we can create a multi-layer neural network, also known as a multi-layer perceptron. More insight on the topic [1] is available if the reader wished to explore further.

Afterwards, the appearance of the *forward pass* algorithm and *backpropagation* algorithms contributed to the success of modern neural network models. Now, they are the basis of some of the most relevant and commonly used machine learning models.

Neural networks are designed to learn complex mathematical functions based on a set of training data points and to generalize these functions to new data from the same distribution. They are trained using a dataset of input-output pairs, and the goal of the training is to learn a function that maps inputs to outputs based on the examples in the training dataset. The predictions made by the neural network will generally be most accurate for inputs that are similar to the ones that the network was trained on. This can be compared to using a Taylor Series approximation to model a function, which may be accurate within a certain range but could fail outside of that range.

However, there are some results, the *Universal Approximation Theorems* that discuss the approximation capabilities of feedforward networks on the space of continuous functions between two Euclidean spaces. One of the versions states that a neural network with one hidden layer that has a sufficient but finite number of neurons can accurately approximate any continuous function with inputs within a specific range, with the desired amount of error, given enough hidden units [43, 44] and depending on the activation functions used [42].

However, we cannot know the necessary size of the network [45]. In practice, having a big enough network is not achievable most of the time, but neural networks are still pretty useful to approximate any kind of function.

Since neural networks were discovered, researchers have tried to maintain their

capabilities while reducing the number of computational units. Nowadays, there is no generic way to determine the best structure for a neural network [14]. The depth and width of neural networks interact with other *hyperparameters*, making impossible to isolate a specific hyperparameter and study it independently. Parameters like the *depth* of the network, the type of the *activation functions* used, the thresholds of the activation functions or the *cost function* are just too many variables to keep in mind. An in-depth analysis is important to carefully choose the hyperparameters that fit the problem we want to solve. This careful selection will decide the *generalization gap*, the difference between the performance of a machine learning model on the training data and its performance in test data.

In 2015, it was proved that the depth of a feedforward neural network is exponentially more valuable than the width [33], using weak assumptions about the activation functions. One of the reasons is that multiple layers allow to learn features at different levels of abstraction, while wide enough layers could memorize the desired outputs. However, adding more layers to a neural network increases the non-linearity of the model. This means that the optimization problem becomes more non-convex, or difficult to optimize, as the number of layers increases. Highly non-convex problems can be challenging for optimization algorithms to solve, and may result in the algorithm failing to find a good minimum or overfitting the dataset. Nonetheless, neural networks tend to converge to optimal local minima despite this non-convexity, as it was pointed out in [15].

1.2 Heart segmentation

For a long time the segmentation was done manually, but it is a tedious and impractical job. However, automating this process can be challenging due to the large variation in heart shape, the varying quality of clinical images and the lack of clear edge or boundary information. Moreover, unlike other organs like the lungs and the liver, the heart is made up of multiple elements, including the left and right ventricular cavities, the aorta, myocardium and atria. In order to develop effective algorithms for WHS, a large dataset of training images is often necessary. In addition, it is difficult to perform comparisons between different methods, largely due to differences in the datasets and evaluation metrics used.

Most of the time the access to large datasets is not possible and the number of training datasets is limited. Furthermore, the technique used to obtain the images can vary the image quality and appearance. This caused that the algorithms developed usually did not have good accuracy, despite the performance and reliability that they actually showed.

Many state of the art methods focused on improving the segmentation using

atlas-based approaches (Anatomical atlases with annotated locations and shapes) [10], which had the advantage of providing solid spatial priors and are effective for use in medical imaging due to their ability to impose constraints on both location and shape. However, this proposals cause artifacts and disconnected regions, so, as a bigger amount of data was becoming accessible [11] another solutions that were based in convolutional neural networks appeared in the last decade.

Convolutional neural networks (CNNs) are a type of machine learning model that have been successful in tasks related to visual data, such as classification, recognition, and segmentation. These models use spatial context and weight sharing between pixels across multiple layers to learn about visual data.

Out of these networks, the fully convolutional (segmentation) networks (FCNs) are one of the most commonly used [13]. FCNs have the advantage of being very efficient for pixel-level predictions because they only involve feedforward operations, and they also incorporate spatial context into the predictions. The architecture consists of two modules: a downsampling path and an upsampling path. The first path includes convolutional and max-pooling layers, which are commonly used in convolutional neural networks for image classification tasks. The second path includes deconvolutional layers, which increase the size of the feature maps and output the score masks.

However, FCNs have some limitations [12], including difficulty in capturing local geometry and the need for a large amount of training data. To outperform the FCNs and to reduce the amount of data needed to train the models, the U-Net was created [16]. Speedly, FCNs and its modified version, U-Net, filled the state of the art [16, 17, 18, 19]. In addition to this models, a measure of the overlap between two sets of data, the Dice similarity coefficient, was proposed [20], increasing the performance of the models by solving the problem of evaluating how well the algorithm identified the boundaries between different objects or regions in the image.

For instance, in the Automatic Cardiac Diagnosis Challenge (ACDC) [21], all eight of the highest ranked segmentation techniques were neural network-based methods. In the M&Ms challenge, all the teams used different variations of the UNet architecture as their baseline model for image segmentation [22]. Finally, in the Multi-Modality Whole Heart Segmentation (MM-WHS) challenge the deep learning models showed better accuracies [23], despite some of the models got bad results in the blinded evaluation.

Focusing on the state of the art of the M&Ms challenge, which dataset will be used in this work, all the teams used different variations of the UNet architecture as their baseline model for image segmentation, as we mentioned before. Four teams used the nnUNet, which includes UNet architectures in 2D and 3D

as well as a cascaded UNet. Four other teams used a traditional UNet, while the remaining teams used UNets combined with residual connections, attention mechanisms, modified structures, or dilated convolutions. One team also proposed using a modified Spatial Decomposition Network with an AdaIN decoder. There was a degree of variability in the backbone architectures used among the different participants.

A further line of research has extended U-Net's architecture to 3D [24] and it seems that it's beginning to obtain promissing results [24, 25]. However, we are going to focus in the 2D approach in this work.

Actual state-of-the-art methods often fail when the input data is not consistent, a problem that plagues models based on the segmentation of heart elements [22]. Additionally, these convolutional models require a sufficient amount of data to learn the multiple elements present in medical images and understand their shapes and relationships [19]. This is a mayor problem, since dealing with a limited number of training datasets is a common issue for most medical imaging computing problems. There is also a difficulty in transferring the weights learned from a fully convolutional network (FCN) to new tasks because the elements within the FCN are closely connected for the current problem.

A tool that has shown promising results solving the inconveniences of CNN is incorporating prior knowledge to convolutional neural networks. Using prior knowledge about the shape of the organs and its spatial location can improve the performance of image analysis techniques, especially when images are compromised by random pixels [26], since it tends to predict values that are located on the reduced-dimension data structure that has been extracted. This is a common problem in cardiac image segmentation, as raw volumetric CMR images from standard scans may contain various artifacts due to limitations in clinical acquisition protocols. Also, it has been proven that incorporating prior knowledge in medical image segmentation has improved the accuracy of the models [34]. One of the reasons is that the artifacts are not propagated to the prediction [28]. Another one is that this models converge faster to the actual shape.

While techniques such as CNN-based segmentation are highly effective, it can be challenging to incorporate prior knowledge about the highly constrained nature of anatomical objects into these approaches. However, many methods have appeared that are not tied to any specific neural network architecture or application and can be used in combination with any of the current leading techniques for segmentation [26]. This in contrast to the classification models that solely use cross-entropy functions that do not take into consideration the underlying information of the output space, since they evaluate each pixel individually.

1.3 Persistent homology applied to heart segmentation

We find in the state of the art two types of topology specification: *connectivity* and *genus* [34]. The first specification focuses on the connectivity of the segmentation, while genus focuses on avoiding spurious or internal holes in the final segmentation. We find some early examples of topology preservation in segmentation [35, 36, 37]. Also, topological losses have been applied for some years to different segmentation problems, like the segmentation of the fetal cortex [38]. However, the state of the art methods focus in developing different kind of CNN with local losses that miss the topological features.

Both Dice similarity coefficient and binary cross-entropy are commonly used as loss functions for training and evaluating image segmentation algorithms, but they have a limitation in that they evaluate each pixel individually and not the overall structure of the image. This can lead to topological errors in the segmentation, since connectivity and holes are ignored. Small voxel-wise mistakes can lead to large topological errors.

In order to address this issue and improve the accuracy of whole-structure segmentation, most researchers are applying techniques from algebraic topology. Algebraic topology is a branch of mathematics that uses algebraic methods to study topological spaces and their properties. It provides a powerful framework for understanding the fundamental characteristics of spaces, such as the number of connected components, the presence of holes or voids, and the connectivity of different parts of the space. Algebraic topology also offers a way to compare and contrast different spaces, which is useful in a wide range of applications. Specifically, one of the main tools in algebraic topology is Homology, which provides a way to quantify and classify the different types of holes or voids in a space.

Given the ability of algebraic topology to extract fundamental characteristics of a space, it is also particularly useful in the field of data analysis, where the data can be represented as a space with different features. In such field, one specific application of algebraic topology is the use of persistent homology.

Persistent Homology is used to extract and summarize topological information from large or complex data sets. This features are analyzed by constructing a sequence of spaces, known as a filtration. For each space in the filtration, the topological features are identified and their lifetimes are recorded. These lifetimes, known as persistence intervals, provide a measure of the stability of the topological feature. The idea behind is to track the evolution of topological features as the scale of analysis changes.

Recently, researchers have been focusing on developing loss functions that prioritize the preservation of topology [29, 30, 31, 32]. Most of these functions compare the model predictions with the Ground Truth by utilizing Betti numbers, a

tool in algebraic topology that can extract topological features of a space. Betti numbers are used to count the number of connected components, holes or voids in a space, and provide a way to summarize and compare the topology of different spaces. As Betti numbers are discrete, it is necessary to use persistent homology to compute topological features of space, a mathematical tool that measures the topological features of data that persist across multiple scales. To study the topology generated by the data, instead of focusing on a specific threshold and computing its persistence, a differentiable function based on persistence diagrams (the central tool from topological data analysis) is used as a regularizer for the neural networks loss function.

Determining the homology of various topological spaces can be a challenging task, but simplifying the space can make the process easier. One approach is to use simplified structures called *cubical complexes*, a union of unit cubes of various dimensions, as a replacement for the original space. These cubical complexes allow us to apply algorithmic computation of homology, which can make the task of understanding the original space easier. On the other hand, one might not have direct access to the original space, but instead only a set of discrete samples which can then be used to construct a cubical complex. Cubical complexes allow representations of volume data. Using cubical complexes allows to avoid usual triangulation of the space and to reduce the size of the abstract complex, since it triangulates spaces created out of pixeled data in the natural way [49]. The application of such complexes has been deeply studied [50, 51, 52, 53].

To encode topological features of cubical complexes we will use multi-scale topological descriptors of data called *persistence diagrams*. To incorporate them as regularizers to optimize a neural network, we will provide a method to compute the gradient of the topological penalty: providing a unified framework for optimization with persistent diagrams using gradient descent is required.

Recently, some differentiability approaches focusing on persistent homology have emerged in an attempt to incorporate persistent homology regularizers to heuristic functions [56, 61, 60]. Using results in [56] we will provide a framework to differentiate the objective function

$$\mathcal{M} \longrightarrow Bar \longrightarrow \mathbb{R},$$

with $\mathcal{M}$ a parameter space with a differential structure and *Bar* the set of persistence diagrams.

We will provide a more extensive introduction to persistent homology in later sections and review the performance and accuracy of different topological regularizers applied to the M&Ms dataset.

1.4 Contributions

In this work, a convolutional neural network with a persistent homology-based topological loss for multi-class image segmentation was implemented. This approach builds upon previous successful uses of persistent homology in image segmentation by incorporating topological information [29, 30, 31, 32, 35, 36, 37]. The topological loss function was integrated into the overall loss function of the CNN to ensure that the model not only considered traditional pixel-level accuracy, but also preserved the topological information of the images. The results of the model were promising, with a higher accuracy in image segmentation compared to traditional geometrical losses.

The model was tested on a dataset of images from the second edition of the M&Ms challenge and it was able to achieve good results in terms of accuracy. In addition, the model was able to generalize well to new images and the results were consistent across different images.

In the following sections, more details on the specific problem addressed, the dataset used, and the pre-processing steps will be provided. Also, a more in-depth analysis of the results will be presented and the limitations of the approach and potential future work will be discussed. Overall, this project contributes to the growing body of research on the use of persistent homology in image segmentation and highlights the importance of preserving topological information for robust image analysis.

Chapter 2

Neural Networks

2.1 References

Prior to the definition of artificial neural networks, we should introduce its main components: neurons, activation functions, biases, weights and layers. We will use the books *Deep Learning Architectures: A Mathematical Approach* [39] and *Deep Learning* [40] as a guide.

2.2 Artificial Neurons

Definition 2.1. ([39], Definition 5.1.1) An abstract neuron is a quadruple (x, w, φ, y), where $x^T = (x_0, \ldots, x_n)$ is the input vector, $w^T = (w_0, \ldots, w_n)$ is the weight vector, with $x_0 = -1$ and $w_0 = b$, the bias and φ is an activation function that defines the outcome function $y = \varphi(x^T w) = \varphi(\sum_{i=1}^{n}(w_i * x_i))$.

The computation of a neuron is divided into two operations: the first one correspond to the summation of the inputs, called *signal*, represented as

$$x^T * w = w^T x = w_1 x_1 + \cdots + w_n x_n - b.$$

The second one corresponds to the application of the activation function. The final result is given by

$$y = f(x) = \varphi(w^T x) = \varphi(w_1 x_1 + \cdots + w_n x_n - b).$$

In the model of the artificial network the abstract neuron adjusts the weights vector w in such a way to minimize a certain error function, called the cost function, that compares the desired target variable $\hat{y}$ to the prediction y, in order to learn the desired variable $\hat{y}$.

Example 2.2. The segmented function

$$g(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 \leq 0 \\ 1 & \text{if } x_1 > 0 \end{cases}$$

can be reproduced updating the weights $w_0 = 0$, $w_1 = 1$ and $w_2 = 0$, so the inequality $x_1 w_1 + x_2 w_2 > b$ is satisfied.

2.3 Activation Functions

Neural networks use nonlinear functions $\varphi \colon \mathbb{R} \to \mathbb{R}$ to learn other nonlinear functions. They are called *activation functions*. An activation function takes the weighted sum of the inputs of the neuron and produces an output. The output of the activation function is used as the final output of the neuron, and it is passed on to the next layer of the network.

Example 2.3. The activation function used in the model of a biological neuron is the function $\varphi \colon \mathbb{R} \to \mathbb{R}$, defined as

$$\varphi(x) = \varphi(\sum_{i=0}^{n}(w_i * x_i)) = \varphi(b + \sum_{i=1}^{n}(w_i * x_i)) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n}(w_i * x_i) > b \\ 0 & \text{if } \sum_{i=1}^{n}(w_i * x_i) \leq b \end{cases}$$

where $b^T = (b_1, \ldots, b_n)$ is called the bias vector.

