UNIVERSITAT DE BARCELONA

Treball final de grau

DOBLE GRAU DE MATEMÀTIQUES I INFORMÁTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Medical Image Segmentation with Limited Data

Autor: Iván Canales

Director:	Dr. Simone Balocco
Realitzat a:	Departament de Matemática Aplicada
Barcelona,	13 de juny de 2022

Abstract

Ischemic Heart Disease (IHD) is one of the leading causes of mortality in Spain; early diagnosis is key. Intravenous ultrasound imaging (IVUS) can help identify symptoms of IHD, at the cost of segmenting a large volume of frames by medical professionals. While promising, automated image segmentation using Convolutional Neural Networks (CNN) suffer from sample scarcity: a large amount of parameters is often used, and medical imaging datasets are typically small and costly to acquire and label. In this report we study and compare state of the art methods used to deal with sample scarcity. In particular we introduce data augmentation methodologies, specialized training losses and transfer learning methods, and compare their performance on IVUS segmentation of the media and lumen or the artery. Additionally we introduce a promising paradigm, few-shot segmentation, and provide an initial implementation using PFENet. This implementation can avoid significant overfitting, even when trained with a single example, outperforming traditional CNNs on the same segmentation problem.

Resum

Les malalties de les arteries coronaries son una de les primeres causes de mortalitat en Espanya; el diagnòstic precoç és molt important. Les imatges intravenoses per ultrasons (IVUS) poden ajudar a diagnosticar símptomes de malalties coronaries, però requereixen el processament de una gran quantitat de fotogrames per part de professionals mèdics. Encara que els resultats son favorables, l'automatització de la segmentació de les imatges mitjançant l'ús de Xarxes Neuronals Convolucionals (CNN) sofreixen de la manca de dades: habitualment contenen una gran quantitat de paràmetres, i les dades disponibles són habitualment poques i costosos d'etiquetar. En aquest informe estudiem i comparem tècniques d'última generació per combatre la manca de dades. En particular, introduïm mètodes d'augment de dades, funcions de pèrdua especialitzades i tècniques de transferència de coneixement; i comparem l'efectivitat d'aquests per a la segmentació de la media i el lumen de les artèries. Addicionalment, introduïm el prometedor paradigma de segmentació 'few-shot' (amb poques dades) i n'oferim una implementació inicial. Aquesta pot evitar el sobre-ajustament en l'entrenament, fins i tot quan s'entrena amb un únic exemple, millorant els resultats obtinguts mitjançant CNNs tradicionals.

2020 Mathematics Subject Classification. 68T07

Dedicado a Luisa y Josefa.

Contents

Chapter 1

Introduction

1.1 Motivation

Ischemic heart disease (IHD)—sometimes called coronary artery disease—is one of the leading causes of mortality. In 2020, more than a 30% of the deaths in Spain can be attributed to some form of IHD[7]. Therefore, early detection and diagnostic of IHD is of interest for the healthcare system.

Intravascular ultrasound (IVUS) is a medical imaging methodology designed to study the inner wall (endothelium) of blood vessels. It has been established as the gold standard for *in vivo* imaging of the vessel wall of coronary arteries [10] replacing angiographic imaging in cases where it is considered unreliable. It is therefore an important diagnostic tool for the diagnostic of coronary artery disease.

However, IVUS comes at a high human cost. The vessel morphology in the frames must be identified by trained physicians and a large amount of frames are generated per session. There is an interest to automate the task of interpreting the IVUS frames, in particular the problem of segmentation: given a greyscale frame obtained through IVUS, identify the interior of the vessel, through which blood flows (lumen), and the wall of the artery (media).

In the recent years, the rise of deep (machine) learning has brought upon improvements on automatic image segmentation, which attempts to alleviate the human cost of semantically segmentation, in particular for medical imaging. Unfortunately these models do require large amounts of labelled data to provide favourable results.

1.2 Our Contribution

In this project we implement and evaluate approaches to dealing with lack of labelled data in image segmentation. Using the two datasets provided by [2], we analyse three distinct approaches. First, data augmentation can be used to generate synthetic samples without the need for medical intervention. Second, we explore manipulating the training loss to improve model performance. Finally we employ fine tuning methods, and we study the limit case of just one training sample (one-shot learning).

We study the influence of diverse data augmentation methods: affine transforms such as rotations, flips, and zooms; and elastic deformations. For the elastic deformations we propose an implementation that aims to mimic the involuntary movements of the hand when drawing the masks. We couple these with tweaks on the training process used in [25], such as min–max normalization of the input images. After evaluation, we observe an increase in performance of the models that use data augmentation, with varying degrees of influence.

Our study of training losses compares the efficacy of categorical crossentropy, and the Jaccard and Hausdorff distances. A loss based on an approximation of the Hausdorff distance is provided as an effort to minimize the computational cost of the naive implementation. We argue that this loss could lead to improvements in the performance of the model, and provide theoretical results and qualitative analysis to support our claim.

For the segmentation task, we use fully convolutional Artificial Neural Networks, largely based on the U-Net [24] architecture. We then analyse the effects of using a pre-trained Inception ResNet [14] as the backbone for feature extraction to improve segmentation performance.

Finally we explore the 'few-shot' learning architecture, or learning the problem using a very reduced training set—usually five or less examples. For this purpose we specialize a pre-trained PFENet [29] through fine-tuning. This leads to promising results, outperforming significantly the other models—when trained with a single example. We obtain a competitive performance in the segmentation of the media, comparable to the results obtained using more data.

1.3 Literature Review

For the IVUS segmentation task, Convolutional Neural Networks are the most used machine learning algorithms [20] [1] [31] [32] [33] [22] . U-Net based architectures

are common among the deep learning architectures [1] [31] [32]. Other methodologies used in the literature—but not in the scope of this report—include traditional supervised machine learning methods like Support Vector Machines [30] and Random Forests [30], and other image processing algorithms like the 3D-helical snake segmentation described in [13].

When it comes to data augmentation, affine transformations are commonly used [1] [33]. Some papers propose the use of blurring (Gaussian, Average, Median) [33], however we know that the artefacts in IVUS imaging follow speckle noise. [32] propose other ad hoc methods adapted to IVUS imaging designed to reproduce 'difficult frames'.

1.4 Structure of the Project

The execution of this project has been structured into four distinct phases: familiarization with tooling, review of previous results, improvements to previous models, and research and application of few shot algorithms.

The first phase begins in December of 2021 with the first meeting. The subsequent monts were allocated to learn and familiarize ourselves with the machine learning frameworks used, as well as the dataset and the problem to solve.

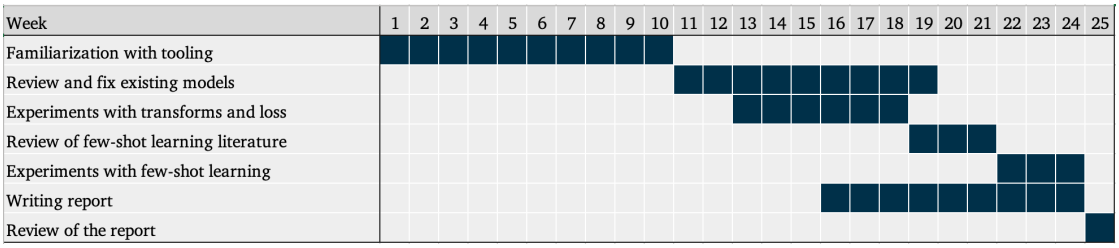

Figure 1.1: Gantt diagram showing the timeline (in weeks).

In the second phase, the focus of the developing was to construct a framework for prototyping the models, distributable as a PYTHON package using TENSORFLOW and KERAS as the deep learning framework . As seen in Fig. 1.1, this lasts until week 19. The reason for that, was that the task to adapt the previous material was underestimated. Several implementations were missing, and on the existing implementations non-trivial changes had to be made. Notable examples are the Jaccard loss and elastic transformation implementations, which were rewritten in their entirety.

It is also worth noting, that adapting what was present required in depth understanding of the implementation, as well as familiarity with the dataset structure. In short, this phase was originally conceived to comprise minimal changes to use the existing models, however it ended up requiring in depth knowledge of, not only the concepts, but also their implementation.

The third and fourth phases include most of the original work for this project. In the third phase we study methods that can improve the existing models. This this include the study of the Hausdorff distance from a theoretical and qualitative perspective, along with the implementation of the Hausdorff loss.

For the fourth and final phase, we take on the task of researching and using few-shot learning models. This was challenging from both a conceptual an technical point of view. Not only did it require to learn how the models worked, but also required familiarity with an alternative deep learning framework, PyTorch , to be able to reuse pre-trained weights. Adapting the model to our segmentation task required knowledge about the inner workings of the training and evaluation methods, which lead to a considerable technical effort.

Chapter 2

Technical Background

In this chapter we introduce the foundations of Neural Networks. We describe this family of algorithms and introduce concepts that will be used in subsequent chapters. Since we focus our efforts on Convolutional Neural Networks, we also provide some theoretical solutions that might shed some light on their performance for this task.

2.1 Neural Networks

The Deep Learning algorithms, namely Neural Networks and in particular Convolutional Neural Networks, have become the golden standard of machine learning algorithms in the last decades for tasks such as image classification and segmentation, requiring minimal preprocessing steps.

In this chapter we introduce the theoretical foundations of this class of machine learning algorithms, and we will present some results that attempt to explain the empirical results observed after using Convolutional Neural Networks.

Definition 2.1. Let X, W and Y be two real vector spaces, we define a **layer** as an application Λ where:

1. $\Lambda : X \times W \to Y$ is the composition of a differentiable linear (w.r.t. X) function $f : X \times W \to Y$ and a differentiable non-linearity $\psi : Y \to Y$.
2. $\dim X$ is the **input dimension** of Λ, and $\dim Y$ the **output dimension**.
3. for some $\theta \in W$—the **weights**—we indicate $\Lambda_\theta := \Lambda(\star, \theta)$ as the layer with fixed weights.

Note 2.2. When a layer is the matrix product of the input vector and its weights (followed by the non-linearity), it is often called a **dense** layer.

Definition 2.3. Let $X = \mathbb{R}^n$ and $Y = \mathbb{R}^m$ be two sets, then for some $k > 0$, a **Feedforward Artificial Neural Network** (ANN) Φ is an application

$$\Phi : X \times \mathbb{R}^k \to Y,$$

such that $\Phi(x, \theta) = (\Lambda_1(\star, \theta_1) \circ \Lambda_2(\star, \theta_2) \circ \cdots \circ \Lambda_r(\star, \theta_2))(x)$, and the Λ_i $(1 \leq i \leq k)$ are its layers, defined as

$$\Lambda_i : \mathbb{R}^{n_i} \times \mathbb{R}^{k_i} \to \mathbb{R}^{m_i}, \tag{2.1}$$

where $n = n_1$, $n_{i+1} = m_i$, $m = m_r$ and $\theta \mapsto (\theta_1, \theta_2, \ldots, \theta_r)$ describes an isomorphism $\mathbb{R}^k \cong \mathbb{R}^{k_1} \times \mathbb{R}^{k_2} \times \cdots \times \mathbb{R}^{k_r}$. From now on, we assume w.l.o.g. θ to be the concatenation of every θ_i.

Remark 2.4. If the layers 2.1 were linear, for a fixed set of weights θ, (2.1) would be isomorphic to the product of r matrices, and therefore defined by a matrix $M(\theta) \in \mathcal{M}^{n \times m}$.

Note 2.5. The name 'Artificial Neural Network' is a reference to the fact that they take an inspiration from the neuron networks in the human brain. Usually ANNs are portrayed as directed graphs 2.1, the nodes of which are often referred as 'neurons'. The qualifier 'feedforward' refers to the directed graph representation having no loops, or equivalently, that the i-th layer is not the input of the j-th layer for any $j \leq i$.

Definition 2.6. Let $X \subset \mathbb{R}^n$ be a linear space, and $f : X \to \mathbb{R}$ a differentiable function. Then f is **convex** if, and only if,

$$f(x) \geq f(y) + \nabla f(y) \cdot (x - y) \quad \forall x, y \in X$$

where

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}$$

is the gradient of the function at x and $(\cdot)$ is the dot product.

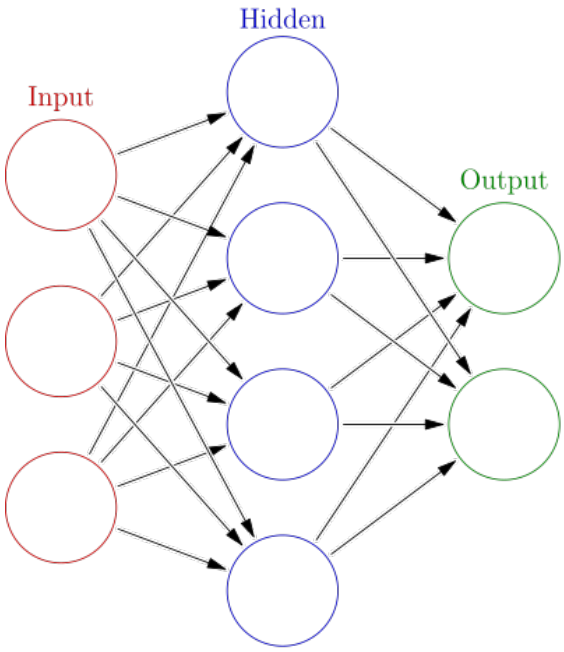

Figure 2.1: Neural network with 9 neurons, one 'hidden' layer and one 'output' layer. [4]

Definition 2.7. Let $X \subset \mathbb{R}^n$, we say that a function $\ell : X \times X \to [0, \infty)$ is a **loss** if

1. ℓ is a convex differentiable function.
2. $\ell(x, y) = 0 \Leftrightarrow x = y$.