2.4 Cost Functions

The mathematical tool that measures how well a neural network is performing on a given task is the cost function. It is a function that measures the deviation of the model predictions from the target outputs, the error. It is used to optimize the weights and biases of the connections between neurons during the training process, by minimizing the error between the predicted output and the true output.

Observation 2.1. A cost function can also be called an error function or a loss function.

Example 2.4. In classification tasks a common cost function is the cross-entropy function

$$J(w, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

It is used both in binary and multi-class classification problems.

Another useful cost function to calculate the deviation between a continuous function and a predicted function is the Supremum Error function.

Example 2.5. *Supremum Error Function.* Given a function $g\colon [0,1] \to \mathbb{R}$ and a function $f_{w,b}$ being the predicted mapping of the input and outputs of the previous function, then the cost function between g and f is

$$C(w,b) = \sup_{x\in[0,1]} |f_{w,b}(x) - g(x)|.$$

2.5 Neural Network's Layers

When taking a group of artificial neurons, each with identical inputs $\{x_i\}_{i=1}^n$ and activation function φ, and connecting them together to another neuron, we create a *layer*, a group of artificial neurons assembled together to perform a specific task within the network.

Most neural networks are constructed as a series of stacked layers, each made up of interconnected neurons that process and transmit information. These networks can be represented as graphs, with edges representing the connections between neurons and associated weights. The first layer is referred to as the *input layer* and the last layer is referred to as the *output layer*.

Notation 2.1. The weights are represented with $w_{ij}^{(l)}$, being (l) the index of the layer, i the index of the input neuron and j the index of the output neuron. The index $i = 0$ represents the bias. The neurons $x_i^{(l)}$ are represented using the lower index i to indicate a neuron in the (l)-layer.

As stated in a previous section, a neuron in the $(l+1)$-layer receives a number n of inputs $\{x_i^{(l)}\}_{i=1}^n$ and returns an output $s_n^{(l+1)}$. An activation function φ is applied afterwards obtaining

$$x_n^{(l+1)} = \varphi(s_n^{(l+1)}) = \varphi(\sum_{i=1}^{n}(w_{ij}^{(l+1)}x_i^{(l)} - b_j^{(l+1)})).$$

We can use matrix notation to make the signals in a layer more understandable:

$$\begin{pmatrix} s_1^{(l+1)} \\ \vdots \\ s_n^{(l+1)} \end{pmatrix} = \begin{pmatrix} w_{11}^{(l+1)} & \dots & w_{n1}^{(l+1)} \\ \vdots & & \vdots \\ w_{1n}^{(l+1)} & \dots & w_{nn}^{(l+1)} \end{pmatrix} \begin{pmatrix} x_1^{(l)} \\ \vdots \\ x_n^{(l)} \end{pmatrix} - \begin{pmatrix} b_1^{(l+1)} \\ \vdots \\ b_n^{(l+1)} \end{pmatrix}$$

$$\begin{pmatrix} x_1^{(l+1)} \\ \vdots \\ x_n^{(l+1)} \end{pmatrix} = \begin{pmatrix} \varphi(s_1^{(l+1)}) \\ \vdots \\ \varphi(s_n^{(l+1)}) \end{pmatrix}$$

Notation 2.2. In a (l)-layer we will express the weights matrix as $W^{(l)}$, the input vector as $X^{(l-1)}$, $b^{(l)}$ will be the bias vector and $s^{(l)}$ will be the signal vector.

Using the previously defined notation, the previous relations can be represented as

$$s^{(l)} = {W^{(l)}}^T X^{(l-1)} - b^{(l)},$$
$$X^{(l)} = \varphi(s^{(l)}) = \varphi({W^{(l)}}^T X^{(l-1)} - b^{(l)}).$$

Definition 2.6. The input of a neural network is a finite non-empty sequence $X = (x_1, \ldots, x_n) \in \mathbb{R}^n$.

It is a feature vector from a training example and matches the first layer of the network.

Definition 2.7. The output of a neural network is a finite non-empty sequence $Y = (y_1, \ldots, y_n) \in \mathbb{R}^n$.

The output is the prediction of the neural network and matches the last layer of the network.

Definition 2.8. The depth of a neural network is defined as the $L \in \mathbb{N}$ number of layers.

Definition 2.9. The depth of the lth layer is defined as the $n_l \in \mathbb{N}$ number of neurons.

2.6 Feedforward Neural Networks

A specific type of neural networks is the *Feedforward Neural Network*. These networks have a sequence of layers that are linked together and contain groups of interconnected neurons. The information flows in one direction through these layers. They are also known as *Deep Feedforward Networks* or *Multilayer Perceptrons* (MLPs).

The goal of a feedforward network is to find the values of the parameters θ that result in the best approximation of the target function $f*$. To do this, the feedforward network defines a mapping $y = f(x;\theta)$ and adjusts the values of the parameters θ through the learning process. The objective is to find the values of θ that result in the function $f(x;\theta)$ being as close as possible to the function $f*$.

We can define the output of a feedforward neural network as the predicted mapping $f_{w,b} : \mathbb{R}^n \to \mathbb{R}^m$, with

$$Y_n = \varphi(W^{(n)} Y_{n-1} - b^n),$$
$$Y_1 = X.$$

Observation 2.2. Assuming that all neurons in a feedforward neural network are linear, with $\varphi(x) = x$, the output of a linear network would be

$$Y = W^{(n)}Y_{n-1} = W^{(n)} \dots W^{(1)}X - W^{(n)} \dots W^{(2)}b^1 - \dots - W^{(n)}b^{n-1} - b^n.$$

Taking

$$W = W^{(n)} \dots W^{(1)}$$

and

$$b = -W^{(n)} \dots W^{(2)}b^1 - \dots - W^{(n)}b^{n-1} - b^n$$

we obtain

$$Y = WX - b,$$

indicating the equivalence between the neural network and a single linear neuron.

Feedforward neural networks are often referred to as networks because they consist of layers of interconnected neurons that are connected in a specific arrangement. The most common structure for feedforward neural networks is a chain structure, in which the functions performed by the neurons are connected in a linear sequence. Additionally, feedforward neural networks often use affine transformations followed by fixed nonlinear functions, called activation functions, to process and transmit the information.

A formal definition, then, would be:

Definition 2.10. ([39], Definition 6.2.5) Let $U_l = 1, 2, \dots, 0 \leq l \leq L$, and consider the sequence of affine functions $\alpha_1, \dots, \alpha_L$

$$\alpha_l \colon \mathcal{F}(U_{l-1}) \to \mathcal{F}(U_l)$$

and the sequence of activation functions $\varphi^{(l)} \colon \mathbb{R} \to \mathbb{R}$. Then the corresponding feedforward neural network is the sequence of maps $f_0, f_1, \dots, f_L$ where

$$f_l = \varphi^{(l)} \circ \alpha_l \circ f_{l-1}, \quad 1 \leq l \leq L,$$

with f_0 given.

Next we are going to introduce an important example of feedforward neural network: a *Convolutional Neural Network*, but first we need to define the *convolution* operation:

Definition 2.11. Let $x \colon \mathbb{R} \to \mathbb{R}$ be a continuous function, $a \in \mathbb{R}$ and $w \colon \mathbb{R} \to \mathbb{R}$ a continuous weighting function, a *convolution* is a function $s \colon \mathbb{R} \to \mathbb{R}$ with

$$s(t) = \int x(a)w(t-a)\,da.$$

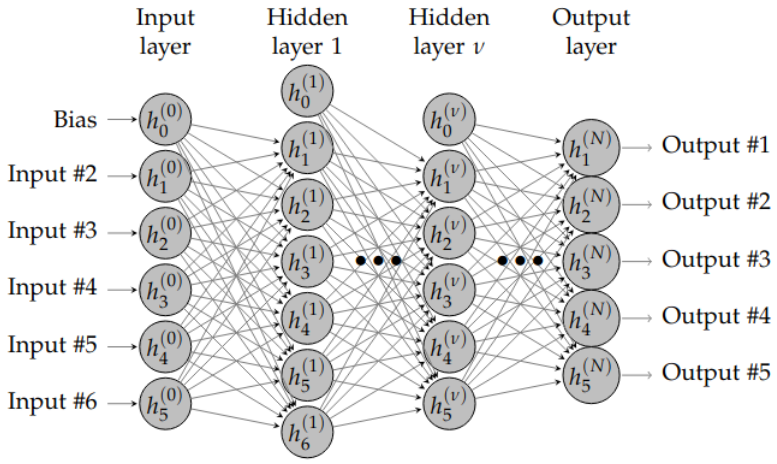


Figure 2.1: Feedforward neural network with N+1 layers and an input and output vectors of equal size. Image source: [55]

Observation 2.3. The convolution function is usually denoted with the asterisk symbol:

$$s(t) = (x * w)(t).$$

Example 2.12. *Convolutional neural networks* (CNNs) are a type of feedforward neural network, known for their use of *convolutional layers*, which are the building blocks of CNNs. These layers are similar to fully-connected layers, but are more efficient to train as they have many weights set to zero.

An example of weights matrix with is

$$W = \begin{pmatrix} w_1 & w_2 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 \end{pmatrix}$$

In the case of convolutions applied to images, a kernel K convolve with an input image I giving a map of features as an output. We can assume that the functions x and w are defined on the integers and apply a discrete convolution [40, Section 9.1]:

$$s(t) = (x * w)(t) = \sum_{i=-\infty}^{\infty} x(a)w(t-a).$$

Since convolutions are applied to multidimensional arrays, we assume that the functions x and w are zero everywhere but in the finite set of values stored in the

arrays. This way, if we apply convolutions to a two-dimensional input I and a two-dimensional kernel K [40, Section 9.1]:

$$s(i,j) = (x * w)(t) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n).$$

A specific case of a convolutional layer is the *Transposed Convolutional Layer*. It performs convolutional operations to carry out upsampling.

Example 2.13. A *pooling layer* is another common component in a convolutional network. It applies a *pooling function* to the output of the layer to further modify it. A pooling function is used to condense the output of the network at a particular location by replacing it with a summary statistic computed from the nearby outputs.This helps to reduce the spatial dimensionality of the output and make the model more robust to small changes in the input.

A common pooling operation is the *max pooling* [46], a down-sampling technique that reduces the spatial size of the input by taking the maximum value over a rectangular set of neighboring pixels:

[39, Section 15.1] Let $f\colon [a,b] \to \mathbb{R}$ be a continuous function and consider the equidistant partition of the interval $[a,b]$

$$a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b.$$

The partition size, $\frac{b-a}{n}$, is called *stride*. Denote by $M_i = \max_{x_{i-1},x_i} f(x)$ and consider the simple function $S_n(x) = \sum_{i=1}^n M_i \mathbb{1}_{[x_{i-1},x_i])}(x)$. The process of approximating the function $f(x)$ by the simple function $S_n(x)$ is called *max pooling*.

Finally, the last example of layer used in convolutional neural networks is the *Dropout* layer.

Example 2.14. A *dropout layer* is a layer that randomly removes a specified percentage of neurons from the network during training, this percentage is a hyperparameter. The purpose of this is to create a balance between the model's performance during training and testing, thus reducing *overfitting*. However, dropping too many neurons might decrease the dimension of the parameter space so much that can cause the parameter space to shrink excessively and, as a result, the model will not fit the data well. This situation is called *underfit*.

Considering the output of a layer with N neurons as

$$y = \sum_{j=1}^{N} \sigma(w_j x + b_j),$$

we drop uniformly n neurons one at a time, obtaining the outputs

$$y_j = y - \sigma(w_j + b_j), \quad j = 1, \ldots, n.$$

The dropout is performed with a probability of $q_i = q = \frac{1}{n}$. Then, the expected output is given by

$$\sum_{j=1}^{n} q_j y_j = \frac{1}{n} \sum_{j=1}^{n} y_j = (1 - \frac{1}{n})y = (1 - q)y.$$

2.7 Gradient descent

Deep learning models employ optimization techniques to minimize a cost function. One key distinction between linear models and neural networks is that the nonlinearity of neural networks often results in nonconvex loss functions, meaning that the optimal solution is not necessarily a global minimum. This makes it more challenging to find the optimal values of the model parameters using traditional optimization methods. As a result, neural networks are typically trained using iterative gradient-based optimizers that aim to drive the cost function to a very low value, rather than finding the global minimum. These optimizers use the gradient of the loss function to update the model parameters in a direction that reduces the loss by determining the derivative of the function $f'(x)$, which indicates the direction to change the input in order to decrease the output:

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x).$$

Observation 2.4. Maximization is also possible in the optimization of the loss function using a minimization algorithm to minimize $-f(x)$.

We will use the concepts of *partial derivative*, *gradient* and *directional derivative*. At a given point x, the partial derivative $\frac{\partial u}{\partial x_i}$ measures the rate of change of the function f with respect to the variable x_i as x_i increases, keeping all other variables as constants. It describes how much the function f changes as a results of a change in the variable x_i at the point x.

On the other hand, the *gradient vector* is the generalization of the derivative in the case where the derivative is computed with respect to a vector.

The gradient vector $\nabla f(x_1, \ldots, x_n)$ computes the direction in which the function f increases most rapidly and the direction orthogonal to the level surfaces, given inputs $x_1, \ldots, x_n$. At each point in the domain of the function f, $\nabla f(x_1, \ldots x_n)$ assigns a vector with element i being the partial derivative of f with respect to the component x_i.

The directional derivative in a particular direction u, which is a unit vector, represents the slope of the function f in that direction. It indicates the rate at which the function f changes as a result of a change in the direction u. To determine the direction in which the function f decreases faster, we can compute the

directional derivative in various directions. The directional derivative provides us with information about the rate of change of f in a specific direction at a given point, and we can use it to identify the direction in which f decreases the most.

To simplify the problem of minimizing this function, we will establish a relationship between the directional derivative and the gradient vector.

Theorem 2.15. [41, Section 14.6, Theorem 3] If f is a differentiable function of x and y, then f has a directional derivative in the direction of any unit vector $\vec{u} = \langle a, b \rangle$ and

$$D_u f(a,b) = f_x(x,y)a + f_y(x,y)b.$$

Proof. If we define a function$g$ of the single variable h by

$$g(h) = f(x_0 + ha, y_0 + hb)$$

then, by the definition of a derivative, we have

$$g'(0) = \lim_{h \to 0} \frac{g(h) - g(0))}{h} = \lim_{h \to 0} \frac{f(x_0 + ha, y_0 + hb) - f(x_0, y_0))}{h} = D_u f(a,b). \quad (2.1)$$

On the other hand, we can write $g(h) = f(x,y)$, where $x = x_0 + ha, y = y_0 + hb$, so the Chain Rule gives

$$g'(h) = \frac{\partial f}{\partial x}\frac{dx}{dh} + \frac{\partial f}{\partial y}\frac{dy}{dh} = f_x(x,y)a + f_y(x,y)b.$$

If we now put $h = 0$, then $x = x_0$, $y = y_0$, and

$$g'(0) = f_x(x_0, y_0)a + f_y(x_0, y_0)b. \quad (2.2)$$

Comparing Equations 3.1 and 3.2, we see that

$$D_u f(a,b) = f_x(x_0, y_0)a + f_y(x_0, y_0)b. \qquad \square$$

Extending this theorem to a number n of variables (which is a matter of notation) we can use it to calculate the partial derivative using the gradient vector, since $f_x(x,y)a + f_y(x,y)b = \langle f_x(x,y), f_y(x,y) \rangle \cdot \vec{u}$. Then, to decrease f we should move in the direction of the negative gradient $-\nabla f(x_1, \ldots, x_n)$. This method is called *gradient descent method* or *steepest descent method*.