In practice a loss function is used to obtain a measure of "how good" the outputs of an Artificial Neural Network are, as compared to the outputs of the target function, and as such will often its definition might vary according to the problem.

Example 2.8. The **mean square error** function defined by

$$MSE(Y, \hat{Y}) := \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y})^2$$

is a common loss used for regression problems.

2.1.1 Optimization

One of the reasons why Artificial Neural Networks have become mainstream in the last decades has been the invention of the back-propagation algorithms [17]. Using backprop-

agation, we can show that, not only the Neural Network is differentiable (with respect to its weights), but we also have an efficient way to calculate it.

Using the gradient of the network with respect to its weights, it is possible to iteratively refine or 'train' the model by updating the weights using the opposite direction of the gradient. The simplest option is Gradient Descent described as

Algorithm 1 Gradient Descent

Input θ the initial weights, X the training samples and Y the training labels; γ a small value.

Output θ the trained weights.

while not converged **do**
    $\theta \leftarrow \theta - \gamma \nabla(\ell(\Phi(X, \theta), Y))$
end while

However, when the dataset is too large, it might not be feasible to evaluate the whole dataset at once.When that is the case, Stochastic Gradient Descent (SGD) is used:

Algorithm 2 Stochastic Gradient Descent

Input θ the initial weights, X the training samples and Y the training labels; γ a small value.

Output θ the trained weights.

for iteration in $\{1, 2, \dots\}$ **do**
    SHUFFLE(X, Y)
    for (x, y) sample of (X, Y) **do**
        $\theta \leftarrow \theta - \gamma \nabla(\ell(\Phi(x, \theta), y))$
    end for
end for

For Stochastic Gradient Descent only one random sample is used at a time to calculate update the weights. A compromise between SGD and Gradient Descent is using small batches (mini-batches) of samples instead of a single sample. Additionally other variations of the SGD exist, in particular the Adaptive Moment Estimation (ADAM) optimizer, which additionally uses the second moments of the gradients.

2.2 Convolutional Neural Networks

Definition 2.9. Let $f, g : \mathbb{R} \to \mathbb{C}$ be complex functions. We define the **convolution** of f and g as

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(\tau - t)d\tau.$$

Suppose $f, g : \mathbb{Z} \times \mathbb{Z} \to \mathbb{C}$ complex functions defined over the 2-dimensional set of integers, then we define the **discrete two-dimensional convolution** of f and g as

$$(f * g)(i, j) = \sum_{(x,y)\in\mathbb{Z}\times\mathbb{Z}} f(i, j)g(i - x, j - y),$$

when the series converges.

Definition 2.10. Let $k \in \mathbb{R}^{n\times n}$ and $I \in \mathbb{R}^{N\times M}$ a real valued matrix, where $n < N$, M is an odd integer, we define the application the convolution of kernel k on M as

$$(I * k)[i, j] = \sum_{(s,t)\in\mathbb{Z}\times\mathbb{Z}} k[s, t] \cdot I[u + s, v + t] = \sum_{s=1}^{n}\sum_{t=0}^{n} k[s, t] \cdot I[u + s, v + t]$$

where $u = i - \lceil n/2 \rceil$, $v = j - \lceil n/2 \rceil$. Notice that $I[i, j]$ (or $k[i, j]$ respectively) is defined as

$$I[i, j] = \begin{cases} I_{ij} & 0 \le i \le N, 0 \le j \le M \\ 0 & \text{otherwise} \end{cases}$$

Remark 2.11. Notice that the preceding definition is equivalent to a two-dimensional discrete convolution as defined in 2.9.

Definition 2.12. We define a **2D convolutional layer** Λ as

$$\begin{aligned} \Lambda : X \times W &\to Y^c \\ (x, \theta) &\mapsto (\psi(x * \theta_1), \cdots, \psi(x * \theta)) \end{aligned}$$

with activation $\psi : Y \to Y$ as non-linear application, and weights θ.

Proposition 2.13. *Let $\Lambda : X \times \mathbb{R}^k \to Y$ be a 2D convolutional layer with associated kernel θ. Then $\Lambda(x, \theta)$ is the composition of a linear function and a non-linearity.*

Proof. Let $\Lambda(x, \theta) = (\psi(f(x, \theta_c)), \cdots, \psi(f(x, \theta_c))) = \Psi(F(x, \theta))$, where Ψ is ψ generalized to c channels, clearly Ψ is a non-linearity and therefore we just need to prove that $F(x, \theta)$ is linear. Since we saw that f is a convolution, and therefore linear. Then F is a composition of linear functions and in consequence $\Lambda = \Psi \circ F$ is the composition of a linear function and a non-linearity. □

Now we will introduce two results found in [12], that might shed some insight on why CNNs have such performance.

Definition 2.14. We introduce the following notation introduced in [12] as

1. Upper-case letters denote a *product* of dimensions: $D = d_0 \times \cdots \times d_N$.
2. Subscripts in a tensor denote a *slice* along a dimension: if $W \in \mathbb{R}^{d \times r}$, W_i is the i-th column of the matrix; if $W \in \mathbb{R}^{n \times m \times r}$, W_i is the matrix corresponding to the i-th slice along the 3rd dimension.
3. For two tensors $W \in \mathbb{R}^{D \times r_s}$ and $Q \in \mathbb{R}^{D \times r_t}$, the tensor $[WQ] \in^{D \times (r_s + r_t)}$ denotes the concatenation along the last dimension.

Definition 2.15. The following definitions will be used in the statement of the theorems:

1. A size-r set of K factors $(W^1, \ldots, W^K)_r \in \mathbb{R}^{(D^1 \times r)} \times \cdots \times \mathbb{R}^{(D^K \times r)}$ is defined to be a set of K tensors where the final dimension of each tensor is equal to r. Notice that the superscript in D^i does not denote exponentiation.
2. A function $\varphi : \mathbb{R}^{D^1} \times \cdots \times \mathbb{R}^{D^K}$ is **positively homogeneous** with degree p if

$$\forall \alpha \geq 0, \quad \varphi(\alpha w^1, \ldots, \alpha w^K) = \alpha^p \varphi(w^1, \ldots, w^K)$$

Remark 2.16. Notice that the second definition of 2.15 implies that $\varphi(0, \ldots, 0) = 0$ for all $p \neq 0$.

Remark 2.17. If f is a linear function, then it is positively homogeneous of degree 1:

$$f(aw^1, \ldots, aw^K) = af(w^1, \ldots, w^k).$$

Remark 2.18. If φ is positively homogeneous with degree p, its composition with the **Rectified Linear Unit** (ReLU) defined as

$$\begin{aligned} \psi^+ : \mathbb{R}^D &\to \mathbb{R}^D \\ x &\mapsto \max(0, x) \end{aligned}$$

where $\max$ is the element-wise maximum, is also positively homogeneous with degree p. Indeed:

$$\begin{aligned} (r \circ \varphi)(aw^1, \ldots, aw^K) &= \psi^+(a^p \varphi(w^1, \ldots, w^K)) \\ &= a^p \max(0, \varphi(w^1, \ldots, w^K)) \\ &= a^p (\psi^+ \circ \varphi)(w^1, \ldots, w^K). \end{aligned}$$

Similarly its composition with **max pooling** defined, using the same notation as 2.10,as

$$m(I)(i,j) = \max_{0 \leq s,t \leq n} I(u+s, v+t),$$

is also positively homogeneous with degree p.

Definition 2.19. Let

- $\Phi_r(S; W^1, \ldots, W^K)$ be the r-sized feedforward ANN of linear layers with weights $W^1, \ldots, W^K$ and activation functions $\Psi_1, \ldots, \Psi_K$ evaluated on an input S as:

  $$\Phi_r(S; W^1, \ldots, W^K) = \Psi_K(\Psi_{K-1}(\ldots \Psi_1(S \cdot W^1 \ldots W^{K-1}) \cdot W^K),$$

  where the size r is the last dimension of the weights $W^1, \ldots, W^K$;

- ℓ is a loss, and $\Theta : W_1 \times \cdots \times W_k \to \mathbb{R}$ is a positive convex and differentiable function (the **regularization function**), that takes into account the number of parameters of the network (implied by the size r);

- λ is a positive scalar that balances the effect of the regularization function against the loss ℓ.

We define the **Classification Problem** [12] as the non-convex[1] optimization problem of the form

$$\min_{r>0} \min_{(W^1,\ldots,W^K)} f_r(W^1, \ldots, W^K),$$

where

$$f_r(W^1, \ldots, W^K) = \ell(Y, \Phi_r(S; W^1, \ldots, W^K)) + \lambda \Theta_r(W^1, \ldots, W^K). \tag{2.2}$$

Theorem 2.20. ***Global optimality from local minima*** *[12] Consider the non-convex classification problem defined in 2.19, in which $\Phi_r(S; W^1, \ldots, W^L)$ and $\Theta_r(W^1, \ldots, W^K)$ can be expressed as the sum of positively homogeneous mapping of the same degree. Then, any local minimizer such that*

$$\exists i_0 \in \{1, ..., r\}, \quad (W^1_{i_0}, \ldots, W^K_{i_0}) = (0, \ldots, 0),$$

is also a global minimizer of the problem.

[1] ANN tend to have several minima

Remark 2.21. Notice that the previous statement is not true, in general, for any ANN. However convolutional layers, with ReLU and max pooling as their non-linear activation functions; with Batch Normalization instead of ℓ_1 or ℓ_2 do fulfil the requirements.

Corollary 2.22. Given a function $f_r(W^1, \ldots, W^K)$ like (2.2), any local minimizer of the optimization problem

$$\min_{(W^1,\ldots,W^K)_r} f_r(W^1, \ldots, W^K)$$

is a global minimizer if $f_{r+1}([W^1 0], \ldots, [W^K 0])$ is a local minimizer of f_{r+1}

Note 2.23. From the preceding corollary we obtain a characterization for the global minima of our optimization problem.

Theorem 2.24. ***Global minima can be found by local descent*** *[12] Given a function $f_r(W^1, \ldots, W^K)$ like* (2.2)*, if $r \leq dim X$ (where X is the output of the ANN) then from any point $(Z^1, \ldots, Z^K)$ such that $f_r(Z^1, \ldots, Z^K) < \infty$ there must exist a non-increasing path from $(Z^1, \ldots, Z^K)$ to a global minimizer of $f_r(W^1, \ldots, W^K)$*

Note 2.25. From the preceding theorem, we can infer that for a network 'large enough', using local gradient descent will eventually lead to a global minimum, regardless of the weight initialization.

Chapter 3

State of the Art

In this chapter we introduce diverse methods that can be applied when training data is very limited. The first methodology we in introduce is data augmentation, then transfer learning and finally few-shot learning.

3.1 Data Augmentation

It is a fact that a limited dataset will negatively impact the performance of Deep Learning algorithms, since the amount of training parameters is often very large. The obvious solution would be to simply increase the dataset. Even though it might not be reasonable—due to the costs or difficulty associated to obtaining them—to request more samples to the provider, we can use techniques to generate synthetic images.

In this project we consider two main classes of transformations: affine transformations and elastic deformations. These transformations are applied to the training dataset randomly, often with random variation of the parameters.

Affine transformations geometrical operations that preserve lines, including: rotations, flips, zooms, shear, displacements and reflection. With affine transformations we attempt to reproduce small variations that would appear when images of the same artery section are captured repeatedly: for example the catheter closer to the artery walls, or a different capture angle.

Elastic deformations, as described by Algo. 3, attempt to simulated the involuntary movement fo the hand when drawing the outlines of the masks by the clinical experts. This approach is shown to improve models that exceed those of elastic deformations [26], when used on hand drawn datasets (such as MNIST).

Algorithm 3 Elastic Deformations

Input $a \in \mathcal{M}(\mathbb{R}^{N\times M})$, $\alpha \in [0,\infty)$, $\sigma \in [0,\infty)$
Output $b \in \mathcal{M}(\mathbb{R}^{N\times M})$

$x \leftarrow \mathbf{0} \in \mathcal{M}(\mathbb{R}^{N\times M})$
$y \leftarrow \mathbf{0} \in \mathcal{M}(\mathbb{R}^{N\times M})$
for all $1 \leq i \leq N, 1 \leq j \leq M$ **do**
    generate random $u, v \in (0,1)$
    $x_{i,j} \leftarrow u * \alpha$
    $y_{i,j} \leftarrow v * \alpha$
end for
$x \leftarrow \text{GaussianBlur}(x, \sigma)$
$y \leftarrow \text{GaussianBlur}(y, \sigma)$
for all $1 \leq i \leq N, 1 \leq j \leq M$ **do**
    $r \leftarrow x_{ij}$
    $s \leftarrow y_{ij}$
    $b_{ij} = a_{rs}$ ▷ a_{rs} is 0 for any indices "outside" the image. (zero-padding)
end for

3.2 Transfer Learning

Data augmentation might not be enough to prevent the model overfitting if the variation of the simulated samples is not large enough. As a tool to deal with this, we introduce **transfer learning**. From a high-level perspective, transfer learning refers to the techniques used to reuse pre-trained models on novel tasks. More specifically, the transfer learning problem can be expressed as finding the transformation from a task $\mathcal{T}_S = (Y_S, f_S)$ of learning a target function $f_S : X_S \to Y_S$, where X_S is the feature space and Y_S label space (for the task); into the task $\mathcal{T}_T = (Y_T, f_T)$ of learning a target function $f_T : X_T \to Y_T$, and either $X_S \neq X_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$ (or both).

Definition 3.1. We define a **feature vector** as a vector associated to an n dimensional vector space (the **feature space**) that represents some object $\mathcal{O}$.

Example 3.2. A feature vector representing an image could be a vector of pixels in $[0,1]^{n\times m}$.

Example 3.3. Examples of transfer leaning are:

1. using a model trained on pictures of cats to detect pictures of dogs (same feature space, different tasks);

2. using a model trained to detect sentiments on written text to detect sentiments on verbal speech (same tasks, different feature space).

It is true that, if the tasks are too diferent, it would not be possible to use the model (e.g. the dimensions of the inputs and outputs might not be compatible), but for the sake of simplicity pre and post processing will be assumed and that the models are Artificial Neural Networks. Additionally we will assume that the task are always classification, but it general this doesn't have to be necessarily true—the methods described can still be applied.

3.2.1 Fine-tuning

The simplest implementation of transfer learning is fine-tuning. Here, we use the weights of a model trained on a task T_S as the weight initializations for our model. Then, the new model is trained on a task $T_D \neq T_S$. Often the model is used 'as is', and a minimum of changes is made. It is common, however, to replace the first and last layers of the model, since those are often tightly coupled to the previous inputs or outputs respectively.

Doing this may help avoiding overfitting the model, by providing weight initializations that require less effort to optimize. Overfitting is a common problem when training with limited data, and describes the phenomenon of a model learning a representation that is very coupled to the training dataset, possibly due to a lack of variability, and then failing to generalize to examples out of that dataset. This is the equivalent of finding a local minimum that does not correspond with the global minimum.

3.2.2 Feature Extraction

Before defining what feature extraction is, we introduce autoencoders. Autoenconders are a class of Artificial Neural Networks. These algorithms attempt to learn the representation of unlabelled data in some feature space, embedding semantic information in the vector representation. A notable example of this is the WORD2VEC[19][18] family of algorithms, which transform the vector v representing a word as the i-th word in the dictionary ($v_i = 1$, $v_j = 0$ if $i \neq j$) into a smaller vector, where similar vectors—for some metric like euclidean or cosine distance—represent similar concepts.

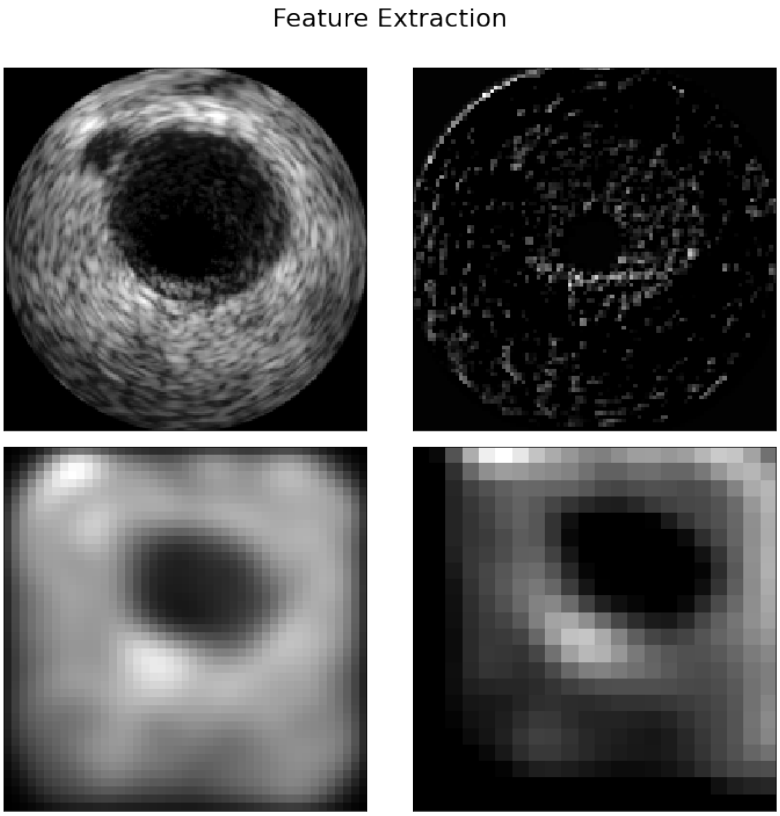

Figure 3.1: Visualization of the outputs of internal layers when using InceptionResnet as an encoder

The concept of **feature extraction** is motivated by the conjecture that, when the model is trained, the internal layers act as autoencoders of the input information, extracting relevant data for the classification task. If we assume the feature spaces for the tasks to be similar enough, in the sense that there is a transformation between them that preserves the semantic information (e.g. resizing images), a model could be used to extract this relevant encodings by acquiring the output of the inner layers. This is often referred as using the model as the **backbone** of another model. 3.1.

By employing feature extraction, we lift the burden of learning this generic features from the new model, thus reducing the amount of parameters. Then the new model can focus the training efforts on learning how to classify the data using the learned features.

This transfer learning method differs from fine tuning mostly in the fact that the backbone model is often relegated to be a sub-network of the main model. Often, the

weights of the backbone are frozen, i.e. not trained. This lowers the amount of trainable parameters and might be desirable in some cases.

3.3 Few-shot Learning

Sometimes, our source model needs to be reused for a set of tasks, for which we have a very limited amount of data. This means that, a priori, we don't know the task for which the model must be trained. One example of this is shown in [21], where the model must be prepared to learn a sinusoidal function, but the parameters are unknown.

The previous techniques, fine tuning and feature extraction, can also be used to generate models for the target tasks. However in cases of extreme data scarcity, with as low as 1 training examples, those methods might not be enough to prevent severe overfitting.

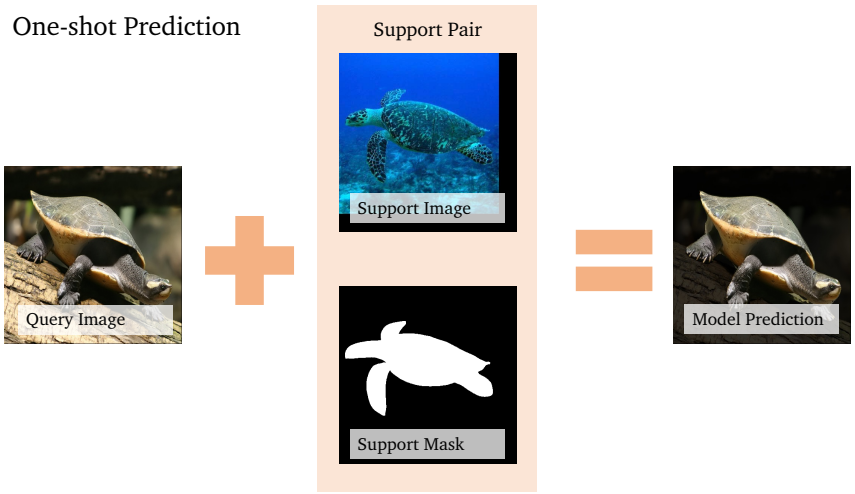

Figure 3.2: Outline of one-shot prediction scenario. The class 'turtle' is a novel class and the model is provided with a support image and mask as part of its input.

The most extreme case of few-shot learning is one-shot learning, illustrated in Fig. 3.2. Here, we show the scenario of a model learning to segment turtles in an image with only an image–mask pair as context. Notice that for the few-shot segmentation problem we asume that the new classes are unseen—in Fig. 3.2 the model has never 'seen' a turtle. The methods reviewed in this section are used for both few- and one-shot learning, and

those concepts might be used interchangeably.

In the literature we find two main approaches to solve the few-shot problem: **meta-learning** based methods [3][9][11], and **metric** learning methods [28][27][36].

3.3.1 Meta-learning

Meta-learning approaches the few-shot problem by learning weight initializations that can be used to learn quickly—that is in few steps—from the training dataset, which is assumed to be very small. Informally, the training task for the model is 'learning to learn'. More concretely, given an initial subset of training tasks $\{\mathcal{T}_i\}_{i\in I}$ (or classification classes) we want to learn weights, such that for any task (resp. Classification class $\mathcal{T}_j$, $j \notin I$, we learn a model in as few steps as possible.

Algorithm 4 REPTILE

Input subset of tasks $T = \{\mathcal{T}_i\}_{i\in I}$, θ vector of initial weights,

1: **for** step in $\{1, 2, \dots\}$ **do**
2: select $\mathcal{T} \in T$ with associated weight $\ell_{\mathcal{T}}$.
3: compute $\tilde{\theta}$ the updated weights after k steps of SDG or ADAM.
4: $\theta \leftarrow \theta + \epsilon(\tilde{\theta} - \theta)$
5: **end for**

[21] propose the REPTILE algorithm described in Alg. 4. As opposed to the approach followed in fine-tuning, the model is trained on a distribution of tasks with their associated datasets. This method is able to learn weight initializations that suit the distribution of tasks—as opposed to one task in the distribution. In some examples, such as the sinusoidal function learning, a model using weights pre-trained using REPTILE performs better and can be trained in less iterations than a model using traditional weight intializations [21].

3.3.2 Metric Based Learning

A different approach is **metric based** learning. Here the goal is to learn a representation of the input data in an appropriate feature space, that is then used along some kind of distance function to predict the labels based on proximity to samples in the support set.

Prototypical Networks

Prototypical Networks [27] attack the few-shot problem for classification by assuming that, for every class, there is a prototype that represents the class. Then, classification can be done by finding the closest prototype.

Given an embedding function $f_\Phi : \mathbb{R}^n \to \mathbb{R}^m$, the prototype for a support set S_k is

$$c_k := \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\Phi(x_i).$$

Then a distribution over the classes for the query point x is generated using a distance d and soft-max as

$$\boldsymbol{P}_\Phi(y = k|x) = \frac{\exp(-d(f_\Phi(x), c_k)}{\sum_{k' \in K} \exp(-d(f_\Phi(x), c_{k'}))}$$

where K denotes the set of prototype classes.

PFENet

In our results we will show an application of the Prior Guided Feature Enrichment Network (PFENet) [29] applied to semantic IVUS segmentation. PFENet is closely related to prototypical networks, and extends the model proposed in CANet [35]. Both models use a backbone—ResNet50[14] in our case—to extract features from the query and support images.

In general features in the middle levels are preferred since it is conjectured that high level features are more class specific[29], however PFENet exploits these high-level features to provide semantic segmentation cues from the support pair.

From the high level features, PFENet constructs a prior mask, i.e. a mask that contains the probability of pixels belonging to a class. Let us define the support pair $S = (X_S, Y_S)$ and the query pair $Q = (X_Q, Y_Q)$. The segmentation masks Y_S and Y_Q are binary masks where 1 is the indicator of the mask and 0 denotes background. The target mask Y_Q is unknown to the model.

From the support pair and the query image, we compute the high level features, extracted from the backbone, as

$$\Phi(X_Q), \quad F_S := \Phi(X_S) \circ Y_S, \tag{3.1}$$

where $\circ$ denotes the Hadamard product. Eq. (3.1) sets the support features of the background to 0, because Y_S is a binary mask.

Then, for each query pixel $x_{ij} \in F_Q$, we compute

$$p_{ij} := max_{x_{rs} \in F_S} \cos(x_{ij}, x_{rs}),$$

using the cosine similarity $\cos$. Notice that x_{ij} and x_{rs} are vectors with elements for every channel in the network output, and c_{ij} are scalars.

Finally the prior mask is constructed as $P_Q = \{p_{ij}\}$, and it is normalized using min-max normalization

$$P_Q = \frac{P_Q - \min(P_Q)}{\max(P_Q) - \min(P_Q) + \epsilon},$$

for some small epsilon—set to 10^{-7}.

Now we have the input query image X_Q, the support image–mask pair (X_S, Y_S) and the prior mask P_Q. [29] propose a feature enrichment module (FEM) with the purpose of refining the query features using the prior and the support features.

A sketch of the process is:

1. **Inter-source Enrichment**: where the input is projected to different scales and interacted with the support features and prior mask independently.

2. **Inter-scale Interaction**: where essential information between query–support features is selectively passed across scales.

3. **Information Concentration**: where the features in different scales are merged to yield the final refined query feature.

The PFENet model is trained using what is called 'episodic training': the training task is split into 'episodes' of support S and query Q subsets of the same class. Then the model is evaluated using the support image–mask pairs along the query image, and the error is evaluated using the query mask.

Chapter 4

Methodology

All the models, and both training and evaluation code are implemented in PYTHON 3.9 using OPENCV 4.5, TENSORFLOW 2.8, KERAS 2.8 and PYTORCH 1.11 . For the research and experimentation we use a computer running UBUNTU 20.04 LTS with an NVIDIA GEFORCE GTX 960 graphics card.

For the task of model comparison, we will train on the first 109 frames of dataset B. An exception to this is the few-shot task, which will use the first frame of the first patient. The models will be trained using a train–validation split of 90/10, and we will generate a distribution of models using K-Fold crossvalidation (for $K = 3$).

4.1 Data

For training and evaluation purposes, we use the datasets provided by [2].

Dataset A (or Boston) contains 77 images extracted from *in vivo* pullbacks of human coronary arterires, from 22 patients. The hardware used to acquire them is an iLab IVUS, from Boston Scientific. Frames are taken at 40MHz and are displayed as gray-scale PNG images with a resolution of 512×512 pixels. For this dataset, frames are not consecutive and acquired at random instances of the cardiac cycle. This dataset will be used in its totality as a training set to benchmark few-shot segmentation.