In the gradient descent algorithm we start at a chosen point x, which is normally a random point, and pick a learning rate η. At each iteration we compute the gradient of the function at the current point $\nabla f(x_1, \ldots, x_n)$ and we update the current point by moving in the opposite direction of the gradient $-\nabla f(x_1, \ldots, x_n)$,

using the learning rate η to control the size of the steps. The algorithm will be as follows:

Algorithm 1 Gradient descent algorithm

```
procedure GradientDescentAlgorithm(x, η)
    z ← x
    for i ← 0 to N do
        z ← z − ∇f(z)η
    end for
end procedure
```

There are many variations of the algorithm, like *stochastic gradient descent*, in which the gradient is computed using a randomly picked portion of the observation. Nowadays, *stochastic gradient descent* algorithm is the dominant training algorithm for deep learning models.

2.8 Backpropagation

To calculate the gradient of the error, we utilize a widely-known algorithm called *backpropagation*. It is a supervised learning algorithm that uses *gradient descent* to train a neural network.

Let $C(w, b)$ be a neural network's cost function. The goal is to compute the gradient of the function $\nabla C = (\nabla_w C, \nabla_b C)$.

Using the chain rule, the partial derivatives of the weights $w_{i,j}^{(l)}$ and biases b_j are

$$\nabla_w C = \frac{\partial C}{\partial w_{i,j}^{(l)}} = \frac{\partial C}{\partial s_j^{(l)}} \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}, \quad \nabla_b C = \frac{\partial C}{\partial b_j^{(l)}} = \frac{\partial C}{\partial s_j^{(l)}} \frac{\partial s_j^{(l)}}{\partial b_j^{(l)}}.$$

The first factor will be denoted as

$$\delta_j^{(l)} = \frac{\partial C}{\partial s_j^{(l)}}$$

and we define it as the sensitivity of the error in relation of the signal.

Also, we can simplify both expressions computing the second factor of each partial derivative:

$$\frac{\partial s_j^{(l)}}{\partial w_{i,j}^{(l)}} = x_i^{(l-1)}, \quad \frac{\partial s_j^{(l)}}{\partial b_j^{(l)}} = -1.$$

Thereby, we obtain after the corresponding substitutions

$$\frac{\partial C}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} x_i^{(l-1)}, \qquad \frac{\partial C}{\partial b_j^{(l)}} = -\delta_j^{(l)}.$$

Finally, we obtain the expression

$$\nabla C = (\nabla_w C, \nabla_b C) = \delta_j^{(l)}(x_i^{(l-1)}, -1).$$

We deduce that in order to obtain the gradient ∇C it is sufficient to compute the deltas $\delta_j^{(l)}$.

In order to obtain an algorithm to calculate the partial derivatives, we express the deltas of the $(l-1)$-layer in terms of the ones in the (l)-layer. First step is to take in consideration that the signal $s_j^{(l-1)}$ affects all signals $s_j^{(l)}$, since every delta depends on all the deltas from the posterior layers

$$\delta_i^{(l-1)} = \frac{\partial C}{\partial s_i^{(l-1)}} = \sum_{j=1}^{d^{(l)}} \frac{\partial C}{\partial s_j^{(l)}} \frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}}.$$

Substituting yields

$$\delta_i^{(l-1)} = \sum_{j=1}^{d^{(l)}} \delta_j^{(l)} \frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}}.$$

We have obtained an expression to represent the delta $\delta_i^{(l-1)}$ in terms of the deltas $\delta_j^{(l)}$ from the above layer. Focusing in the partial derivative $\frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}}$,

$$\frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}} = \frac{\partial}{\partial s_i^{(l-1)}}\Big(\sum_{j=1}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} - b_j^{(l)}\Big).$$

Using $x_j^{(l)} = \varphi(s_j^{(l)})$,

$$\frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}} = \frac{\partial}{\partial s_i^{(l-1)}}\Big(\sum_{j=1}^{d^{(l-1)}} w_{ij}^{(l)} \varphi(s_i^{(l-1)}) - b_j^{(l)}\Big) = w_{ij}^{(l)} \varphi'(s_i^{(l-1)}).$$

Finally, we obtained an expression to compute the gradient, the *backpropagation formula*

$$\delta_i^{(l-1)} = \varphi'(s_i^{(l-1)}) \sum_{j=1}^{d^{(l)}} \delta_j^{(l)} w_{ij}^{(l)}.$$

Using this formula we iterate backwards applying the derivatives of the weights and biases of the (l)-layer to compute the ones in the $(l-1)$-layer. The calculations *backpropagate* through the layers. The computation of the last layer's deltas depend on the cost function.

Example 2.16. Assuming the cost function is the half mean squared error,

$$C(w,b) = \frac{1}{2}\|x^{(L)} - z\|^2.$$

We can compute the partial derivative of the cost function with respect to the signal $s_j^{(L)}$ to obtain the delta $\delta_j^{(L)}$,

$$\delta_j^{(L)} = \frac{\partial C}{s_j^{(L)}} = (x^{(L)} - z_j)\varphi'(s_j^{(L)}).$$

Chapter 3

Persistent Homology

3.1 References

Our references during the whole section will be the book *Computational Topology* [47] and the paper *A framework for differential calculus on persistence barcodes* [56].

3.2 Complexes

3.2.1 Simplicial Complexes

Definition 3.1. Let $S = \{u_0, u_1, \ldots, u_k\}$ be a set of points in $\mathbb{R}^d$. A point $\sum_{i=0}^{k} \lambda_i u_i$ is called an *affine combination* of the set S if the values of λ_i add up to 1. The set of all such affine combinations is known as the *affine hull* of S.

Definition 3.2. A set of points $S = \{u_0, u_1, \ldots, u_k\}$ is *affinely independent* if for any two identical affine combinations $\sum \lambda_i u_i$ and $\sum \mu_i u_i$ we have $\lambda_i = \mu_i$ for all i.

We derive that each affine hull is a k-plane when the $k+1$ points are affinely independent. When determining whether or not a set of $k+1$ points are affinely independent, it is necessary to check for linear independence among the k vectors $u_i - u_0$ for $1 \leq i \leq k$. In the context of $\mathbb{R}^d$, it is impossible to have more than d linearly independent vectors. As a result, the maximum number of affinely independent points that can exist in $\mathbb{R}^d$ is $d+1$.

We will use the notation I to refer to the index set of labels.

Definition 3.3. A *convex combination* is an affine combination with $\lambda_i > 0$ for all $i \in I$. The set of all such affine combinations is known as a *convex hull*.

Definition 3.4. We define a *k-simplex* as the convex hull of a set S of $k+1$ affinely independent points, $\sigma = \text{conv}\{u_0, \ldots, u_k\}$. The dimension of σ is $\dim \sigma = k$.

The points $u_0, \ldots, u_k$ are the *vertices* of the k-simplex σ, while the *edges* are convex hulls of pairs of vertices. When the convex hull is a subset $H \subset S$ with $H \neq \emptyset$, it is called a *face*. The face is *proper* if $H \not\subset S$. For faces τ we write $\tau \leq \sigma$, and we write $\tau < \sigma$ if τ is a proper face of σ. In the latter case we say that σ is a (*proper*) *coface* of τ.

The *boundary* of a simplex σ is the collection of all its proper faces, and the complement of the boundary is called the *interior* of σ.

Definition 3.5. A *simplicial complex* K is a finite collection of simplices such that for any simplex $\sigma \in K$, if $\tau \leq \sigma$ then $\tau \in K$, and for any two simplices $\sigma, \sigma_0 \in K$, their intersection $\sigma \cap \sigma_0$ is either empty or a face of both σ and σ_0.

3.2.2 Abstract Complexes

Next we are going to define an abstraction of a simplicial complex, eliminating coordinates from the definition. This will simplify computations. This abstraction will still follow the intersection properties of simplicial complexes in Definition 3.5,

Definition 3.6. An *abstract simplicial complex* is a finite collection of non-empty sets A in which if $\alpha \in A$ and $\beta \subseteq \alpha$, then $\beta \in A$.

The elements of α are referred to as *vertices* and the elements of A are referred to as simplices. The dimension of an abstract complex is the maximum dimension of its simplices,

$$\dim(A) = \max_{\alpha \in A}(\dim(\alpha)) = \max_{\alpha \in A}(\operatorname{card} \alpha - 1).$$

Definition 3.7. A *subcomplex* of B is an abstract simplicial complex $A \subseteq B$.

Definition 3.8. A *filtration* is sequence of abstract simplicial complexes $\{K_i\}_{i \in I}$ such that $K_0 \subseteq \cdots \subseteq K_n$.

Abstract simplicial complexes are a useful tool to analyse the properties of topological spaces.

Transforming a geometric simplicial complex into an abstract simplicial complex is a straightforward process: it involves replacing each vertex's coordinates with a unique label. However, the inverse process, converting an abstract simplicial complex into a geometric one, is somewhat more challenging. This requires assigning coordinates to the vertices in a way that satisfies the conditions for a geometric simplicial complex. If this is possible, the resulting geometric simplicial complex is referred to as a *geometric realization*.

3.2.3 Cubical Complexes

Digital images have a cubical structure, since two dimensional images are composed of pixels and three dimensional images are made of voxels. We are going to present an example of an abstract simplicial complex which is useful to obtain topological descriptors of pixelized data, formed by glueing together cubes of various dimensions, a *cubical complex*.

Definition 3.9. We define an *elementary interval* as a closed interval $I_e \subset \mathbb{R}$ with $I_e = [m, m+1]$, $m \in \mathbb{N}$.

Definition 3.10. We define an *n-cube* $\sigma \subset \mathbb{R}^n$ as the product of n elementary intervals $C = \prod_{i=1}^{n} I_{e_i}$. The number of non-degenerate elementary intervals n is defined as the *dimension* of the n-cube.

We will refer to the 3-cubes as *voxels*, the 2-cubes as *squares*, the 1-cubes as *edges* and the 0-cubes as *vertices*. Cubical complexes C composed of elements of dimension 2 (the pixels) will be useful to represent volumes/images V.

Being an abstract simplicial complex, the *dimension* of a cubical complex is the maximum dimension of its n-cubes.

Definition 3.11. A *cubical complex* C is a collection of n-cubes so that if $c \in C$ and $c' \subseteq c$, then $c' \in C$.

There is an example of a filtration that is frequently used while working with cubical complexes:

Definition 3.12. Let $f\colon K_0 \to \mathbb{R}$ be a function defined on the vertices of a simplicial complex K. We define define the *lower-star filtration* of K as the extension of f to every simplex $\sigma \in K$, namely $F(f)(\sigma) = \max_{v \in \sigma} f(v)$.

3.3 Homology

Definition 3.13. [48, Section 2.4.2] Let K be a simplicial k-complex with m_p number of p-simplices, $0 \leq p \leq k$. A *p-chain* c in K is a formal sum of p-simplices added with some coefficients, that is, $c = \sum_{i=1}^{m_p} \alpha_i \sigma_i$ where σ_i are p-simplices and α_i are coefficients. Two p-chains $c = \sum \alpha_i \sigma_i$ and $c' = \sum \alpha_i' \sigma_i$ can be added to obtain another p-chain

$$c + c' = \sum_{i=1}^{m_p} \left(\alpha_i + \alpha_i'\right) \sigma_i.$$

Definition 3.14. A *chain group* or *group of p-chains* $(C_p, +)$ is the abelian group (or vector space, if coefficients in a field are used) containing all p-chains.

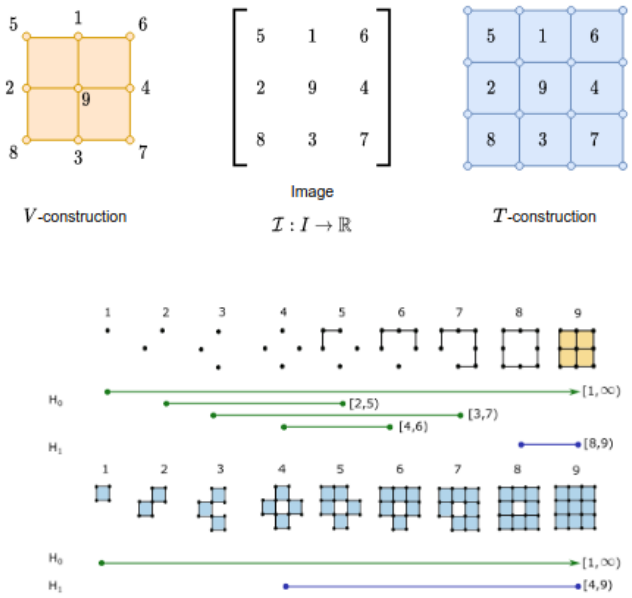


Figure 3.1: Example of a filtration of a cubical complex obtained from a 3×3 image. Image source: [55]

Notation 3.1. A group of p-chains can be represented as $C_p = C_p(K)$, with K a simplicial complex.

We will refer to the $(p-1)$-chain consisting of the sum of the $(p-1)$-faces of a p-simplex σ as the *boundary* of σ.

Definition 3.15. The *p-th boundary operator* is a linear map $\partial_p \colon C_p(K) \to C_{p-1}(K)$ sending every p-simplex $\sigma = [u_0, \ldots, u_n]$ to

$$\partial_p \sigma = \sum_{j=0}^{p} (-1)^j [u_0, \ldots, \hat{u}_j, \ldots, u_n],$$

where $\hat{u}_j$ marks the fact that u_j is omitted.

We will refer to a boundary operator as a *boundary homomorphism*, since it commutes with addition:

$$\partial_p(c + c') = \partial_p \left(\sum a_i \sigma_i + \sum b_i \sigma_i \right) = \partial_p \left(\sum a_i \sigma_i \right) + \partial_p \left(\sum b_i \sigma_i \right) = \partial_p(c) + \partial_p(c').$$

Definition 3.16. A *chain complex* is a sequence of abelian groups together with boundary homomorphisms,

$$\ldots \xrightarrow{\partial_{p+2}} C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} C_{p-2} \xrightarrow{\partial_{p-2}} \ldots$$

such that $\partial_p \circ \partial_{p+1} = 0$ for all p.

Proposition 3.17. [47, Section IV.1] *Fundamental lemma of homology:* $\partial_p \partial_{p+1} d = 0$ for every integer p and every $(p+1)$-chain d.

Proof. [47, Section IV.1] We just need to show that $\partial_p \partial_{p+1} \tau = 0$ for a $(p+1)$-simplex τ. The boundary $\partial_{p+1}\tau$ consists of all p-faces of τ. Every $(p-1)$-face of τ belongs to exactly two p-faces, so $\partial_p \left(\partial_{p+1}\tau\right) = 0$. □

Definition 3.18. A *p-cycle* is a p-chain c with $\partial c = 0$.

Since the boundary operator commutes with addition, all p-cycles are contained in a group that is called the *p-cycle group* $Z_p = Z_p(K)$. It is a subgroup of the group of p-chains. With regard to Z_p, the group of p-cycles Z_p is the kernel of the p-boundary homomorphism $\ker \partial_p = Z_p$, since C_p is sent to the zero of C_{p-1}. Being a subgroup of an abelian group, it is also abelian.

Definition 3.19. A *p-boundary* is a p-chain c with $\partial d = c$, where d is a $(p+1)$-chain.

We can also define the group obtained by applying the boundary homomorphism on the p-chains $\partial_p(C_p)$, the *p-boundary group* $B_p = B_p(K)$. It contains all p-boundaries. We have that $\operatorname{im} \partial_p = \partial_p(C_p) = B_p(K)$. Using Proposition 3.17, it follows that $\partial_{p-1} B_{p-1} = \partial_{p-1}\partial_p(C_p) = 0$ for $p > 0$. Hence, $B_{p-1} \subseteq Z_{p-1}$.

Definition 3.20. [47, Section IV.1] The *p-th homology group* of a simplicial complex K is the p-th cycle group modulo the p-th boundary group, $H_p(K) = Z_p(K)/B_p(K)$. The p-th Betti number is the rank of this group, $\beta_p = \operatorname{rank} H_p(K)$.

Every element of H_p is obtained by adding a p-cycle c to the boundary group, $c + B_p$. The resulting elements of the sum of c and B_p belong to the class $[c]$ called the *homology class of* c. We will say that all elements belonging to $[c]$ are *homologous*. We will denote two homologous elements c, c' as $c \sim c'$.

The rank of a p-homology group indicates the number of p-dimensional features: the rank of a 0-dimensional homology group counts the number of connected components, while the rank of a 1-dimensional homology group counts the number of holes, while the 2-dimensional features are voids.

Since the rank is a discrete quantity, we cannot use it in optimization. Simply comparing Betti numbers in a topological error function would result in a discrete-valued function. We need to define a way to track the evolution of homological features over a filtration.

3.4 Persistent Homology

3.4.1 Persistence Modules

We are going to present one of the key components of persistent homology: the *persistence modules*. They are algebraic structures that can be assigned to a simplicial complex (or more generally, a continuous filtration) in order to track the birth and death of topological features.

Definition 3.21. Let $\mathbb{K}$ be a field. We define a *persistence module* $\mathbb{V}$ over $\mathbb{K}$ as an indexed family of $\mathbb{K}$-vector spaces $\{V_r\}_{r\geq 0}$ and a doubly-indexed family of linear maps

$$\{v_s^t \colon V_s \to V_t \mid s \leq t\}$$

satisfying the composition law

$$v_s^t \circ v_r^s = v_r^t \qquad \forall r \leq s \leq t,$$

with v_t^t being the identity map on V_t for all t .

Definition 3.22. We define a *homomorphism* $\phi \colon \mathbb{V} \to \mathbb{W}$ between two persistence modules $\mathbb{V}$ and $\mathbb{W}$ as a collection of linear maps

$$\{\phi \colon u_t \to v_t \mid t \in T\}$$

such that

$$\begin{array}{ccc} U_s & \xrightarrow{u_t^s} & U_t \\ \downarrow{\scriptstyle \phi_s} & & \downarrow{\scriptstyle \phi_t} \\ V_s & \xrightarrow{v_t^s} & V_t \end{array}$$

commutes for all $s \leq t$.

The collection of persistence modules has the structure of a category, since it is a collection of objects with a collection of morphisms, a composition operation and an identity morphism for every object. We will denote it as *Pers*.

Definition 3.23. Let J be an interval and $\mathbb{K}$ a field. Then $\mathbb{I}_J$ is defined as the $\mathbb{K}$-persistence module with spaces

$$\mathbb{I}_J = \begin{cases} \mathbb{K} & \text{if } t \in J, \\ 0 & \text{if } \textit{otherwise}, \end{cases}$$

and maps

$$i_s^t = \begin{cases} 1 & \text{if } s, t \in J, \\ 0 & \text{if } \textit{otherwise}. \end{cases}$$

Definition 3.24. Let $\mathbb{U}$ and $\mathbb{V}$ be persistence modules. We define the *direct sum* $\mathbb{W} = \mathbb{U} \bigoplus \mathbb{V}$ of $\mathbb{U}$ and $\mathbb{V}$ as

$$W_t = U_t \bigoplus V_t, \quad w_t^s = u_s^t \bigoplus v_s^t.$$

Definition 3.25. [54, Section 1.4] A persistence module $\mathbb{W}$ is *indecomposable* if the only decompositions $\mathbb{W} = \mathbb{U} \bigoplus \mathbb{V}$ are $\mathbb{W} \bigoplus 0$ and $0 \bigoplus \mathbb{W}$.

Proposition 3.26. [54, Proposition 1.2] Interval modules are indecomposable.

Theorem 3.27. [54, Theorem 1.3] Suppose that a persistence module $\mathbb{V}$ over $\mathbb{K} = \mathbb{R}$ can be expressed as a direct sum of interval modules in two different ways:

$$\mathbb{V} = \bigoplus_{l \in L} \mathbb{I}^{J_l} = \bigoplus_{m \in M} \mathbb{I}^{K_m}.$$

Then there is a bijection $\sigma \colon L \to M$ such that $J_l = K_{\sigma(l)}$ for all l.

We derive from this theorem that, if existing, the decomposition of a given persistence module $\mathbb{V}$ as a direct sum of interval modules $\mathbb{V} = \bigoplus_{l \in L} \mathbb{I}^{J_l}$ is unique.

We call the finite multiset J the *barcode* of $\mathbb{V}$. We can also represent the barcode set with a finite multiset B of points of the form $(\inf J, \sup J)$, to which we add the multiset Δ_∞ having multiple copies of the set $\Delta = \{(b, b) \mid b \in \mathbb{R}\}$.

Definition 3.28. [56, Section 2.2.1] A *persistence diagram* is the union $B \cup \Delta^\infty$ of a finite multiset of elements in $\mathbb{R} \times \hat{\mathbb{R}}$, where $\hat{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$, with countably many copies of the diagonal Δ. The set of persistence diagrams is denoted by Bar_Δ.

Definition 3.29. [56, Section 2.2.1] Given two barcodes $D, D' \in \mathrm{Bar}$, viewed as multisets, a *matching* is a bijection $\gamma \colon D \to D'$. The *cost* of γ is the quantity

$$c(\gamma) = \sup_{x \in D} \|x - \gamma(x)\|_\infty \in \mathbb{R} \cup \{+\infty\}.$$

The set of all matchings between D and D' is denoted by $\Gamma(D, D')$.

Definition 3.30. [56, Section 2.2.1] The *bottleneck distance* between two barcodes $D, D' \in \mathrm{Bar}_\Delta$ is defined as

$$d_\infty(D, D') = \inf_{\gamma \in \Gamma(D, D')} c(\gamma).$$

Thereby the space of persistence diagrams can be equipped with a metric structure, opening the possibility of doing metric analysis.

3.4.2 Persistence Diagrams for Homology Groups

Let $\{K_i\}_{i\in I}$ be a filtration. Thus, $K_i \subseteq K_j$ for all $i, j \in I$ with $i \leq j$. Hence K_i is a subcomplex of K_j and has a well defined inclusion map $f^{i,j}\colon K_i \to K_j$. Since homology can be applied to each of the simplicial complexes, the inclusion maps $K_i \to K_j$ induce $\mathbb{Z}_2$-linear maps $f^{i,j}\colon H_p(K_i) \to H_p(K_j)$ for all $i, j \in I$ with $i \leq j$. It follows that

$$f_p^{k,j} \circ f_p^{i,k} = f_p^{i,j} \qquad \forall i \leq k \leq j.$$

Using Definition 3.21 we infer that the homology groups of a filtered simplicial complex together with the induced linear maps form a persistence module. We can then formulate a definition of persistence diagrams for filtrations created by linear maps. First we need some previous definitions.

Definition 3.31. Let $\{K_i\}_{i\in I}$ be a filtration. The *p-th persistent homology* of $\{K_i\}_{i\in I}$ is the pair

$$(\{H_p(K_i)\}_{1\leq i\leq l}, \{f_{i,j}\}_{1\leq i\leq j\leq l}) \quad \forall i, j \in I, \quad i \leq j.$$

From this sequence of complexes we obtain a sequence of homology groups connected by homomorphisms,

$$0 = H_p(K_0) \to H_p(K_1) \to \cdots \to H_p(K_n) = H_p(K).$$

Definition 3.32. The *p-th persistent homology groups* are the images of the homomorphisms induced by inclusion, $H_p^{i,j} = \text{im } f_p^{i,j}$, for $0 \leq i \leq j \leq n$.

Definition 3.33. The *p-th persistent Betti numbers* are the ranks of the p-th persistent homology groups, $\beta_p^{i,j} = \text{rank } (H_p^{i,j})$.

Observation 3.1. It is worth mentioning that $H_p^{i,i} = H_p(K_i)$.

According to [47, Section VII.1], we can express the p-th persistent homology group as

$$H_p^{i,j} = Z_p(K_i)/(B_p(K_j) \cap Z_p(K_i)).$$

This equality works for every dimension p and for every pair $i \leq j$.

Persistent homology groups are algebraic structures that measure the survival of homology classes throughout a sequence. Given that scenario, we say that a class γ in $H_p(K_i)$ is *born* at K_i if $\gamma \in H_p(K_i)$ and $\gamma \in H_p^{i-1,j}$. Likewise, we say that γ *dies* at K_j if $f_p^{i,j-1}(\gamma) \in H_p^{i-1,j-1}$ but $f_p^{i,j}(\gamma) \notin H_p^{i-1,j}$. We will represent the birth of γ at K_i as $b(\gamma) = i$ and the death of γ at K_j as $d(\gamma) = j$. We will refer to the difference between the birth and death of γ as the *persistence of* γ, $\text{pers}(\gamma) = d(\gamma) - b(\gamma)$.

Definition 3.34. We define the *p-th persistence diagram* of the filtration $\{K_i\}_{i\in I}$ as a finite multiset of points $(b(\gamma), d(\gamma))$ in the extended real plane $\hat{\mathbb{R}}^2 = (\mathbb{R} \cup \{\pm\infty\})^2$. The multiplicity of the p-dimensional classes will be represented with $\mu_p^{i,j}$ and is defined as

$$\mu_p^{i,j} \colon = (\beta_p^{i,j-1} - \beta_p^{i,j}) - (\beta_p^{i-1,j-1} - \beta_p^{i-1,j}), \quad \forall i < j, \quad \forall p.$$

Lemma 3.35. [47, Section VII.1] *Fundamental Lemma of Persistent Homology*. Let $\emptyset = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n = K$ be a filtration. For every pair of indices $0 \leq k \leq l \leq n$ and every dimension p, the p-th persistent Betti number is $\beta_p^{k,l} = \sum_{i\leq k} \sum_{j>l} \mu_p^{i,j}$.

3.4.3 Monotonic Functions

We want to extend the definition of persistent homology to ensure the existence of well-defined filtrations on simplicial complexes. To do so we will use a special type of functions called *monotonic* functions that wil allow us to create filtrations. We will redefine the definitions of births and deaths of a homology class applied to monotonic filtrations, since it presents some differences.

Considering a simplicial complex K and a real-valued function $f \colon K \to \mathbb{R}$, we say that f is *monotonic* if $f(\sigma) \leq f(\tau)$ when $\sigma \subseteq \tau$. From the monotinicity we derive that $K(a) = f^{-1}(-\infty, a]$ is a subcomplex of K, for every $a \in \mathbb{R}$. We call $K(a)$ the *sublevel set* of the point a.

Accordingly, given a set of real values $\{a_i\}_{i\in I} \subseteq \mathbb{R}$, with $a_i \leq a_j$ for $i \leq j$, we define the *monotonic filtration* of f respect to a_i as the filtration

$$\ldots \subseteq K(a_{i-1} \subseteq K(a_i) \subseteq K(a_{i+1}) \subseteq \ldots$$

Definition 3.36. Let K be a simplicial complex, $\{a_i\}_{i\in I} \subseteq \mathbb{R}$ a sequence of real values and $f \colon K \to \mathbb{R}$ a monotonic function. Let $\{K_i\}_{i\in I}$ be the monotonic filtration defined by f and $\{a_i\}_{i\in I}$. The *birth* of a homology class γ is defined as $b(\gamma) = f(K_i)$ if γ is born at K_i. Likewise, we define the *death* of a homology class γ as $d(\gamma) = f(K_j)$ in case γ dies at K_j, or $d(\gamma) = \infty$, in case γ does not die.

Using the previous definition we can adapt the definition of the persistence diagram for monotonic functions:

Definition 3.37. Let $\{a_i\}_{i\in I} \subseteq \mathbb{R}$ be a sequence of real values, $f \colon K \to \mathbb{R}$ a monotonic function and $\{K_i\}_{i\in I}$ the monotonic filtration defined by f and $\{a_i\}_{i\in I}$. We define the *p-th persistence diagram* of $\{K_i\}_{i\in I}$ as the multiset of points in $\hat{\mathbb{R}}^2$ denoted as $\mathrm{Dgm}_p(f)$ such that $\mathrm{Dgm}_p(f) = \{(f(K_i), f(K_j)) \,:\, (i,j) \in I\}$.

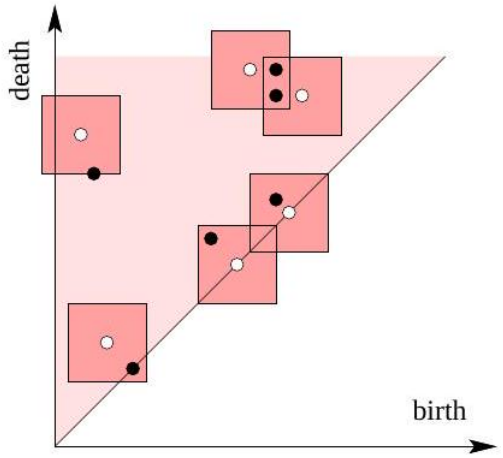


Figure 3.2: Let D and $\hat{D}$ be two different persistent diagrams, where D contains the white points and and $\hat{D}$ contains the black points. We can draw squares twice the bottleneck distance between D and and $\hat{D}$, due to the definition of the bottleneck distance. Image source: [47]

The coordinates of the elements in the persistence diagram $(b,d) \in \mathrm{Dgm}_p(f)$ are the birth and death associated with a specific homology class γ, respectively.