Dataset B (Volcano) contains a larger set of 435 images extracted from *in vivo* pullbacks, from 10 patients. The imaging system used is a Si5 from Volcano Corporation, with frames being taken at 20MHz. Each frame from this dataset is provided with four adjacent frames to provide volumetric insight. The provided images are gray-scale PNG images with a resolution of 384×384. The first quarter of the dataset (109 frames) are

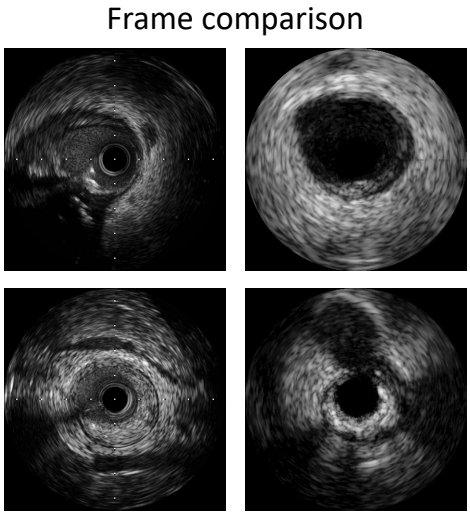

Figure 4.1: Left frames from Dataset A; right frames from Dataset B

used for training, and the rest for evaluation.

The images of the dataset have been labelled (see Fig. 4.3) by two clinical experts, one of them labelling the set twice. For our training and testing purposes we select the second labelling from the expert that labelled twice.

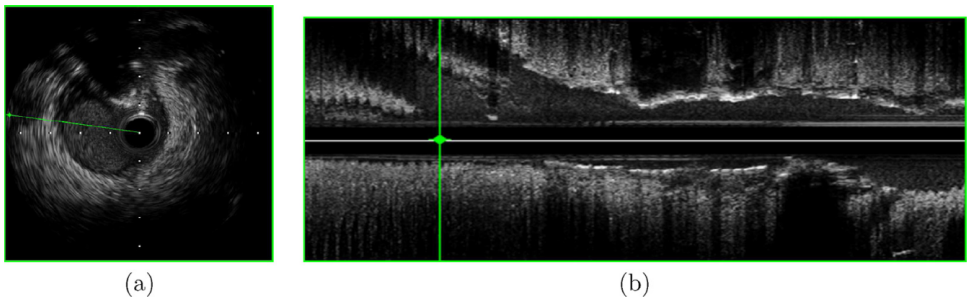

Figure 4.2: Short axis (a) and longitudinal (b) view of the same pullback [2]

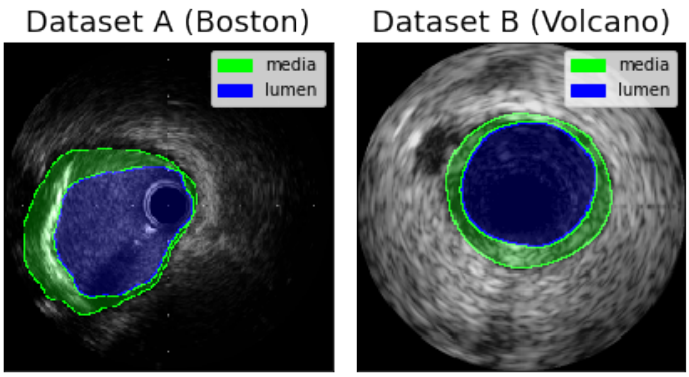

Figure 4.3: Example of IVUS images and the respective segmentation masks for media and lumen.

4.2 Preprocessing

Datasets A and B provide us with IVUS frames and outlines for the media and lumen. The frames are stored as gray-scale Portable Network Graphics (PNG) with labels containing (1) the patient they were extracted from, (2) the frame index in the pullback, (3) the frame index in the sample. As per [2], the samples are provided as groups of 5 frames. To construct the training samples, we read the 5 PNG images corresponding to a sample and concatenate them channel wise as $N \times N \times 5$ tensors (where $N = 512$ for the Boston dataset, and $N = 384$ for the Volcano dataset). Of this we discard the two furthermost frames from the central frame for memory economy reasons.

The labels are stored as lists of 2D coordinates corresponding to vertices of a convex hull. We reconstruct the labels as binary masks for the media and lumen by filling the convex hull (Fig. 4.4). Labels for each sample are then represented as the one-hot encoding of the class of each pixel. That is, for each pixel we identify a vector v such that $v_k = 1$ if the pixel is of the class k, and 0 everywhere else, the classes being background ($k = 0$), media ($k = 1$) and lumen ($k = 2$). Therefore we construct a $N \times N \times 3$ tensor where the 'ones' in the 3rd layer are in convex hull corresponding to the lumen, those in the 2nd layer the points in convex hull corresponding to the media not contained in the lumen, and those in the 1st layer are the complement of the other layers.

An advantage presented by this encoding is that we can use the 'confidence' of the

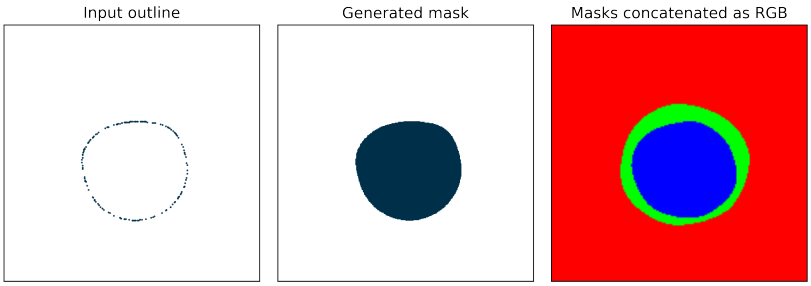

Figure 4.4: Left input vertices (lumen), center binary mask (lumen) and right one-hot encoding as RGB channels.

model for each class (in the interval $[0, 1]$) instead of the true class label for training. The true class can be obtained simply as

$$\text{class}(y_{ij}) = \max\{k \in \{0, 1, 2\} | v_k\}$$

using the previous definition for v.

For the preprocessing of the query images, we resize the images to half their height: 192 pixels for dataset B and 256 for dataset B. Then we perform min-max normalization on the training images $x^{(i)}$, as

$$x^{(i)} \leftarrow \frac{x^{(i)} - \min(x^{(i)})}{\max(x^{(i)}) - \min(x^{(i)})}.$$

4.3 Data Augmentation

In this report we analyse the effects of some data augmentation on IVUS imaging. The methods used will be affine and elastic transformations as described in §3.1.

From affine segmentation we select a subset of operations: rotations, horizontal and vertical flips, and zooms. Notice that other affine transforms exist, including shear and reflection transforms, but after preliminary analysis it was clear that those lead to invalid images.

Additionally we introduce elastic deformations (Algo. 3) to simulate the involuntary movement of the hand when drawing the mask. These will be applied to the training masks (and not the image).

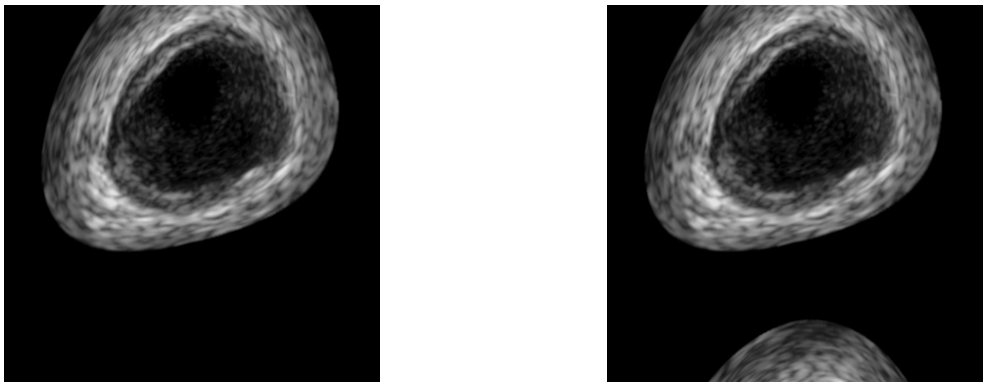

Figure 4.5: Left: image deformed with zero padding. Right: image deformed with reflected padding. This transformations are exaggerated and aren't used for training.

It is important to consider that, when augmenting our dataset careful testing of the data augmentation is required. A model trained on examples that aren't realistic is likely to severely underperform. Finding the right transformations and tuning their parameters is challenging, and to a large extent depends on intuition and visual analysis. It is possible to use trained models on the original dataset to predict the quality of the augmented dataset, but it is still not a trustworthy method we can rely on.

One example of 'invalid transformations' was found exploring the implementation used by [25]. Upon analysis of the generated images, we discovered that the algorithms used 'reflection padding', i.e. the pixels outside the image are obtained by reflecting the original image. However, as seen in Fig. 4.3, this leads to impossible images. In our implementation we use zero padding, i.e. the pixels outside are assumed to be black, because this leads to images that are closer to the original images.

The parameters for the transformations used in our experiments are:

1. zoom range of 20%;
2. rotation range of 0.2π;
3. $\alpha = 2.5$ and $\sigma = 0.24$[1] as parameters of the elastic transforms.

[1] an example of how the parameters affect the images is in Fig. A.1

4.4 Evaluation Metrics

In the original article for the datasets [2], the Jaccard measure (JM), the Difference of Area Percentage (DAP) and Hausdorff distance (HD) are used to evaluate the performance of models on the dataset. Out of these, we select the Jaccard distance (obtained from the Jaccard index), and Hausdorff distance.

In this section we define, and comment the differences of this two losses and what insight they provide. In [25], the Dice coefficient is also used, but, since it can be derived from the Jaccard index we do not consider it.

Definition 4.1. The **Jaccard Index**, Jaccard Measure or sometimes Intersection over Union, compares the similarity of the ground truth and predicted sets as the cardinality of the intersection over the cardinality of the union:

$$\mathrm{J}(A, B) := \frac{|A \cap B|}{|A \cup B|}$$

Clearly, the maximum of this function, $\mathrm{J}(A, B) = 1$, is achieved if, and only if, $A = B$, and the minimum is $\mathrm{J}(A, B) = 0$ whenever A and B don't intersect.

Definition 4.2. From the Jaccard Index, we define the **Jaccard Distance** as

$$d_J(A, B) := 1 - \mathrm{J}(A, B)$$

Remark 4.3. The Jaccard Distance is a metric in the set of finite subsets [16].

Definition 4.4. The **Hausdorff Distance** is defined over pairs of subsets of a metric space (M, d) as

$$\mathrm{HD}(A, B) = \max\{\mathrm{hd}(A, B), \mathrm{hd}(B, A)\},$$

where

$$\mathrm{hd}(A, B) = \sup x \in A \inf_{y \in B} d(x, y)$$

is the one-sided Hausdorff distance. Intuitively the one-sided Hausdorff distance is given by the point of A that is furthest away from all the points in B (and viceversa).

Remark 4.5. The Hausdorff distance 4.4 is a metric over the set of non-empty closed sets. In particular it is a metric over the set of non-empty finite sets.

The Hausdorff distance is sensitive to "noisy" predictions, i.e. masks with a 'main' component close to the target mask, and smaller components at a distance. An example

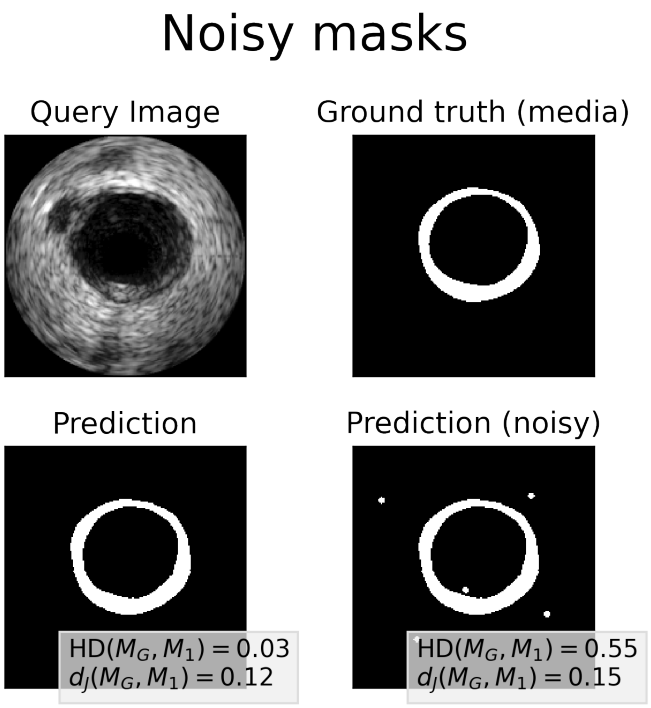

Figure 4.6: Comparison of two sample masks (without M_0 and with noise M_N) against the ground truth M_G.

of this can be observed after studying the distances of the predictions in 4.6, where we observe an increase of over 2000% in terms of the Hausdorff distance after adding the noise. The Jaccard distance, however, only increases a 26.12%. This makes sense since the effect of noise on the intersection and union is minimal.

Additionally, as we see from remark 4.5, the Hausdorff distance the sets to be non-empty. This is reasonable, since for empty sets the distance is not defined. In our evaluation metric we artificially set it to the length of the diagonal of the masks. On the other hand, the Jaccard distance is set to 1 when the intersection is empty, and in particular when any set is empty.