Persistence diagrams are used to extract features from monotonic functions. They are a useful tool to compare the topological structure of different functions and measure their similarity. However, to do so we still need to define a metric returning the distance between persistence diagrams.

As in [47, Section VII.2], we will consider that the diagram consists of finitely many points above the diagonal and infinitely many points on the diagonal. The latter points will be useful to simplify the definition.

Definition 3.38. Let X, Y be persistence diagrams. Let $S = \{\eta \colon X \to Y\}$ be a set of bijections between them. We define the *bottleneck distance* between X and Y as

$$W_\infty(X,Y) = \inf_{\eta \colon X \to Y} \sup_{x \in X} \|x - \eta(x)\|_\infty.$$

The Stability Theorem for Filtrations proves stability of the bottleneck distance:

Definition 3.39. [47, Section VII.2] Stability Theorem for Filtations. Let K be a simplicial complex and let $f, g : K \to \mathbb{R}$ be monotonic functions. For each dimension p, the bottleneck distance between the diagrams $X = \mathrm{Dgm}_p(f)$ and $Y = \mathrm{Dgm}_p(g)$ is bounded above by the L_∞-distance between the given functions, $W_\infty(X,Y) \leq \|f - g\|_\infty$.

3.5 Differentiability

In [56] a differential framework is provided to prove differentiability of the composition

$$\mathcal{L}\colon \mathcal{M} \longrightarrow \mathrm{Bar}_\Delta \longrightarrow \mathbb{R}.$$

The theory focuses on simplicial complexes and needs to be adapted, for this work, to cubical complexes determined using lower-star filtration.

We wil analyse the differentiability of the function $B_p\colon \mathcal{M} \to \mathrm{Bar}_\Delta$. Specifically, of the composition of two functions: first the parametrization F given the lower-filter function obtained from a parametrization $F_0\colon \mathcal{M} \to \mathbb{R}^{K^0}$ on the vertices K_0 of K. The second function is the persistent homology operator Dgm_p:

$$B_p\colon \mathcal{M} \xrightarrow{F} \mathbb{R}^K \xrightarrow{\mathrm{Dgm}_p} \mathrm{Bar}_\Delta.$$

We next provide two differentiability results for cubical complexes. They will allow us to study differentiability of images represented as 3-cubes.

Proposition 3.40. [56, Proposition 5.2] Let $F_0 : \mathcal{M} \to \mathbb{R}^{K_0}$ be a C^r parametrization of filter functions on the vertices of K. The induced parametrization $F\colon \mathcal{M} \to \mathbb{R}^K$ is C^r at each $\theta \notin \mathrm{Sing}\,(F_0)$, where $\mathrm{Sing}\,(F_0)$ is the boundary of the set

$$\left\{\theta \in \mathcal{M} \mid \exists\left(v, v'\right) \in K_0,\, F_0(\theta)(v) = F_0(\theta)\left(v'\right)\right\}.$$

Specifically, for every $\theta \notin \mathrm{Sing}\,(F_0)$, letting

$$\bar{v} : \sigma \in K \mapsto \underset{v \text{ vertex in } \sigma}{\operatorname{argmax}}\, F_0(\theta)(v) \in K_0$$

by breaking ties wherever necessary, there is an open neighborhood U of θ such that $F\left(\theta'\right)(\sigma) = F_0\left(\theta'\right)\left(\bar{v}(\sigma)\right)$ for every $\theta' \in U$ and $\sigma \in K$, from which follows that F is C^r at θ.

We use Proposition 3.40 with F the parametrization given the lower-filtration.

Corollary 3.41. [56, Corollary 5.4] For any C^r parametrization $F_0 : \mathcal{M} \to \mathbb{R}^{K_0}$ on the vertices of K, the induced barcode valued map $B_p\colon \theta \in \mathcal{M} \mapsto \mathrm{Dgm}_p(F(\theta))$ is r-differentiable outside $\mathrm{Sing}\,(F_0)$. Moreover, at $\theta \in \mathcal{M} \setminus \mathrm{Sing}\,(F_0)$, for any barcode template $\left(P_p, U_p\right)$ of $F(\theta)$ and any choice of ordering $\left(\sigma_1, \sigma_1'\right), \ldots, \left(\sigma_m, \sigma_m'\right)$, $\tau_1, \ldots, \tau_n$ of $\left(P_p, U_p\right)$, the map $\bar{B}_p\colon \mathcal{M} \to \mathbb{R}^m \times \mathbb{R}^n$ defined by

$$\bar{B}_p : \theta' \longmapsto \left[\left(F_0\left(\theta'\right)\left(\bar{v}\left(\sigma_i\right)\right), F_0\left(\theta'\right)\left(\bar{v}\left(\sigma_i'\right)\right)\right)_{i-1}^{m}, \left(F_0\left(\theta'\right)\left(\bar{v}\left(\sigma_j'\right)\right)\right)_{j-1}^{n}\right]$$

is a local C^r lift of B_P around θ. The corresponding differential for B_P at θ is:

$$d_{\theta,\tilde{B}_p} B_p(*) = \left[\left(d_\theta F_0 \left(^* \right) \left(\bar{v} \left(\sigma_i \right) \right), d_\theta F_0(*) \left(\bar{v} \left(\sigma_i' \right) \right) \right)_{i=1}^m, \left(d_\theta F_0 \left(^* \right) \left(\bar{v} \left(\sigma_j' \right) \right) \right)_{j=1}^n \right].$$

We represent images as vectors $I \in [0,1]^{n \times m}$, with n the height (in pixels) and m the width (in pixels). The pixels are represented as coordinates of I. Let K_0 be a cubical complex given by $n \times m$ images considering each pixel a vertex. We will restrict the images to the smooth manifold $(0,1)^{n \times m}$ and consider the vertex parametrization $F_0 \colon (0,1)^{n \times m} \to \mathbb{R}^{K_0}$.

Since $F_0 = Id$, F_0 is C^∞ and we can apply Corollary 3.41.

Focusing on maps $V \colon \mathrm{Bar}_\Delta \to \mathcal{N}$, we have differentiability results for different examples of functions.

Definition 3.42. [56, Definition 3.10] Let $V \colon \mathrm{Bar}_\Delta \to \mathcal{N}$ be a map on barcodes. Let $D \in \mathrm{Bar}$ and $r \in \mathbb{N} \cup \{+\infty\}$. V is said to be r-differentiable at D if, for all integers m, n and all vectors $\tilde{D} \in \mathbb{R}^{2m} \times \mathbb{R}^n$ such that $Q_{m,n}(\tilde{D}) = D$, the map $V \circ Q_{m,n} : \mathbb{R}^{2m} \times \mathbb{R}^n \to \mathcal{N}$ is C^r on an open neighborhood of $\tilde{D}$.

Following Definition 3.42, for every m, n the map $V \circ Q_{m,n}$ exists and is unique. The map will be differentiable if it is differentiable for every $\hat{D}$, for every m, n.

Therefore, the map V is ∞-differentiable for every $D \in \mathrm{Bar}_\Delta$.

Let $V \colon \mathrm{Bar}_\Delta \to \mathbb{R}$, with $V(D) = \sum_{(b,d), b<d<\infty} d - b$ for every $D \in \mathrm{Bar}_\Delta$. Let $D \in \mathrm{Bar}_\Delta$ and its inverse image $\hat{D} \in \mathbb{R}^{2m+n}$. Finally, we have that

$$V \circ Q_{m,n} \colon (b_1, d_1, \ldots, b_m, d_m, v_1, \ldots, v_n) \in \mathbb{R}^{2m+n} \mapsto \sum_{i=1}^m d_i - b_i \in \mathbb{R}$$

and $V \circ Q_{m,n}$ is C^∞.

There is also a proof of the differentiability of the bottleneck distance:

Proposition 3.43. [56, Proposition 7.9] For any $D \in \mathrm{Bar}_\Delta$,

(i) d_{Δ^∞} is ∞-differentiable at D, and

(ii) for any $m \in \mathbb{N}$ and $\tilde{D} \in \mathbb{R}^{2m} \times \mathbb{R}^0$ such that $Q_{m,0}(\tilde{D}) = D$, there are exactly two non-zero components in the gradient $\nabla_{\tilde{D}} \left(d_{\Delta^\infty} \circ Q_{m,0} \right)$, one with value $\frac{1}{2}$ and the other with value $-\frac{1}{2}$.

Chapter 4

Experiments

4.1 Environment

The project was written in the Python programming language, version 3.7, which is the default version in Google Collab, the environment used. The libraries that have been used in the project are the Tensorflow framework [68] and the library Gudhi [69], specifically the Python modules.

Tensorflow is a machine learning framework that allows to represent the computation and state of deep learning algorithms during the computation.

On the other hand, the Gudhi library (Geometric Understanding in Higher Dimensions) contains implementations of algorithms and data structures for computational topology.

The high-level Keras API [70], created to simplify the implementation of neural networks and integrated in Tensorflow, was also used in the project.

4.2 Dataset

The dataset focus on the right ventricle segmentation from CMR and it was obtained from the second edition of the M&Ms Challenge (M&Ms-2). It is a dataset for scientific benchmarking to promote the development of deep learning models in cardiac image segmentation.

Three clinical centers from Spain, contributed to the challenge providing 360 cardiac magnetic resonance (CMR) studies for analysis in total.

The subjects included in the study were selected from various cardiovascular disease groups and healthy volunteers. The list of pathologies suffered from the subjects is Dilated Left Ventricle, Dilated Right Ventricle, Hypertrophic cardiomyopathy, Arrhythmogenic cardiomyopathy, Tetrology of Fallot, Inter-atrial communication and Tricuspid regurgitation.

Different scanners were used for both axis, assuring heterogeneity of the data. Each CMR study was manually annotated by an expert clinician, and the annotations were reviewed and finalized by four researchers to reduce variability.

As per the clinical protocol, the short-axis and long-axis 4-chamber views were labeled at the end-diastolic (ED) and end-systolic (ES) phases, which are the phases used to calculate important clinical biomarkers for cardiac diagnosis and monitoring. Three main regions were evaluated: the cavities of the left and right ventricles and the myocardium of the left ventricle, represented by channels 0, 1, and 2, respectively.

4.3 Preprocessing

Due to the lack of an adequated equipment, only a portion of the data published in the second M&Ms challenge was considered. In total, 720 long-axis images.

Each of the two types of CMR images, long-axis and short-axis, was considered independently, since the mixing of both datasets was out of the scope of this work. We focused in the long-axis images in this work.

The input images were rescaled from values in the interval [0,255] to values in the interval [0,1]. Since not all the images had the same resolution, they were reshaped to a common shape. An input shape of 128×128 was selected. We did not pick a bigger shape because the model performance depends on the input images. The information contained within the images was enough to compute the segmentation. Also, a smaller shape could have caused the loss of information.

Regarding the labels, they also were reshaped to a 128×128 shape. Four channels were added, each of them containing the segmentation of an atlas: the channels correspond to the segmentations of the left and right ventricle cavities, the left ventricle myocardium and the background, respectively. The latter was considered for its topological properties.

The final shape of the input data is $128 \times 128 \times 1$ and the shape of the labels and output of the model is $128 \times 128 \times 4$.

Based on the works [76, 77], the subset of tested images for the long-axis consisted of 30% of the total: $616 * 0.3 \approx 216$ images. The remaining set of images was partitioned into 400 images of training data and 104 images of validation data.

4.4 Metrics

When it comes to deep learning segmentation, it's essential to keep an eye on the accuracy and loss of our model. These two metrics play a crucial role in

determining how well our model is performing. To measure the capabilities of our model, we have used the following accuracy metrics: *Categorical Cross Entropy Accuracy*, *Intersection-Over-Union* and *Dice Coefficient*. These methods are some of the most commonly used accuracy functions [72, 73].

The Categorical Cross Entropy Accuracy is defined as

$$CCE(y, \hat{y}) = -\frac{1}{n}\sum_{i=0}^{n-1} y_i * \log(\hat{y}_i),$$

with n the number of samples.

The Intersection-Over-Union is expressed as

$$IoU = \frac{A_I}{A_{GT} + A_{PB} - A_I},$$

with $A_I :=$ Area of intersection, $A_{GT} :=$ Ground Truth Area and $A_{PB} :=$ Predicted Box Area.

The Dice Coefficient is calculated as

$$DSC = \frac{2|X \bigcap Y|}{|X| + |Y|},$$

given two sets X and Y.

The selected optimizer is Adam [71], an efficient extension of the Stochastic Gradient Descent algorithm. The learning rate for the experiments is $\mathrm{lr} = 0.001$.

A weighted combination of the local losses Dice Loss and Pixel-Wise Loss was complemented with topological regularizers:

$$\mathcal{L} = \mathcal{L}_{Dice}(Y, \hat{Y}) + \alpha\mathcal{L}_{Pixel-Wise}(Y, \hat{Y}) + \beta\mathcal{L}_{Regularizer}(Y, \hat{Y}).$$

After some preliminary tests, the coefficients α, β were set to 2.0 and $n_l =$ number of classes of the label, respectively.

The topological loss $\mathcal{L}_{Regularizer}$ is the result of adding the topological loss for every channel. The first topological loss used is the *degree-p total persistence*:

$$\mathrm{Pers}_p(D) \colon = \sum_{(\tau_i, \tau_j \in D)} |\mathrm{pers}(\tau_i, \tau_j)^p|,$$

with D a persistence diagram and $\mathrm{pers}(\tau_i, \tau_j) \colon = |\tau_i, \tau_j|$, given a tuple $(\tau_i, \tau_j) \in D$. For this dataset we have considered $p = 1$, so for the sake of clarity we call this distance the *total persistence*. The other topological loss is the bottleneck distance.

The bottleneck distance is applied to a set of channels $i \in I_1$ for a set of dimensions $d \in I_2$, with $I_1 \subset \{0, 1, 2, 3\}$ and $I_2 \subset \{0, 1\}$. The related loss is

$$\mathrm{Bottleneck}(I_1, I_2, \mathrm{Image}_1, \mathrm{Image}_2) = \sum_{c \in I_1, d \in I_2} \alpha_{c,d} W_\infty(D_1^{c,d}, D_2^{c,d}),$$

with Image_1 the label and Image_2 the prediction, $\alpha_{c,d}$ the weight associated with the channel c and the dimension d and $D_1^{c,d}, D_2^{c,d}$, the persistence diagrams associated with the label and the prediction for c, d. The value of the set of weights is 0.01 for the channels 0,1,3 and 0.025 for the channel 2.

It is important to note that the relationship between accuracy and loss is not inverse, which means that a model can have high accuracy but still have a certain level of uncertainty. This is due to other factors such as overfitting, underfitting, and data distribution that can also affect the model's performance. Additionally, the threshold used to classify predictions can impact accuracy, where a higher threshold would result in higher accuracy but higher uncertainty. Some researchers argue that accuracy alone may not provide a complete picture of the performance of a model, particularly in cases where the dataset is imbalanced [74]. Therefore, it is important to consider multiple factors, not just accuracy and loss [57, 58]. This way, we will only assess the accuracy values from the results in conjunction with a visual examination.

4.5 Training

A custom training loop was incorporated to the model to be able to perform the necessary computations to obtain the persistence diagrams and apply the selected regularizers.

To prevent overfitting, an *early stopping* callback was added to stop the computations of the model once the learning stage finished. Early stopping is an optimization technique that stops the training once a model starts overfitting. This is achieved keeping a record of the losses for every epoch and when the loss stops decreasing for a number of preselected epochs, the training stops.

The selected early stopping value was 20 epochs. The loss with a topological regularizer was checked every iteration and weights of the model were saved in case the loss reached its minimum. When the training finished the weights of the epochs with the best loss were loaded and some predictions were made to do a visual check of the precision of the method.

We have used a batch size of 50 images, so the number of iterations to arrive to an epoch is 8. We also fixed a maximum of 1000 epochs, an arbitrary number but hard to reach, to avoid infinite loops.

Unlike the training loop, the testing and validation loops had a fixed loss of

$$\mathcal{L} = \mathcal{L}_{Dice}(Y, \hat{Y}) + \alpha \mathcal{L}_{Pixel-Wise}(Y, \hat{Y}),$$

with the value of α remaining the same as in the training loop.

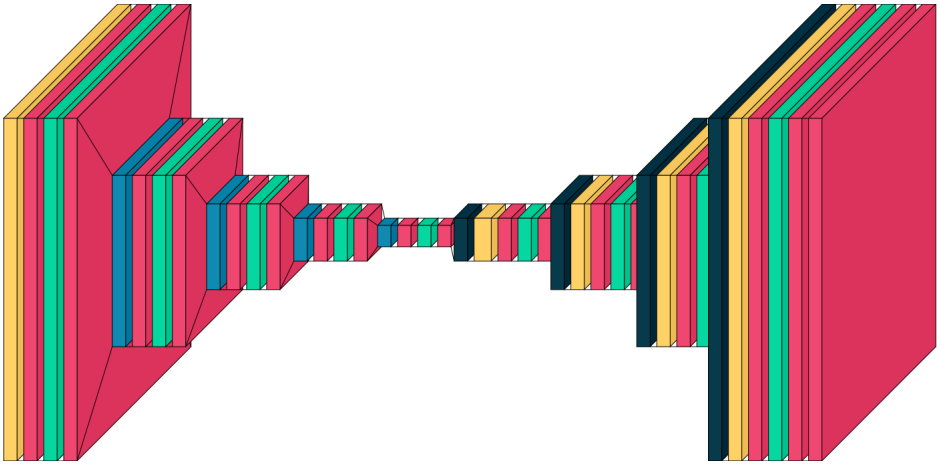


Figure 4.1: 3-D visualization of the model's downsampling and upsampling paths. Source of the library used to create the image: [82].

4.6 Model

The model that has been used is a U-Net Neural Network [16]. This model was created to use the data available more efficiently, by capturing more context, and to reduce computation time [16]. As we explained in a previous section, it is the predefined model in the image segmentation state of the art.

The model follows the typical arquitechture of a convolutional network. It consists of a contraction path and a expansion path. The first is composed of convolutional and max pooling layers. The second consist of concatenations of convolutional layers from the contraction path and transposed convolutional layers. As the authors explained: "*The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization*".

The input data starts with the shape $128 \times 128 \times 1$ and is downsampled through the continuous applications of convolution layers with a 3×3 filter, dropouts of the 10% of the neurons and max pooling layers with a 2×2 filter with stride 2. Then, the data reaches the shape $8 \times 8 \times 160$ and enters in the expansive path.

In the expansive path the data is upsampled with convolutional transpose layers with a 3×3 filter and 3×3 convolutions. The dropout layers located in this path allow the model to achieve more generalization by dropping a 20% of the neurons. The last layer is located at the end of the upsampling path and its a 1×1 convolution.

All convolutional and convolutional transpose layers have a RELU activation function. Through the last layer the output reaches the shape $128 \times 128 \times 4$.

A summary of the networks architecture can be found in a visual representa-

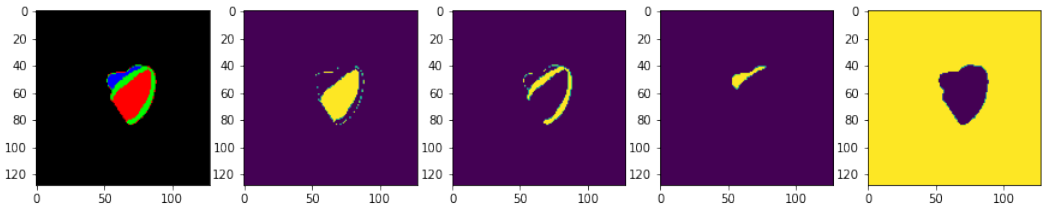


Figure 4.2: A CMR image segmentation next to every one of its four channels.

tion in Figure 4.1 and in Figure 4.8 .

4.7 Results

We will consider the terms GL := Geometrically local Loss (combination of Pixel-Wise Loss and Dice Loss), PWL := Pixel-Wise Loss, DL := Dice Loss, PL := Total persistence Loss, BL := Bottleneck Loss, MeanIoU := Intersection-Over-Union Accuracy, CE := Cross-Entropy Accuracy and DC := Dice Coefficient Accuracy, in the subsequent tables.

We applied several methods to compare the resulting accuracies and identify the best technique to implement CMR segmentation. To reduce the computation time, the persistence diagrams of the four channels were pre-computed before the training started. The pre-computations were done not only to be able to return the persistence diagrams in every iteration, avoiding computations, but also to avoid the complexity of the calculation in the Tensorflow's graph: they were feeded as constants rather than tensors. The precission of the model was tested against the 216 images of the testing dataset, using the accuracies mentioned in Section 4.4.

Due to the artifacts in the labels provided in the dataset, belonging to the halo of the heart chambers (Figure 4.2), the resulted segmentation in the dataset presented some difficulties.

We could see some improvements applying the loss to every channel independently. In this case using the total persistence distance, barcodes augmented the accuracy several points with respect to the non-regularized loss, as it is show in Table 4.1. In the table 4.2 the accuracies for the model with the Bottleneck Loss regularizer are shown.

Nevertheless, using persistent homology in the labels for the four channels did not retrieve a sufficient increase in the results, considering the computation time incremented (doubled) even after the pre-computations were done, as we can see in Table 4.3.

Something to take into account is that the third channel did not have artifacts at all. The fourth channel had a very little amount of artifacts: a number of only

Table 4.1: Comparison of the results of the combination of the geometric losses vs the results calculating the total persistence of all the channels in order, represented with (i) for the *i*th-channel, for every accuracy.

Metric	GL	GL+PL(0)	GL+PL(1)	GL+PL(2)	GL+PL(3)
MeanIoU accuracy (%)	61.0391	62.4377	60.5381	60.0784	65.7350
CE accuracy (%)	95.2583	95.7985	95.3162	95.140 48	96.3788
DC accuracy (%)	96.2591	96.6109	96.2927	96.0831	97.2049
Time spent (hours)	2.255	2.671	2.482	2.930	3.630

Table 4.2: Comparison of the results of the combination of the geometric losses vs the results from the calculation of the bottleneck distance using the actual persistence of all the channels, represented with (i) the *i*th-channel, for every accuracy.

Metric	GL	GL+BL(0)	GL+BL(1)	GL+BL(2)	GL+BL(3)
MeanIoU accuracy (%)	61.0391	64.2347	66.0457	65.7243	63.1268
CE accuracy (%)	95.2583	96.1227	96.4405	96.3299	96.0592
DC accuracy (%)	96.2591	96.9546	97.1654	97.0742	96.8970
Time spent (hours)	2.255	3.248	2.678	3.968	3.654

Table 4.3: Comparison of the results of the combination of the geometric losses vs the results from the calculation of the bottleneck loss using the persistence of all the labels, with (i) representing the dimension, for every accuracy.

Metric	GL	GL+BL(0,1)	GL+BL(1)	GL+BL(0)
MeanIoU accuracy (%)	61.0391	62.0236	64.9727	65.8017
CE accuracy (%)	95.2583	95.5660	96.2320	96.3586
DC accuracy (%)	96.2591	96.5259	97.0676	97.2157
Time spent (hours)	2.255	9.064	4.001	3.453

four images out of the 400 training images in the dataset had irregularities for the dimension 0, matching some of the artifacts in dimension 1 for the other channel, creating holes without color that were considered as independent background components.

However, ignoring the irregular representations in the labels and forcing a preselected set of input diagrams in the regularizers did also improved the segmentation. This method is similar to already existing methods in the state of the art in which a topologically ideal set of labels was considered [29, 30].

Specifically, the inputs were:

- Channel 0 dimension 0: []
- Channel 0 dimension 1: []
- Channel 1 dimension 0: []
- Channel 1 dimension 1: []
- Channel 2 dimension 0: []
- Channel 2 dimension 1: []
- Channel 3 dimension 0: []
- Channel 3 dimension 1: [0., 1.]

Meaning that all the channels have a single connected component and no holes, except for the last channel, the background, with one hole referring to the heart chambers. In the Table 4.4 we can find the results related to the computations.

Table 4.4: Summary of the results forcing a preselected set of ideal persistence diagrams, using different methods for every accuracy.

Metric	GL	GL+PL	GL+BL
MeanIoU accuracy (%)	61.0391	65.9230	63.8866
CE accuracy (%)	95.2583	96.4536	96.0734
DC accuracy (%)	96.2591	97.2698	96.8952
Time spent (hours)	2.255	3.691	8.296

While using the topological losses a careful analysis of the data acquires more relevance, since it could be prone to deteriorate the segmentation with a fixed ideal set of labels.

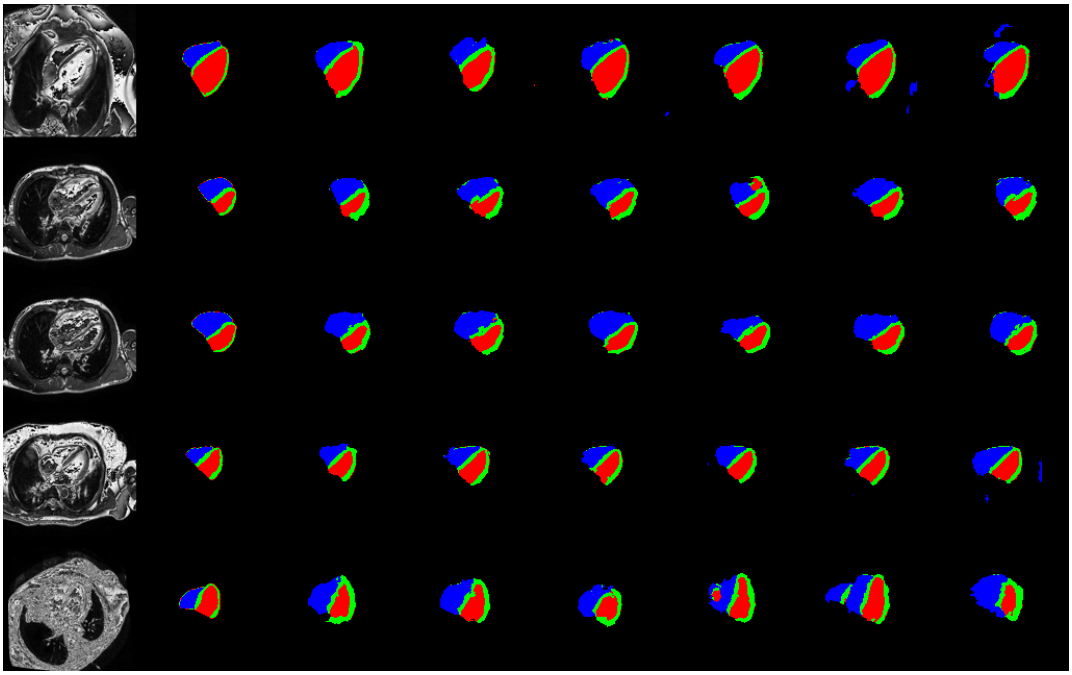


Figure 4.3: Example of segmentations using different methods. The first image is the input image and the second is the label. The next images correspond to the segmentations using the methods: bottleneck loss for dimension 0 and 1 applied to channels 0, 1 and 2; the bottleneck loss applied to a set of ideal labels for dimension 0 and 1; the total persistence applied to a set of ideal labels for dimension 0 and 1; and the loss without regularizers.

These results might indicate that the satisfactory segmentation performed in [29, 30] is due to the lack of artifacts in the labels provided in the ACDC dataset [21].

The computations limited to the fourth channel did not retrieve any results. The segmentation worsen instead.

We find in figure 4.3 an example of segmentation using all the methods. Images in order: input image, label, segmentation using bottleneck for channels 2 and 3, segmentation using bottleneck with a set of ideal labels, segmentation using sum of persistence for channels 2 and 3, segmentation using persistence with an ideal set of labels, segmentation without regularizers.

A summary of the accuracies can be found in Figure 4. It is a comparison between accuracy plots. The first column corresponds to the MeanIoU Accuracy, the second to the categorical Accuracy and the third to the Dice Coefficient Accuracy. Every row corresponds to a training with a different loss.

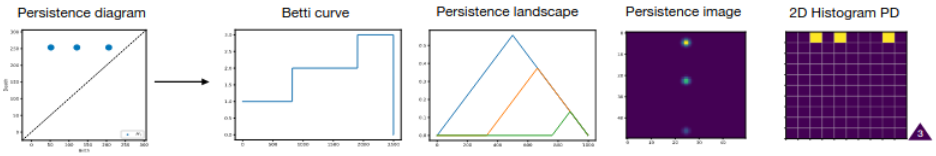


Figure 4.4: Four vectorized summaries extracted from a persistence diagram. Image source: [78]

4.8 Next steps

We also considered the use of finite-dimensional vector representations of persistent diagrams. The persistence diagrams vectorizations have been proposed as topological priors and have shown predictive power [63, 62].

This priors summarize the information provided by the persistent homology features: *Betti Curves* [65], *Persistence Landscape* [63], *Persistent Image* [62] and *2D Histogram* [66], among others.

Specifically, previous results on persistence images have shown relevant performance gains comparing this technique with other methods like the bottleneck or wasserstein distances [62]. Nonetheless, the computation time for persistent images in this project exceeded to a large degree the computation of the other methods, even with a reduced dataset.

However, a new line of research have recently appeared that promise an injective invariant reduction in the complexity of the data applying some filters. This method is called *convolutional persistence* [67] and could be helpful to alleviate the problem of efficiency of some persistent homology methods in dataset segmentation.

Another line of research is the *Ensemble deep learning models* for segmentation. They are the result from a combination of multiple models, aggregating the output of each of them.