Despite the limitations mentioned above, the Hausdorff distance is able to encode information about the shape of the masks—as opposed to the Jaccard index which only considers the intersection. The Hausdorff distance is often consistent with the intuition of which mask is better from a qualitative point of view, but the Jaccard distance might lead to surprising results.

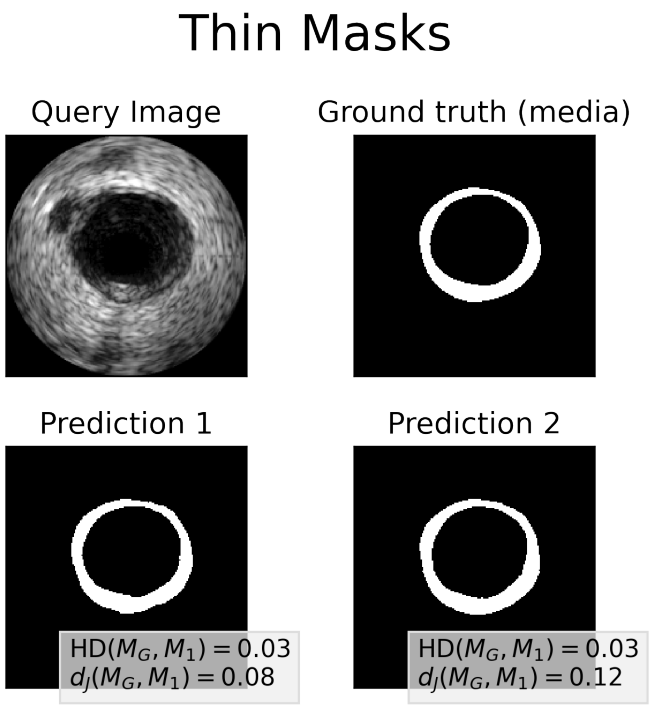

Figure 4.7: Comparison of two perceptually identical sample masks and their respective Hausdorff and Jaccard distances.

We illustrate this in Fig. 4.7. Here we show two perceptually identical masks as prediction examples, and yet the Jaccard distance shows significant variation. Hausdorff distance, however, remains low and with a similar value. Our conjecture is that when the masks are "thin" like we see in Fig. 4.7, small differences in the intersection will overly influence the Jaccard distance.

From the previous analysis, we conclude that both metrics measure different things, and there is value in considering them both when it comes to evaluating our model. However, considering how examples like the one shown in 4.7 are qualitatively almost indistinguishable, and we would expect the metrics to reflect this. That is only true in the case of the Hausdorff distance, and therefore we conjecture that it could lead to higher performance.

4.5 Loss Functions

The choice of loss function determines to a certain extent the task that the model is trying to optimize. The segmentation problem is eminently a classification problem between labels and predictions, even if a (label) prediction is cast for each pixel in the query image. Also, in our case we have to consider that we are actually predicting three classes: the media, the lumen and the background.

Often in the literature [5], some variation of cross-entropy minimizing loss is seen as a 'catch-all' loss for this category of problems. However, in our case we do have target measures, so it is natural to consider them as the foundation for our loss. When using a loss constructed from our metrics we should expect to obtain better qualitative and quantitative results. Both the Jaccard distance and the Hausdorff distance are minimized as the sets become more similar, and as such we can use them as replacements for the cross-entropy loss with minimal changes.

We conjecture, however, that a loss based on the Hausdorff loss will lead to improvements against a loss based on the Jaccard distance. In fact we show two theoretical results: Lemma 4.7 and Lemma 4.9.

Definition 4.6. Let A be a set in a metric space (M, d), we define the **generalized ball** of radius ε as

$$U_\varepsilon(A) := \bigcup_{x \in A} \{y \in M | d(x, y) \leq \varepsilon\}.$$

Lemma 4.7. *Let (M, d) be a metric space, and define $F(M)$ the set of finite closed non-empty subsets of M. Let $X \in F(M)$. Define for all $k > 0$, $0 < h \leq 1$, the set*

$$P_{k,h} = \{Y \in F(M) | \operatorname{HD}(X, Y) \leq k, |X \cap Y| \geq h|X|\}.$$

Then, for every $0 < h \leq 1$,

$$\begin{aligned} m_h : (0, \infty) &\to [0, 1] \\ k &\mapsto \sup \{d_J(X, Y) | Y \in P_{k,h}\} \end{aligned} \tag{4.1}$$

is monotonic and

$$m_h(k) \xrightarrow{k \to 0} 1 - h$$

Proof. If $Y \in P_{k,h}$ then

$$d_J(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|} \leq 1 - h\frac{|X|}{|X \cup Y|}.$$

Now we show that $|X \cup Y|$ can be bound by a function of k. Let $\mathrm{HD}(X, Y) = \varepsilon \leq k$. Then we know that

$$\mathrm{hd}(X, Y) \overset{X \text{ and } Y \text{ are closed}}{=} \max_{y \in Y} \min_{x \in X} d(x, y) \leq \varepsilon. \tag{4.2}$$

Consider X_ε the generalized ball of radius ε around X. From (4.2), every $y \in Y$ is contained in the ball of radius ε of some $x \in X$. Therefore

$$Y \subseteq X_\varepsilon \implies X \cup Y \subseteq X_\varepsilon.$$

Now define $c(\varepsilon) := |X_\varepsilon|$, then $|X \cup Y| \leq c(\varepsilon)$. From (4.1) $\varepsilon \leq k$ and finally

$$d_J(X, Y) \leq 1 - h\frac{|X|}{c(k)} =: m(k).$$

Clearly $c(k)$ is monotonic increasing w.r.t. k, $h\frac{|X|}{c(k)}$ is monotonic decreasing and, because

$$1 > h\frac{|X|}{|X \cup Y|} \geq h\frac{|X|}{c(k)} > h,$$

$m(k)$ is monotonic increasing. Therefore for a fixed h, $m_h(k) \xrightarrow{k \to 0} 1 - h$.

□

Remark 4.8. Notice that the converse statement for Lemma 4.7 does not hold. Indeed, even if we fix the intersection, it is not possible to provide a bound for the Hausdorff distance using the Jaccard distance. We illustrate this with a counterexample. Let $S = X \cap Y$ be the intersection, we construct Y' as $S \cup Z$, where Z is any set that does not intersect X such that $|Y'| = |Y|$. Clearly $d_J(X, Y) = d_J(X, Y')$. However we can make the Hausdorff distance $\mathrm{HD}(X, Y')$ arbitrarily large, independently of the value of the Jaccard distance and the cardinality of the sets.

Lemma 4.9. *Let (M, d) be a metric finite space and define $F(M)$ the set of closed non-empty subsets of M. Let $X \in F(M)$ and define*

$$Q_k := \{Y \in F(M) | k < |Y|\}.$$

Then for all $k > 0$, there exists some $\varepsilon > 0$ such that

$$\mathrm{HD}(X, Y) \leq \varepsilon \implies X \cup Y \neq \emptyset \quad \forall Y \in Q_k. \tag{4.3}$$

Proof. Consider k fixed, $Y \in Q_k$ and $\varepsilon > 0$. We know that

$$|X \cup Y| + |X \cap Y| = |X| + |Y| \geq |X| + k.$$

From the proof of lemma 4.7, we can substitute

$$|X_\varepsilon| + |X \cap Y| \geq |X| + k \implies |X \cap Y| \geq |X| + k - |X_\varepsilon|.$$

Then, eq. (4.3) is equivalent to say that we can select some $\varepsilon_0 > 0$ such that $k > |X_{\varepsilon_0}| - X = |X_{\varepsilon_0} \setminus X|$. We set $\varepsilon_0 = \sup\{\varepsilon > 0 \mid k > |X_\varepsilon \setminus X|\}$.

□

Lemma 4.7 leads us to believe that, if the intersection of the prediction and target mask are large enough, minimizing the Hausdorff distance is guaranteed to minimize the Jaccard distance. This is a reasonable requirement since the model is expected to intersect the target as much as possible anyway.

From 4.9 we learn that, under some conditions of cardinality for the prediction, a Hausdorff distance low enough will eventually lead to intersection of the target and prediction. This reaffirms our intuition that the model will eventually intersect predictions with targets.

Unfortunately, calculating the Hausdorff distance between two sets comes at a high computational cost and high memory requirements, since the naive implementation requires quadratic memory allocation (compared to the inputs).

To correct the performance requirements of the Hausdorff distance, we find an approximation of the Hausdorff distance. Similarly to [15], we use an approximation based on distance transforms.

Consider $Y, \hat{Y} \in \mathcal{M}^{n \times m}(\{0, 1\})$ the target and predicted 2D binary masks, where 0 represents the background and 1 the foreground. We define the distance transform as

$$\begin{aligned} T : \mathcal{M}^{n \times m}(\{0,1\}) &\to \mathcal{M}^{n \times m}(\mathbb{R}^+) \\ Y_{ij} &\mapsto \min_{k,l|Y_{kl}=1} d([i,j],[k,l]) \end{aligned}$$

Then, using the distance transform, we construct the loss as:

$$L_{DT}(Y, \hat{Y}) := \frac{1}{|\Omega|} \sum_{\Omega} (Y - \hat{Y})^{(2)} \circ (T(Y)^{(\alpha)} + \hat{Y}), \tag{4.4}$$

where $A^{(\alpha)}$ indicates element-wise exponentiation and $\circ$ the Hadamard or element-wise product, and Ω is the set of indices. Notice that, in contrast to the original algorithm by

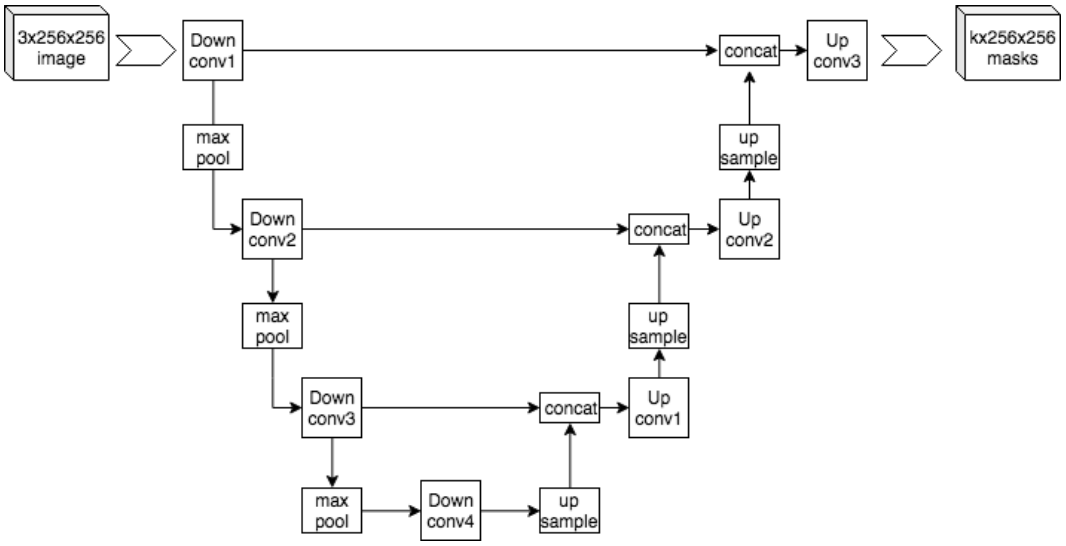

Figure 4.8: Example of U-Net architecture [34].

[15], in (4.4) we add Y to the distance transform. The intuition behind that is that we also would like to have a non-zero loss when the predicted mask is strictly contained in the ground truth.

4.6 Network Architecture

The model selected for the first learning task is based on the U-NET [24] architecture. U-NET is a fully convolutional neural network. It is composed by a sequence of dimensionally reducing convolutions and max pooling layers that aim to extract features of different scales, followed by a sequence of deconvolutions that reuse the features extracted at every layer Fig. 4.8.

In order to increase the performance of the model, we will use a larger model through feature extraction to improve the feature extracting capabilities of our U-NET model. Our choice of network is INCEPTION RESNET, for which weights pre-trained on a large dataset [6] are available on the KERAS platform. With the pre-trained model, we will use the output of layers in the low, mid and top part of the network as feature extractors, replacing the convolutions in the U-NET. From now on we will refer to this model as the RESNET architecture.

These features, particularly for the middle layers, are thought to be task agnostic. This

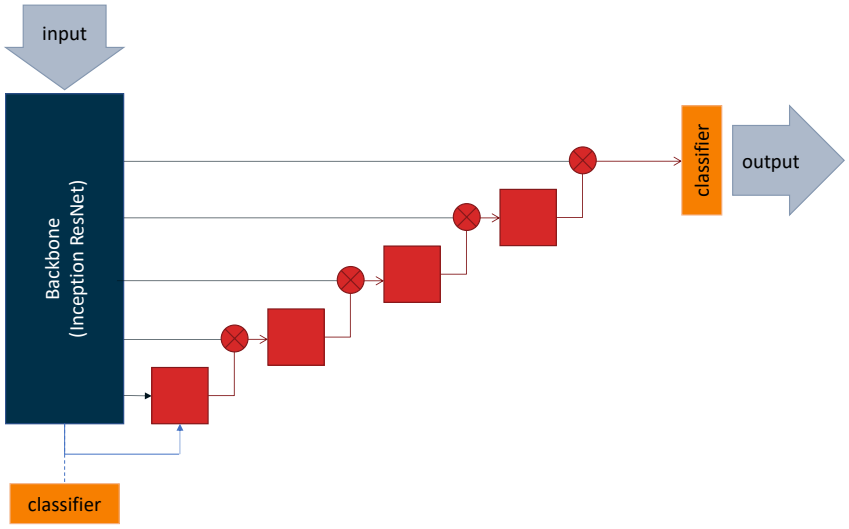

Figure 4.9: In blue the backbone encoder, in red the added convolutional blocks and orange classifier layers.

could prove very helpful in our problem, since we want to minimize the effort spent on learning task agnostic features. Often, when applying feature extraction, the backbone is "frozen" to reduce the amount of training weights. After preliminary testing, we discovered that fine-tuning our backbone leads to better performance. Our hypothesis is that our images are quite different from those in [6]: the channel contain volumetric instead of color information, and the ultra-sound images are significantly different to those taken with a digital camera.