Ensemble models have been applied to a variety of medical imaging tasks, showing an increment in accuracy [79, 80, 81]. Two pretrained models, one trained to segment long-axis CMR images and the other trained to segment short-axis CMR images, could be ensembled to obtain a model that can learn the segmentation of a CMR image in both axis for the same patient and use it to learn twice as the number of features and hopefully learn the correlation between them.

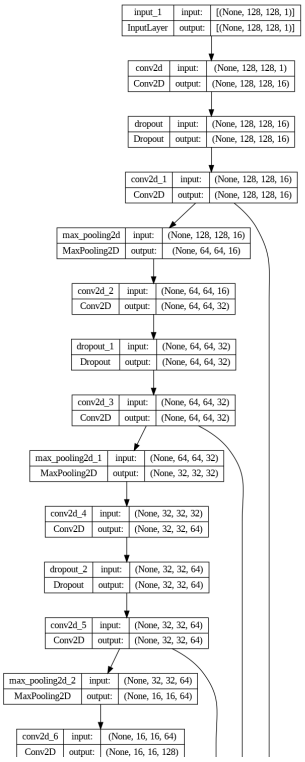


Figure 4.5: Row 1

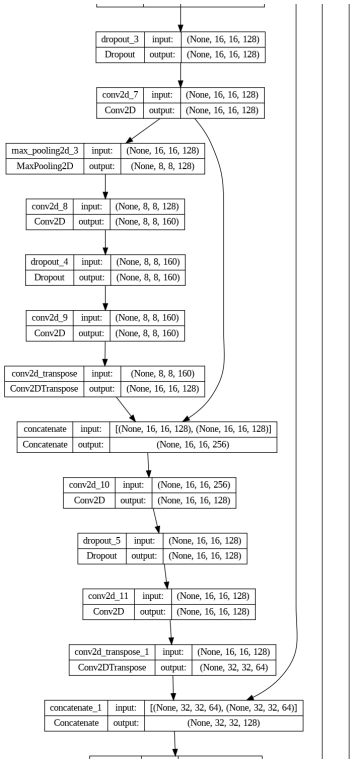


Figure 4.6: Row 2

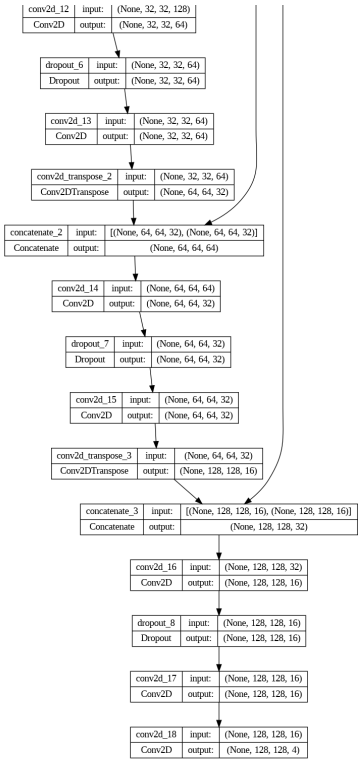


Figure 4.7: Row 3

Figure 4.8: Model summary

Chapter 5

Conclusions

In this work we implemented a cutting-edge method to regularize a CMR segmentation using persistent homology. We built a convolutional model to perform segmentation in CMR images and applied topological regularizers to enhance the segmentation. The underlying hypothesis of this work was that by incorporating persistent homology, a segmentation model can avoid large topological errors caused by traditional geometrical losses. As a result, we developed and evaluated a topological regularizer.

A topological analysis of the data was necessary to improve the segmentation, which was tricky due to the perturbations in the labels. The regularization method improved the segmentation compared with geometrical losses. Not only the accuracy was higher, but also the segmentations were visually more consistent.

However, we found that applying the regularizer only to one channel rather than all the channels at the same time not only resulted in a reduction of computation time, but also in higher accuracy. This could be attributed to the fact that the parameters learned by the model to optimize certain channels may negatively impact the segmentation of the remaining channels.

Moreover, by applying a set of ideal labels, that is, calculating the persistence diagrams ignoring the artifacts in the labels, we obtained worse results. To perform the segmentation focusing in the shape of the chambers and ignoring the artifacts retrieved a better accuracy than performing the segmentation without a topological regularizer. Nevertheless, the accuracy obtained from the use of the actual persistence diagrams was higher.

Our findings also indicated that setting an ideal set of labels may not be the best method for segmentation when working with data that has perturbations. This method retrieved segmentations that ignored the halo surrounding the chambers in the label images. This could be a problem not only because of the loss of accuracy but also because the halo represents the walls of the chambers, which

helps geometrical losses such as dice loss or pixel-wise loss in the segmentation.

Furthermore, since every channel has topological properties on its own, finding a loss that that incorporate them to the model could improve the segmentation. There is a need to for a combined approach to analyze the channels altogether.

While our approach may have some processing time limitations, since every set of persistence diagrams added to the bottleneck loss increments the accuracy substantially, it could still be used for real-time segmentation with appropriate adjustments. Certain optimization techniques, such as using heuristics to speed up the classification step or carefully selecting and evaluating features, could improve the overall speed of the process. Furthermore, the techniques mentioned in Section 4.8 could use persistent homology to improve the segmentation and at the same time being computationally efficient. Finally, testing the model using the entire dataset, with a deeper model and a higher early stopping value, would allow us to fully evaluate the capabilities of our methods.

Bibliography

[1] Wang, Haohan, and Bhiksha Raj. "On the origin of deep learning." arXiv preprint arXiv:1702.07800 (2017).

[2] Balaban, Stephen. "Deep learning and face recognition: the state of the art." Biometric and surveillance technology for human and activity identification XII 9457 (2015): 68-75.

[3] Matsunaga, Daiki, Toyotaro Suzumura, and Toshihiro Takahashi. "Exploring graph neural networks for stock market predictions with rolling window analysis." arXiv preprint arXiv:1909.10660 (2019).

[4] Mc Namara, Kevin, Hamzah Alzubaidi, and John Keith Jackson. "Cardiovascular disease as a leading cause of death: how are pharmacists getting involved?." Integrated pharmacy research & practice 8 (2019): 1.

[5] Zhuang, Xiahai, et al. "Evaluation of algorithms for multi-modality whole heart segmentation: an open-access grand challenge." Medical image analysis 58 (2019): 101537.

[6] McGill Jr, Henry C., C. Alex McMahan, and Samuel S. Gidding. "Preventing heart disease in the 21st century: implications of the Pathobiological Determinants of Atherosclerosis in Youth (PDAY) study." Circulation 117.9 (2008): 1216-1227.

[7] O'Donnell, Martin J., et al. "Global and regional effects of potentially modifiable risk factors associated with acute stroke in 32 countries (INTERSTROKE): a case-control study." The lancet 388.10046 (2016): 761-775.

[8] Marian, Ali J., and Eugene Braunwald. "Hypertrophic cardiomyopathy: genetics, pathogenesis, clinical manifestations, diagnosis, and therapy." Circulation research 121.7 (2017): 749-770.

[9] Ripley, David & Musa, TA & Dobson, Laura & Plein, S & Greenwood, J. (2016). Cardiovascular magnetic resonance imaging: What the general cardiologist should know. Heart. 102. heartjnl-2015. 10.1136/heartjnl-2015-307896.

[10] Ghosh, Tarun Kanti, et al. "Multi-class probabilistic atlas-based whole heart segmentation method in cardiac CT and MRI." IEEE Access 9 (2021): 66948-66964.

[11] Medrano-Gracia, Pau, et al. "Challenges of cardiac image analysis in large-scale population-based studies." Current cardiology reports 17.3 (2015): 1-7.

[12] Shuai, Bing, Ting Liu, and Gang Wang. "Improving fully convolution network for semantic segmentation." arXiv preprint arXiv:1611.08986 (2016).

[13] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[14] Loukas, Andreas. "What graph neural networks cannot learn: depth vs width." arXiv preprint arXiv:1907.03199 (2019).

[15] Berner, Julius, et al. "The modern mathematics of deep learning." arXiv preprint arXiv:2105.04026 (2021).

[16] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[17] Chen, Hao, et al. "Iterative multi-domain regularized deep learning for anatomical structure detection and segmentation from ultrasound images." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2016.

[18] Kamnitsas, Konstantinos, et al. "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation." Medical image analysis 36 (2017): 61-78.

[19] Ravishankar, Hariharan, et al. "Joint deep learning of foreground, background and shape for robust contextual segmentation." International Conference on Information Processing in Medical Imaging. Springer, Cham, 2017.

[20] Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." 2016 fourth international conference on 3D vision (3DV). IEEE, 2016.

[21] Bernard, Olivier, et al. "Deep learning techniques for automatic MRI cardiac multi-structures segmentation and diagnosis: is the problem solved?." IEEE transactions on medical imaging 37.11 (2018): 2514-2525.

[22] Campello, Victor M., et al. "Multi-centre, multi-vendor and multi-disease cardiac segmentation: the M&Ms challenge." IEEE Transactions on Medical Imaging 40.12 (2021): 3543-3554.

[23] Zhuang, Xiahai, et al. "Evaluation of algorithms for multi-modality whole heart segmentation: an open-access grand challenge." Medical image analysis 58 (2019): 101537.

[24] Çiçek, *Özgü*n, et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation." International conference on medical image computing and computer-assisted intervention. Springer, Cham, 2016.

[25] Yu, Lequan, et al. "Automatic 3D cardiovascular MR segmentation with densely-connected volumetric convnets." International conference on medical image computing and computer-assisted intervention. Springer, Cham, 2017.

[26] Oktay, Ozan, et al. "Anatomically constrained neural networks (ACNNs): application to cardiac image enhancement and segmentation." IEEE transactions on medical imaging 37.2 (2017): 384-395.

[27] Nosrati, Masoud S., and Ghassan Hamarneh. "Incorporating prior knowledge in medical image segmentation: a survey." arXiv preprint arXiv:1607.01092 (2016).

[28] Duan, Jinming, et al. "Automatic 3D bi-ventricular segmentation of cardiac images by a shape-refined multi-task deep learning approach." IEEE transactions on medical imaging 38.9 (2019): 2151-2164.

[29] Clough, James R., et al. "A topological loss function for deep-learning based image segmentation using persistent homology." IEEE Transactions on Pattern Analysis and Machine Intelligence 44.12 (2020): 8766-8778.

[30] Byrne, Nick, et al. "A persistent homology-based topological loss for CNN-based multi-class segmentation of CMR." IEEE Transactions on Medical Imaging (2022).

[31] Mosinska, Agata, et al. "Beyond the pixel-wise loss for topology-aware delineation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[32] Hu, Xiaoling, et al. "Topology-preserving deep image segmentation." Advances in neural information processing systems 32 (2019).

[33] Eldan, R., and O. Shamir. "The power of depth for feedforward neural networks: arXiv preprint." (2016).

[34] Nosrati, Masoud S., and Ghassan Hamarneh. "Incorporating prior knowledge in medical image segmentation: a survey." arXiv preprint arXiv:1607.01092 (2016).

[35] Han, Xiao, Chenyang Xu, and Jerry L. Prince. "A topology preserving level set method for geometric deformable models." IEEE Transactions on Pattern Analysis and Machine Intelligence 25.6 (2003): 755-768.

[36] Zeng, Yun, et al. "Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N–D images." Computer vision and image understanding 112.1 (2008): 81-90.

[37] Vicente, Sara, Vladimir Kolmogorov, and Carsten Rother. "Graph cut based image segmentation with connectivity priors." 2008 IEEE conference on computer vision and pattern recognition. IEEE, 2008.

[38] Caldairou, Benoît, et al. "Segmentation of the cortex in fetal MRI using a topological model." 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro. IEEE, 2011.

[39] Calin, Ovidiu. Deep learning architectures. Springer International Publishing, 2020.

[40] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[41] Stewart, James. Calculus. Cengage Learning, 2015.

[42] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks." Neural networks 3.5 (1990): 551-560.

[43] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989): 359-366.

[44] Cybenko, George. "Approximation by superpositions of a sigmoidal function." Mathematics of control, signals and systems 2.4 (1989): 303-314.

[45] Barron, Andrew R. "Universal approximation bounds for superpositions of a sigmoidal function." IEEE Transactions on Information theory 39.3 (1993): 930-945.

[46] Zhou, Yi-Tong, and Rama Chellappa. "Stereo matching using a neural network." ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing. IEEE Computer Society, 1988.

[47] Edelsbrunner, Herbert, and John L. Harer. Computational topology: an introduction. American Mathematical Society, 2022.

[48] Dey, Tamal Krishna, and Yusu Wang. Computational topology for data analysis. Cambridge University Press, 2022.

[49] Wagner, Hubert, Chao Chen, and Erald Vuçini. "Efficient computation of persistent homology for cubical data." Topological methods in data analysis and visualization II. Springer, Berlin, Heidelberg, 2012. 91-106.

[50] Gray, Stephen B. "Local properties of binary images in two dimensions." IEEE Transactions on computers 100.5 (1971): 551-561.

[51] Li, Xiaoxing, Paulo RS Mendonça, and Rahul Bhotika. "Texture analysis using Minkowski functionals." Medical Imaging 2012: Image Processing. Vol. 8314. SPIE, 2012.

[52] Schladitz, Katja, Joachim Ohser, and Werner Nagel. "Measurement of intrinsic volumes of sets observed on lattices." 13th International conference on discrete geometry for computer imagery. Vol. 4245. Berlin, Heidelberg, New York: Springer, 2006.

[53] Svane, Anne Marie. "Estimation of intrinsic volumes from digital grey-scale images." Journal of mathematical imaging and vision 49.2 (2014): 352-376.

[54] Chazal, Frédéric, et al. The structure and stability of persistence modules. Vol. 10. Berlin: Springer, 2016.

[55] Bleile, Bea, et al. "The persistent homology of dual digital image constructions." Research in Computational Topology 2. Springer, Cham, 2022. 1-26.

[56] Leygonie, Jacob, Steve Oudot, and Ulrike Tillmann. "A framework for differential calculus on persistence barcodes." Foundations of Computational Mathematics 22.4 (2022): 1069-1131.

[57] Guo, Chuan, et al. "On calibration of modern neural networks." International conference on machine learning. PMLR, 2017.

[58] Krishnan, Ranganath, and Omesh Tickoo. "Improving model calibration with accuracy versus uncertainty optimization." Advances in Neural Information Processing Systems 33 (2020): 18237-18248.

[59] Chen, Chao, et al. "A topological regularizer for classifiers via persistent homology." The 22nd International Conference on Artificial Intelligence and Statistics. PMLR, 2019.

[60] Chazal, Frédéric, and Vincent Divol. "The density of expected persistence diagrams and its kernel based estimation." arXiv preprint arXiv:1802.10457 (2018).

[61] Br*üe*l-Gabrielsson, Rickard, et al. "Topology-Aware Surface Reconstruction for Point Clouds." Computer Graphics Forum. Vol. 39. No. 5. 2020.