4.6.1 Few-shot Scenario

Our proposed few-shot model is largely based on the Prior Guided Feature Enrichment Network (PFENet)[29] (see §3.3.2 for more details) using the ResNet50 architecture [14] as the backbone, which in turn is pre-trained on ImageNet [6].

Initially, our intention was to use the provided weights for the model, trained on the PASCAL VOC 2012 [8] dataset, and benchmark the performance of the network using just one example. Unfortunately the predictions weren't as expected, and we conjectured that, since the model was trained using RGB images taken with a camera as the training

set, it was failing to extract features of gray-scale IVUS images.

The proposed solution is modifying the model training routines to accept the images from dataset A, and using them to fine-tune PFENET to hopefully learn to understand IVUS images, and then use those weights as the basis for our experiments.

More specifically, the model was trained using the lumen of dataset A as the training classes, and the media as the validation class, using a 80/20 split of the dataset. The reason for not training both lumen and media is that, per [29], we want to use a different class for validation—instead of different samples for the same class.

For this model a fine-tuning step on the query dataset (Dataset B) will not be necessary. Instead, along the predictions, the first frame of patient 01 will be provided to the network, which will use it to extract features from the query frames (see §3.3.2). Notice that this frame is included in the designated training fourth of the dataset, and therefore evaluation can be performed on the same test examples without risk of overlap.

4.7 Statistical Significance

For our quantitative results, we compare the average performance of the models described above. In order to do so, we will employ Student's t-test. Our results will be computed as the average of the loss of the predictions against the ground truth, in terms of the Jaccard and Hausdorff distance.

We evaluate a total of four metrics for each sample: Hausdorff and Jaccard distances for the media and lumen segmentation. The one-shot models in §4.6.1 will be the exception; only Hausdorff and Jaccard distances for lumen segmentation will be evaluated.

Our training set is non-standard in the sense that Dataset samples are groups of five frames, of which we select a subset of three. Because of this, we can't fulfil the requirement of independence. The frames however are 2-dependent, since a example will overlap with at most 3 frames.

Using the results in [23], we calculate the variance of m-dependent random variables as

$$v_m := \gamma(0) + 2\sum_{j=1}^{m}\gamma(j)$$

where $\gamma(0) := \mathrm{Cov}(X_t, X_{t+h})$.

Another factor to consider is that models evaluated on the same set are clearly not independent. However, the ordered pairs $(\hat{Y}_1^i, \hat{Y}_2^i)$ are 2-independent, where $\hat{Y}_1$ is the output of the model 1 and $\hat{Y}_2$ the model 2 evaluated on the i-th frame.

Chapter 5

Results

In this chapter we will analyse the results obtained after evaluating the models and proceed to compare them.

5.1 Data Augmentation

The first goal of interest is testing whether the transformations described in 14.3 lead to improvements. It is mention in that section that there is a possibility of using parameters or transformations that do not represent the reality and worsen the performance of the model. We will use the bare U-Net model as our benchmark. It does not use any form of transfer learning so we expect it to only learn from the dataset. As the training loss we use the Jaccard loss, because we know from [25] that it has good performance. As our evaluation metric we have chosen Hausdorff distance.

In Fig. 5.1 we observe a clear improvement when using either transform, particularly on the precision of the model for the segmentation of the media. It is not clear, however if either of the data augmentations—affine transforms or elastic deformations—are preferable when it comes to training. However,

Then, we evaluate the models using Inception ResNet as a backbone. The differences are less extreme, as seen in Fig. 5.2, but we can still see improvements when using data augmentation. In particular, elastic deformations seem to lead to better performance for the media segmentation task. Notice that this contradicts results from [25].

After this results, since we have seen that both affine and elastic transformations lead to improvements, both will be used in combination when training the subsequent models.

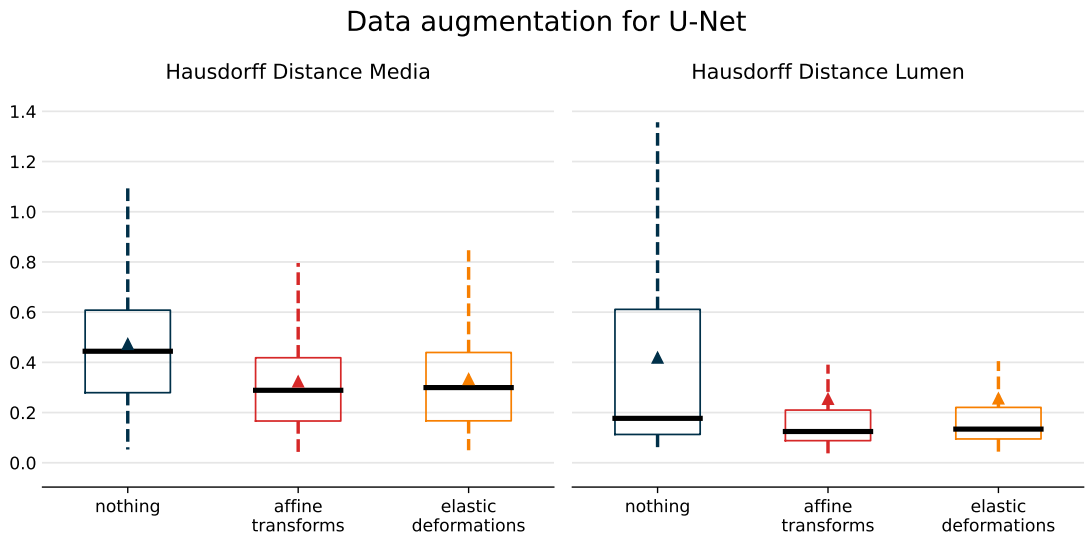

Figure 5.1: Effects of data augmentation on the U-Net network.

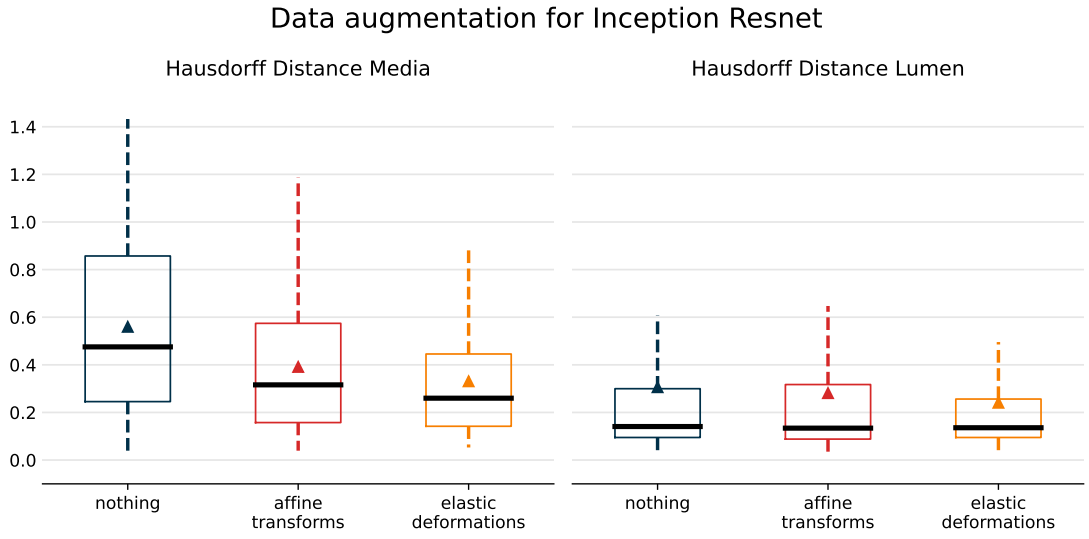

Figure 5.2: Effect of data augmentation when using feature extraction.

5.2 Hausdorff Loss

In equation (4.4) we introduce a training loss based on distance transforms. We would like to know if using this loss variation will increase the performance of the model the

same way the Jaccard loss did against the categorical crossentropy loss[25].

Let there be two identical models, using the architecture described in §4.6, except for their training loss, where the model ResNet$_J$ is trained using the Jaccard loss and ResNet$_H$ is trained using our Hausdorff distance approximation. Both models are trained using the same preprocessing steps; data augmentation is applied as mentioned in the previous chapter (a combination of affine transformations and elastic deformations applied randomly as describe in §4.3).

Then our hypotheses are:

1. the distribution of the means for ResNet$_H$ will have a lower mean Hausdorff distance on the test set—since we are optimizing an approximation of the Hausdorff distance.

2. based on 4.9, we expect to not see an increase on the mean Jaccard distance on the test set for ResNet$_H$ (with respect to ResNet$_J$).

Additionally we expect to see an improvement in the quality of the segmentation masks from a qualitative point of view.

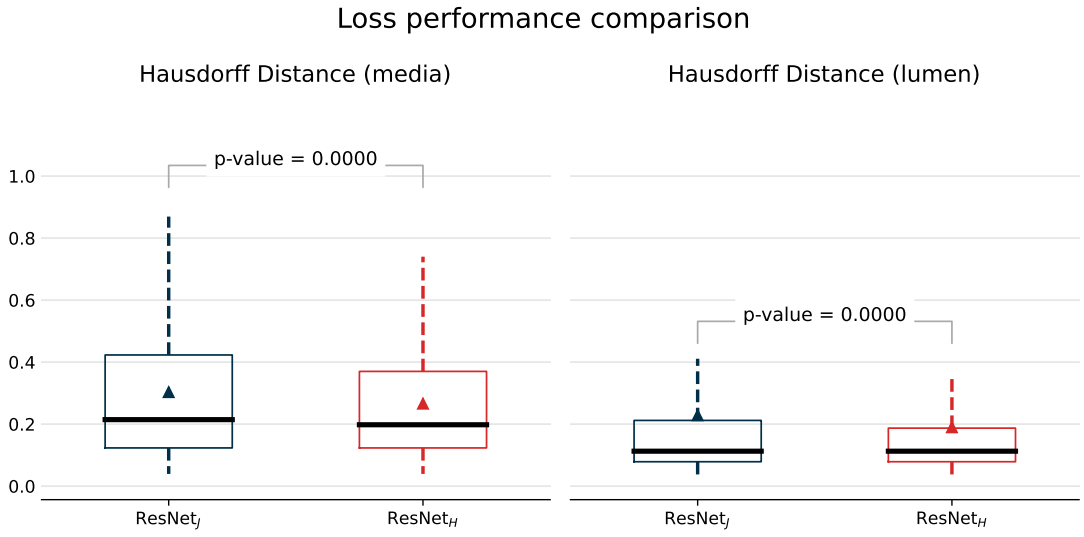

Figure 5.3: Comparison of the ResNet$_J$ and ResNet$_H$ models, evaluated using Hausdorff distance.

Indeed, we find a decrease of the error (Fig. 5.3), in terms of the Hausdorff distance, for the segmentation of the media and lumen of 13.98% and 20.45% respectively when

using ResNet$_H$; this difference is statistically significant (p-value < 0.01). At the same time, supporting hypothesis 2, we fail to reject the null hypothesis for the alternative of the loss of model I having a lower population mean, with p-values of 0.279 and 0.949 for the segmentation of the lumen and media, respectively.

In Section 4.5 we conjectured that the Hausdorff metric embedded some geometric information about the mask, and that masks that were visually similar tended to have similar Hausdorff distances to the query mask. To, qualitatively, test this hypothesis we selected the 5 predictions with the worst score for each model, and then we compared them visually.

From inspection it seems that the ResNet$_H$ is less likely to mistake shadows for the lumen (see: Fig. 5.4, rows 3 and 4; Fig. 5.5, rows 4 and 5). It also appears like the predictions of ResNet$_H$—trained using an approximation of the Hausdorff distance—tend to spread less. Interestingly, frames 24 and 1 for patient 7 and 8 respectively, appear in both cases. This could be a consequence of these frames being "difficult" frames to segment in general.

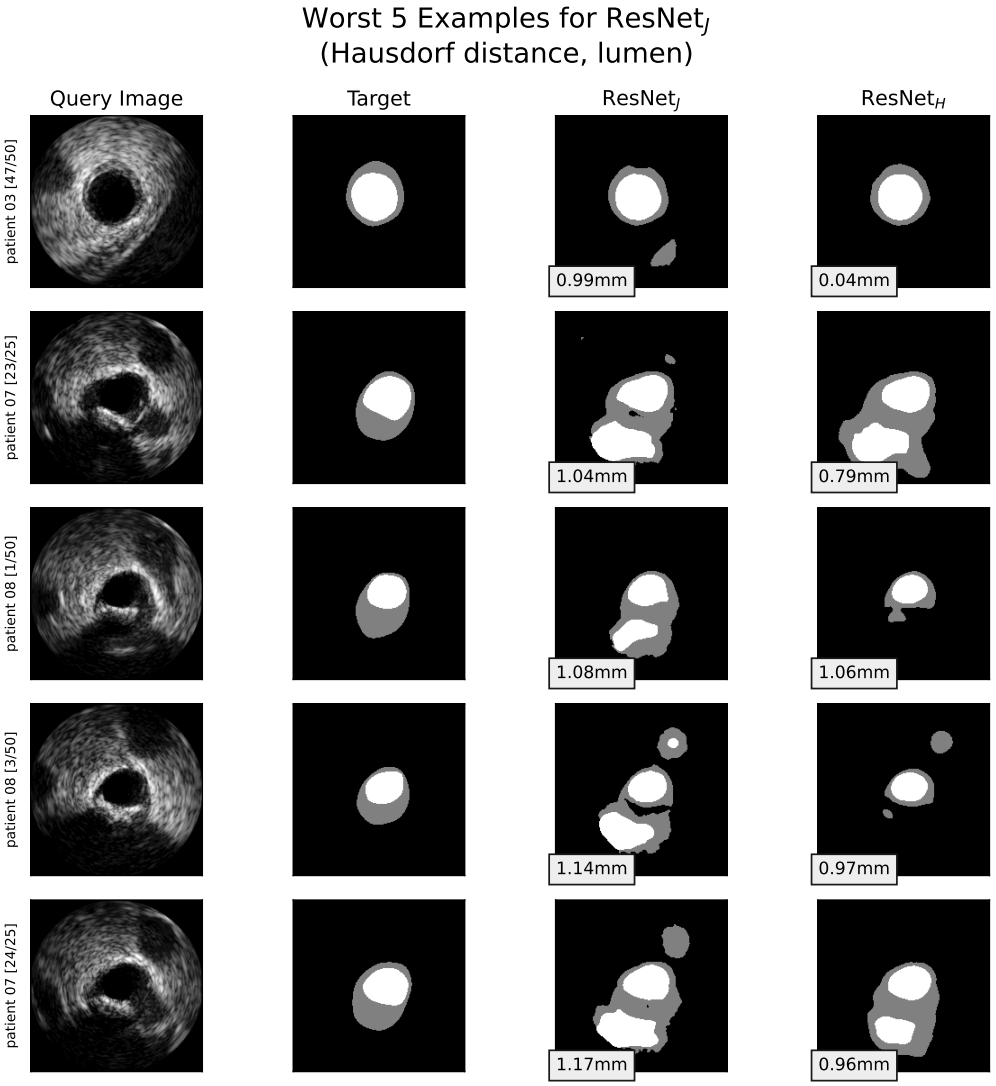

Figure 5.4: Comparison of the predictions from ResNet$_J$ and ResNet$_H$ for the 5 samples the ResNet$_J$ obtains the worst score.

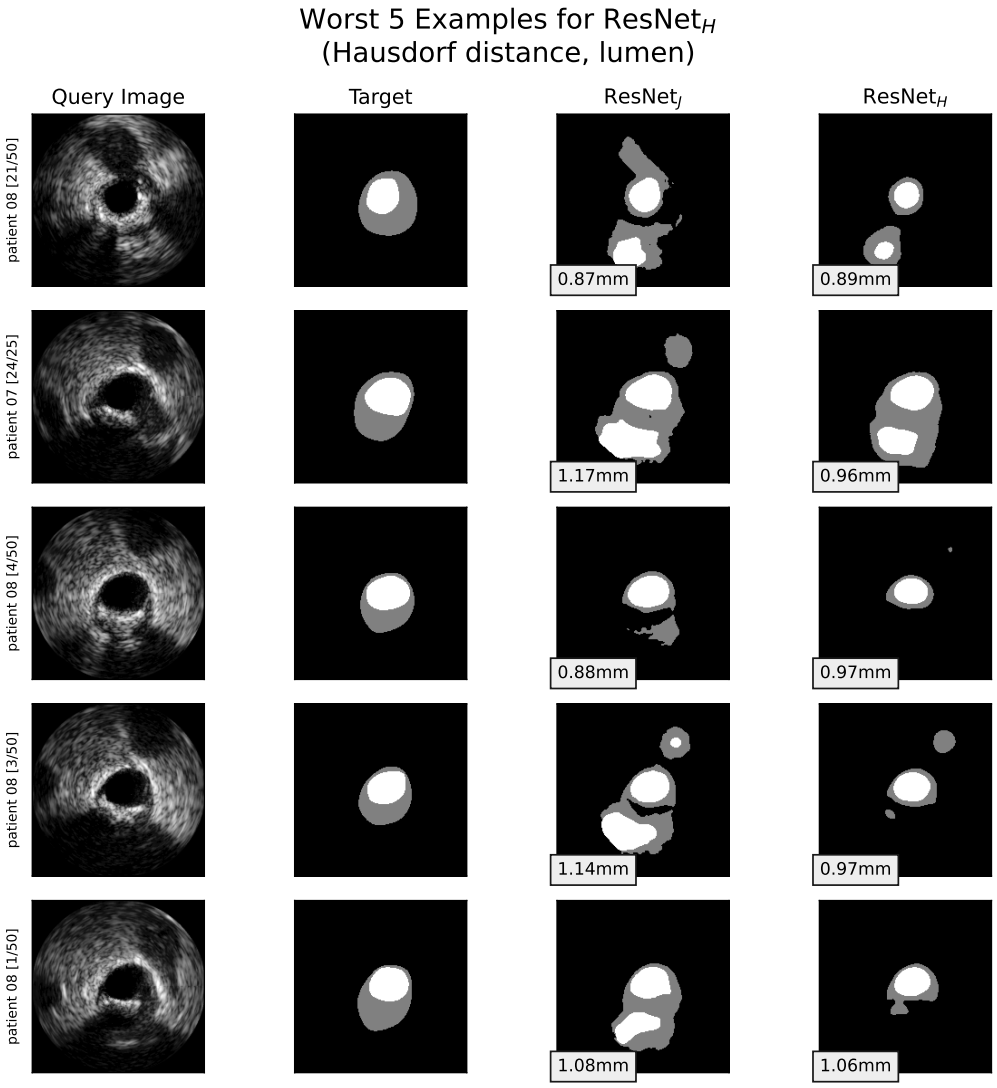

Figure 5.5: Comparison of the predictions from from ResNet$_J$ and ResNet$_H$ for the 5 samples where the ResNet$_H$ obtains the worst score.

5.3 Few-shot Scenario

In the evaluation of the few-shot scenario we compare: the INCEPTION RESNET model defined in §4.6 (ResNet$_S$) and PFENet. [29]. These models were trained on the Dataset A, and then evaluated on the Dataset B using the first frame of the dataset as a support image. The datasets are considered different enough, and therefore the evaluation is considered to be on a novel class. Note that for model I this means training for 20 epochs—empirically selected by monitoring the training loss.

Here the null hypothesis is that by using PFENet we don't obtain significant improvements for the performance on the novel dataset, using ResNet$_S$ as the benchmark.

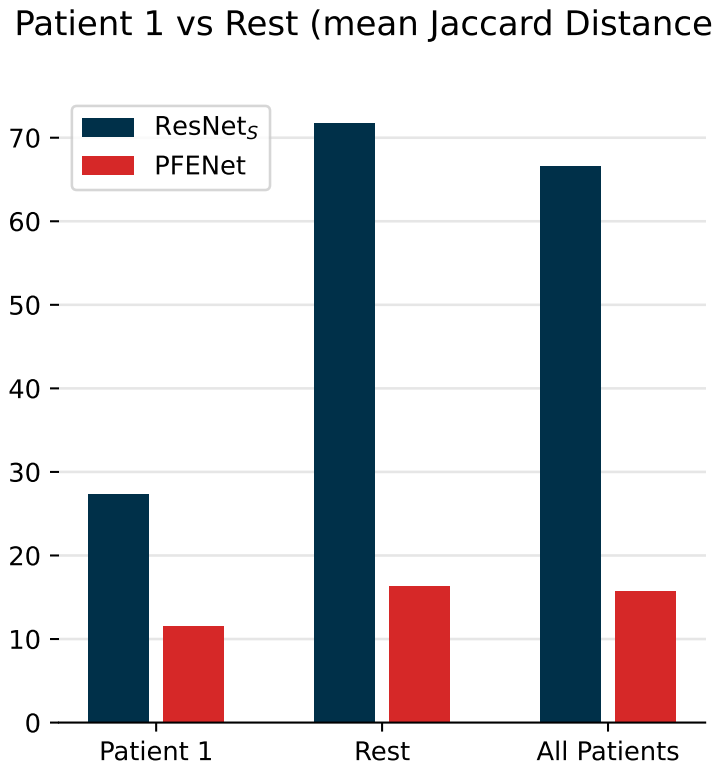

Figure 5.6: Resilience of ResNet$_S$ vs PFENet to diferent data

However, qualitatively we see that the predictions from the PFENet seem to be reasonable, whereas ResNet$_S$ often fails to detect lumen at all. Quantitatively, we have that ResNet$_S$ has a mean Jaccard distance of 0.67 ± 0.19 and Hausdorff distance of 0.25 ± 0.12

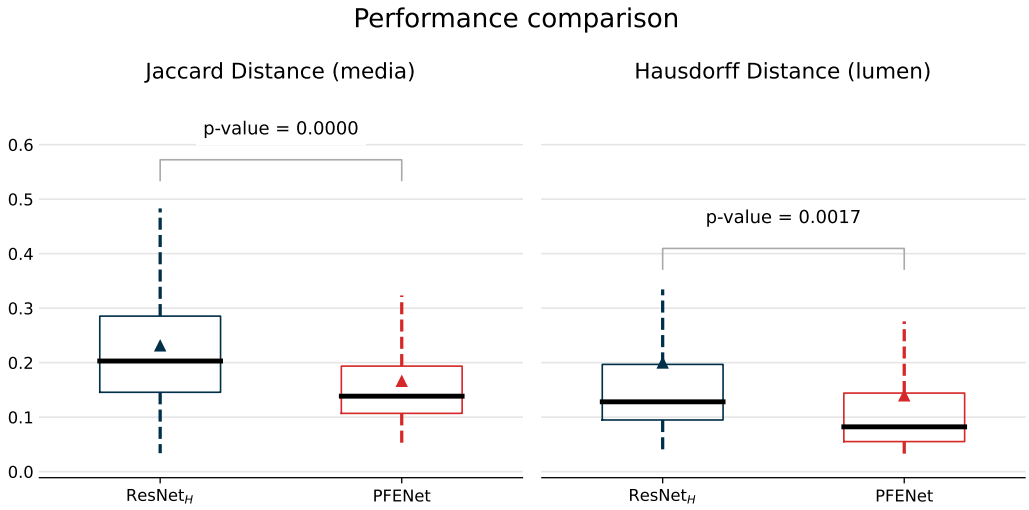

Figure 5.7: Comparison of ResNet$_H$ (trained on one fourth of the dataset), and PFENet (trained on one sample), whith the respective p-values of the significance tests.

mm.[1] PFENet, however has a mean Jaccard distance of 0.16 ± 0.05 and mean Hausdorff distance of 0.13 ± 0.05 mm. With this results we reject the null hypothesis (p-value < 0.001).

Additionally, we would expect that the performance would be higher for the patient 1 than the rest—since the support frame comes from patient 1. Interestingly, as we can see in 5.6, this does happen to ResNet$_S$, but not to PFENet. This confirms the idea that the PFENet [29] is resilient against overfitting and therefore suitable for the one-shot scenario.

For the final test we will compare the performance of the PFENet against the best model of §5.2. Surprisingly PFENet outperforms the Inception ResNet based architecture on the segmentation of the lumen. Indeed we observe a 28% decrease on the mean Jaccard distance, and a 30% decrease on the mean Hausdorff distance (see Fig. 5.7). [2] The improvement in performance is significant (p-value < 0.01).

[1] The images without a mask are discarded, otherwise the mean is not defined.

[2] These results might not agree with those on the table found in conclusions. This is because to be able to use the m-dependent t-test we must compare on one of the folds for the ResNet, whereas usually we will use all the folds since they are more representative of the distribution.