[62] Adams, Henry, et al. "Persistence images: A stable vector representation of persistent homology." Journal of Machine Learning Research 18 (2017).

[63] Bubenik, Peter. "Statistical topological data analysis using persistence landscapes." J. Mach. Learn. Res. 16.1 (2015): 77-102.

[64] Berry, Eric, et al. "Functional summaries of persistence diagrams." Journal of Applied and Computational Topology 4.2 (2020): 211-262.

[65] Gameiro, Marcio, Konstantin Mischaikow, and William Kalies. "Topological characterization of spatial-temporal chaos." Physical Review E 70.3 (2004): 035203.

[66] Lacombe, Théo, Marco Cuturi, and Steve Oudot. "Large scale computation of means and clusters for persistence diagrams using optimal transport." Advances in Neural Information Processing Systems 31 (2018).

[67] Solomon, Elchanan, and Paul Bendich. "A Convolutional Persistence Transform." arXiv preprint arXiv:2208.02107 (2022).

[68] Abadi, Martín, et al. "TensorFlow: a system for Large-Scale machine learning." 12th USENIX symposium on operating systems design and implementation (OSDI 16). 2016.

[69] Maria, Clément, et al. "The gudhi library: Simplicial complexes and persistent homology." International congress on mathematical software. Springer, Berlin, Heidelberg, 2014.

[70] P.W.D. Charles, Project Title, (2013), GitHub repository, https://github.com/charlespwd/project-title

[71] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[72] Thoma, Martin. "A survey of semantic segmentation." arXiv preprint arXiv:1602.06541 (2016).

[73] Nordström, Marcus, et al. "On Image Segmentation With Noisy Labels: Characterization and Volume Properties of the Optimal Solutions to Accuracy and Dice." arXiv preprint arXiv:2206.06484 (2022).

[74] Wang, Zhenzhou. "Deep learning for image segmentation: veritable or overhyped?." arXiv preprint arXiv:1904.08483 (2019).

[75] Ahmad, Tariq, et al. "Machine learning methods improve prognostication, identify clinically distinct phenotypes, and detect heterogeneity in response to therapy in a large cohort of heart failure patients." Journal of the American Heart Association 7.8 (2018): e008081.

[76] Gholamy, Afshin, Vladik Kreinovich, and Olga Kosheleva. "Why 70/30 or 80/20 relation between training and testing sets: a pedagogical explanation." (2018).

[77] Joseph, V. Roshan. "Optimal ratio for data splitting." Statistical Analysis and Data Mining: The ASA Data Science Journal (2022).

[78] Fasy, Brittany Terese, et al. "Comparing distance metrics on vectorized persistence summaries." Topological Data Analysis and Beyond Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020). 2020.

[79] Zheng, Hao, et al. "A new ensemble learning framework for 3D biomedical image segmentation." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. No. 01. 2019.

[80] Kang, Jaeyong, and Jeonghwan Gwak. "Ensemble of instance segmentation models for polyp segmentation in colonoscopy images." IEEE Access 7 (2019): 26440-26447.

[81] Hann, Evan, et al. "Deep neural network ensemble for on-the-fly quality control-driven segmentation of cardiac MRI T1 mapping." Medical image analysis 71 (2021): 102029.

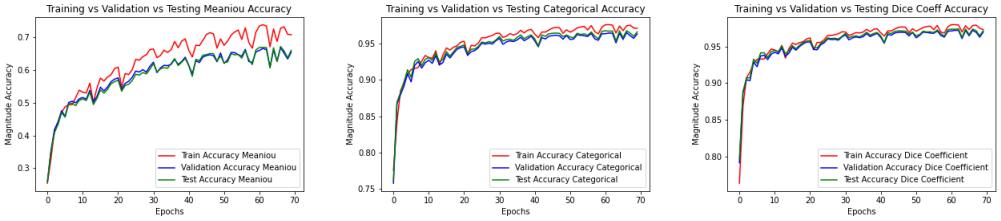
[82] Gavrikov, P. "Visualkeras." (2020).

Appendices

I Figures from results

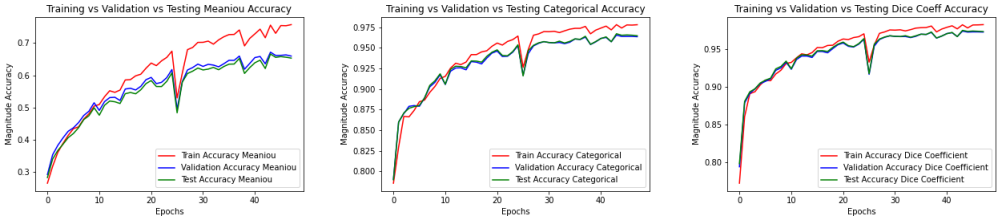
In the next pages we will find the graphics corresponding to the resulting acuracies of the different methods discussed in Section 4.7.

(a)



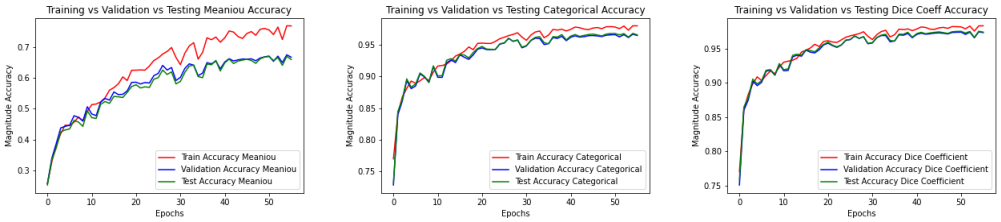
Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the total persistence loss regularizer considering the channel 0.

(b)



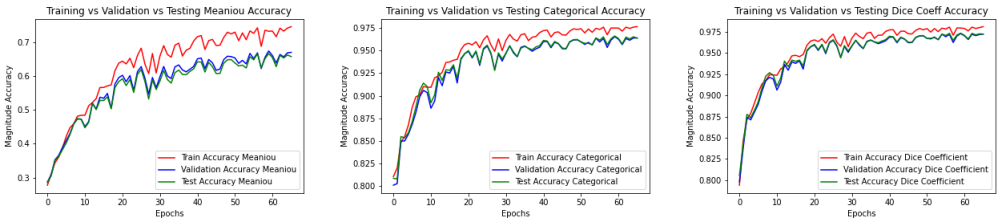
Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the total persistence loss regularizer considering the channel 1.

(c)



Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the total persistence loss regularizer considering the channel 2.

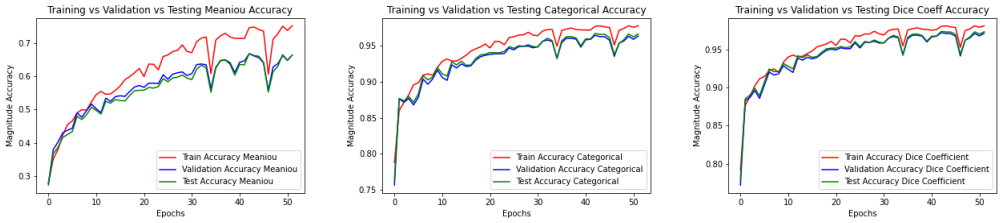
(d)



Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the total persistence loss regularizer considering the channel 3.

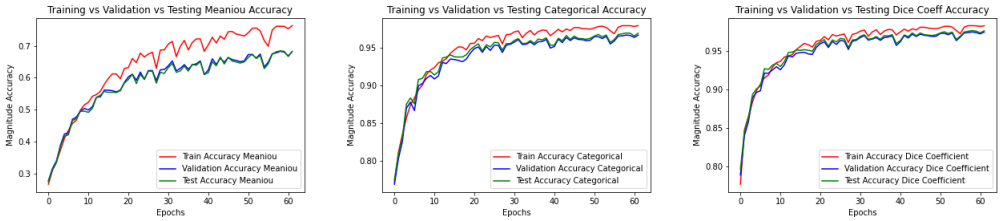
Figure 1: Comparison between accuracy plots for every channel using the total persistence regularizer.

(a)



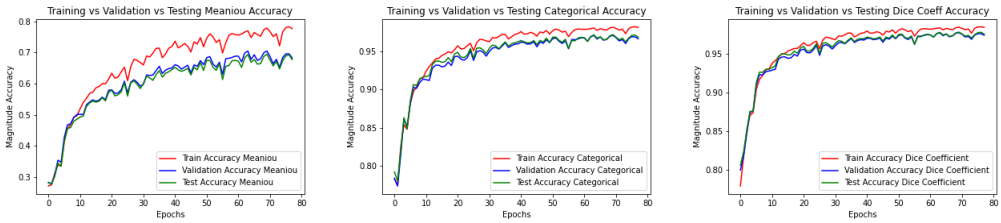
Accuracies of the long-axis segmentation model using a combination of dice loss, pixel loss and the bottleneck loss regularizer for all channels and for dimension 0 and 1.

(b)



Accuracies of the long-axis segmentation model using a combination of dice loss, pixel loss and the bottleneck loss regularizer for all channels and for dimension 1.

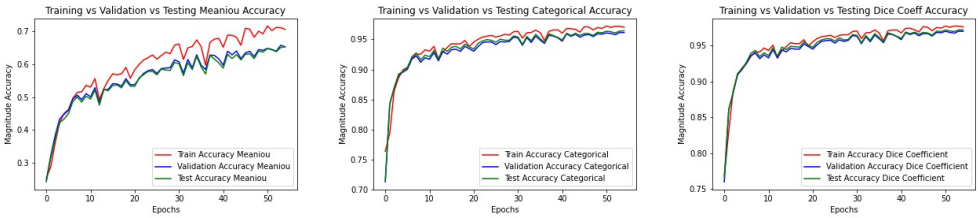
(c)



Accuracies of the long-axis segmentation model using a combination of dice loss, pixel loss and the bottleneck loss regularizer for all channels and for dimension 0.

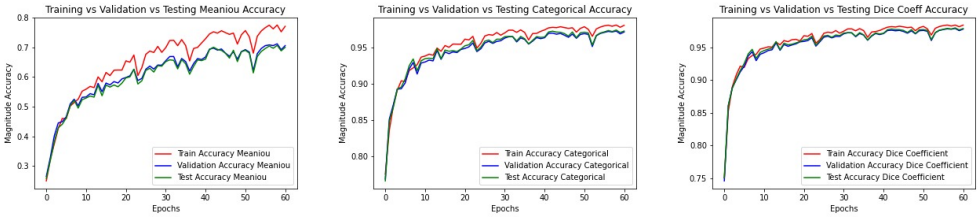
Figure 2: Comparison between accuracy plots for every channel and varying the dimension.

(a)



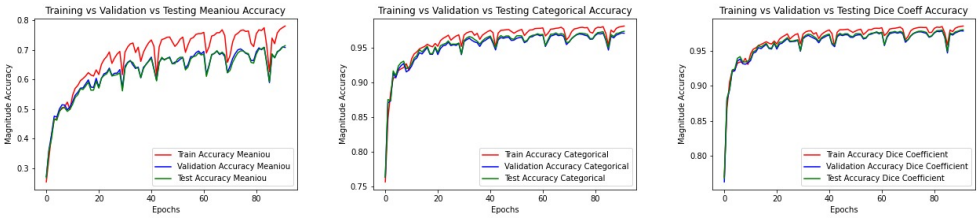
Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the bottleneck loss regularizer considering the channel 0.

(b)



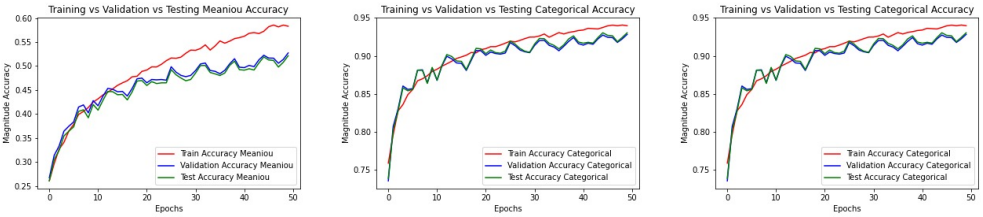
Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the bottleneck loss regularizer considering the channel 1.

(c)



Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the bottleneck loss regularizer considering the channel 2.

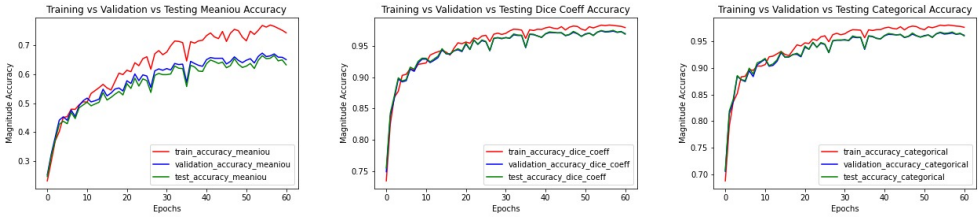
(d)



Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss and the bottleneck loss regularizer considering the channel 3.

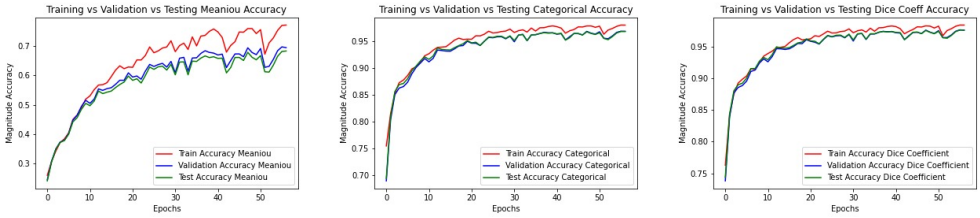
Figure 3: Comparison between accuracy plots for every channel using the bottleneck regularizer.

(a)



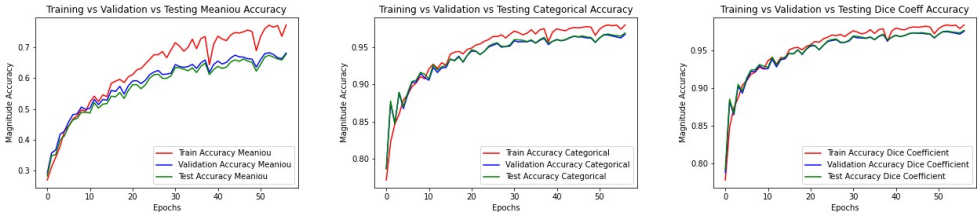
Accuracies of the long-axis segmentation model using a combination of dice loss and pixel loss.

(b)



Accuracies of the long-axis segmentation model using a combination of dice loss, pixel loss and the total persistence loss regularicer with the inputs the persistence diagrams considering only three connected components in the labels.

(c)



Accuracies of the long-axis segmentation model using a combination of dice loss, pixel loss and the bottleneck loss regularizer with the inputs the persistence diagrams considering only three connected components in the labels.

Figure 4: Comparison between an accuracy plot of a segmentation performed without regularizers and accuracy plots forcing from a segmentation performed with a preselected set of ideal persistence diagrams.