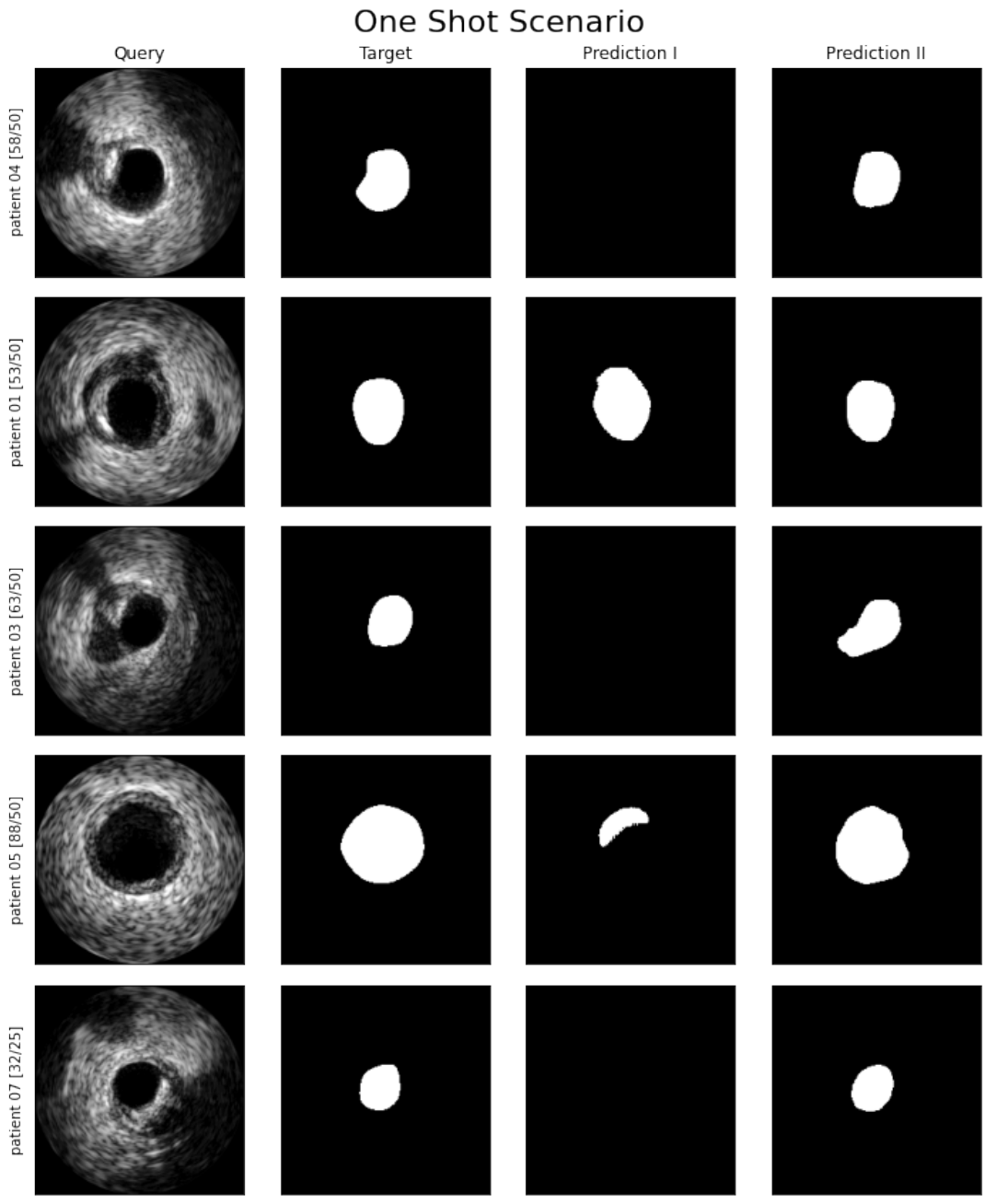

Figure 5.8: Qualitative overview of ResNet$_S$ vs PFENet on random images

Chapter 6

Conclusions

IVUS image segmentation is a challenging problem. Not only is image segmentation a particularly hard classification problem, but also the images tend to be quite difficult to interpret, even by medical professionals.

Furthermore, the lack of data makes training Deep Learning algorithms particularly challenging.

6.1 Contributions

In this project we extend the work done by [25], providing contributions in the preprocessing of data, model and loss selection, and the few-shot learning scenario.

For the preprocessing side of the methods, we introduce min-max normalization of the images and improvements in the implementations of data augmentation methods. Min-max normalization is relevant particularly in the few-shot scenario, where we use two distinct datasets simultaneously. For data augmentation we provide an alternative elastic deformation algorithm, and we apply it asymmetrically, i.e. only on the masks, to improve the results in [25].

For the model and loss selection, we provide a modular framework, implemented as an installable PYTHON package, for rapid development of models. We complement our empirical study with novel theoretical results (see lemmas 4.7 and 4.9). Finally we provide statistical justifications for the selection of our Hausdorff loss (see §5.2) instead of the previously used Jaccard loss, as well as our usage of data augmentation.

The last topic studied in this project is the few-shot learning scenario. Here we study a method to provide a trained model using only k samples; we select $k = 1$ training samples

(one-shot training). We use fine-tuning on a general purpose implementation (PFENET [29]), and provide a model that, not only dramatically outperforms other models in the one-shot training scenario , but also obtains excellent results compared to models trained with a larger split (see §5.3).

6.2 Summary of Results

In this section we provide a summary of the results analysed in depth in §5. In Table 6.1 we display a comparison of notable results, including models D and F from [25].

Comparison of results				
	lumen$_{JD}$	lumen$_{HD}$ (mm)	media$_{JD}$	media$_{HD}$ (mm)
Model D†	0.206 (0.135)	0.255 (0.297)	0.478 (0.170)	0.439 (0.332)
Model F†	0.236 (0.135)	0.235 (0.254)	0.455 (0.166)	0.318 (0.237)
ResNet$_H$	0.189 (0.102)	0.154 (0.154)	0.496 (0.165)	0.258 (0.175)
ResNet$_J$	0.198 (0.123)	0.209 (0.232)	0.415 (0.155)	0.282 (0.215)
PFENet	0.158 (0.089)	0.129 (0.139)		
ResNet$_S$	0.666 (0.376)	∞ (nan)		

Table 6.1: The results of Model D† and F†, originally by [25], are as evaluated using our split.

Model D and F are based on the same RESNET architecture used in this report, where model D uses affine transforms, and model F elastic deformations. Originally, [25] concludes that model D is better and that elastic transforms do not lead to better improvements. Using our improved implementation, we find out that that is not necessarily true.

Our proposed best models are ResNet$_H$ and PFENet (rows 3 and 6 of Table 6.1), where:

- ResNet$_H$ uses an architecture inspired by U-NET [24] and a pre-trained INCEPTION RESNET as a feature extractor, and is trained using a combination of affine and elastic deformations. It uses our Hausdorff loss implementation.
- PFENet is largely based on the implementation by [29], modified and fine-tuned using IVUS images (from Dataset A).

Additionally, for comparison we show models ResNet$_J$ and ResNet$_S$. ResNet$_J$ is identical to ResNet$_H$, except it uses the Jaccard loss instead of the Hausdorff loss; ResNet$_S$ is

ResNet$_H$ trained on the same example as PFENet.

6.2.1 Future Work

- Fine-tuning the PFENet on Dataset A made it usable for predicting on Dataset B. Maybe this could be extended to the other models: first learn to extract features from ultrasound images and then train on the target dataset.
- When training PFENet we realized that we didn't have many classes to use for episodic training. It could prove valuable to generate a dataset aggregating varied medical imaging segmentation datasets.
- It is evident on §5.2 that there are some particular "difficult frames" because of shadows and noise. I would be interesting to see that if training is done with a selection of difficult frames we would see some improvements.
- IVUS images present granular artefacts (speckle patterns) that are a consequence of ultrasound interferences. A speckle noise simulation could be valuable as a data augmentation method.

Appendix A

First

A.1 Numerical Results

Here we will display the tables with the numerical results of the models used throughout this report.

Data augmentation				
	lumen$_{JD}$	lumen$_{HD}$ (mm)	media$_{JD}$	media$_{HD}$ (mm)
U-Net$_{none}$	0.27 (0.15)	0.42 (0.45)	0.54 (0.15)	0.47 (0.25)
U-Net$_{affine}$	0.22 (0.14)	0.25 (0.31)	0.50 (0.16)	0.32 (0.21)
U-Net$_{elastic}$	0.21 (0.12)	0.20 (0.22)	0.50 (0.15)	0.29 (0.18)
ResNet$_{none}$	0.24 (0.15)	0.31 (0.36)	0.48 (0.15)	0.56 (0.36)
ResNet$_{elastic}$	0.24 (0.13)	0.24 (0.25)	0.45 (0.16)	0.33 (0.25)
ResNet$_{affine}$	0.23 (0.15)	0.28 (0.30)	0.49 (0.17)	0.39 (0.28)

Table A.1: Comparison of data augmentation results

Loss				
	lumen$_{JD}$	lumen$_{HD}$ (mm)	media$_{JD}$	media$_{HD}$ (mm)
ResNet$_{H}$	0.20 (0.11)	0.17 (0.17)	0.44 (0.16)	0.28 (0.21)
ResNet$_{J}$	0.20 (0.12)	0.19 (0.20)	0.43 (0.16)	0.31 (0.23)

Table A.2: Comparison of the choice of loss used in §5.2

	Few-shot comparison			
	lumen$_{JD}$	lumen$_{HD}$ (mm)	media$_{JD}$	media$_{HD}$ (mm)
PFENet	0.158 (0.089)	0.129 (0.139)		
ResNet$_S$	0.666 (0.376)	∞ (nan)		
ResNet$_H$	0.210 (0.118)	0.191 (0.193)	0.430 (0.165)	0.280 (0.195)

Table A.3: First two rows are the one-shot models, the last one serves as a reference.

A.2 Additional Resources

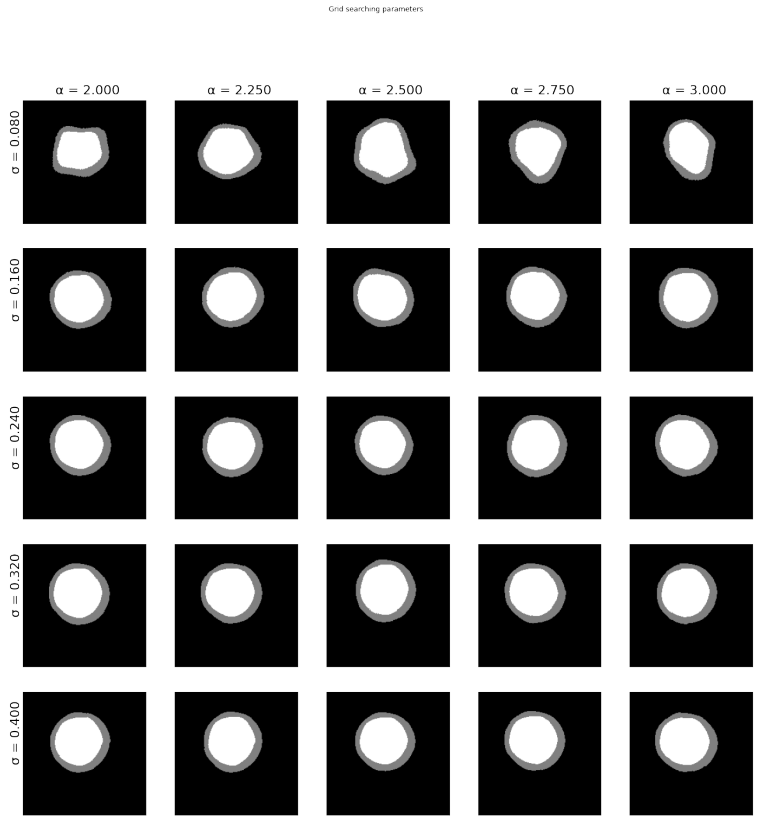

Figure A.1: Effects of different elastic transform parameters on the shape of the mask.

Bibliography

[1] Chirag Balakrishna, Sarshar Dadashzadeh, and Sara Soltaninejad. «Automatic detection of lumen and media in the IVUS images using U-Net with VGG16 Encoder». In: *arXiv preprint arXiv:1806.07554* (2018).

[2] Simone Balocco et al. «Standardized evaluation methodology and reference database for evaluating IVUS image segmentation». In: *Computerized Medical Imaging and Graphics* 38.2 (2014). Special Issue on Computing and Visualisation for Intravascular Imaging, pp. 70–90. ISSN: 0895-6111. DOI: https://doi.org/10.1016/j.compmedimag.2013.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0895611113001298.

[3] Qi Cai et al. «Memory Matching Networks for One-Shot Image Recognition». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4080–4088. DOI: 10.1109/CVPR.2018.00429.

[4] Wikimedia Commons. *Artificial neural network with layer coloring*. File: Colored neural network.svg. 2013. URL: https://en.wikipedia.org/wiki/File:Colored_neural_network.svg.

[5] François Chollet. In: (2022). URL: https://keras.io/examples/vision/oxford_pets_image_segmentation/.

[6] J. Deng et al. «Construction and Analysis of a Large Scale Image Ontology». In: Vision Sciences Society. 2009.

[7] *Estadística de defunciones según la causa de muerte. Últimos Datos*. Nov. 2021. URL: https://www.ine.es/uc/tlPx9LBx.

[8] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. 2017. DOI: 10.48550/ARXIV.1703.03400. URL: https://arxiv.org/abs/1703.03400.

[10] Hector M Garcia-Garcia et al. «IVUS-based imaging modalities for tissue characterization: similarities and differences». In: *The international journal of cardiovascular imaging* 27.2 (2011), pp. 215–224.

[11] Spyros Gidaris and Nikos Komodakis. «Dynamic Few-Shot Visual Learning Without Forgetting». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 4367–4375. DOI: 10.1109/CVPR.2018.00459.

[12] Benjamin Haeffele and René Vidal. «Global Optimality in Neural Network Training». In: July 2017, pp. 4390–4398. DOI: 10.1109/CVPR.2017.467.

[13] Abdelaziz Hammouche et al. «Automatic IVUS lumen segmentation using a 3D adaptive helix model». In: *Computers in Biology and Medicine* 107 (2019), pp. 58–72. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2019.01.023. URL: https://www.sciencedirect.com/science/article/pii/S0010482519300290.

[14] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[15] Davood Karimi and Septimiu E. Salcudean. «Reducing the Hausdorff Distance in Medical Image Segmentation With Convolutional Neural Networks». In: *IEEE Transactions on Medical Imaging* 39.2 (2020), pp. 499–513. DOI: 10.1109/TMI.2019.2930068.

[16] Sven Kosub. *A note on the triangle inequality for the Jaccard distance*. 2016. DOI: 10.48550/ARXIV.1612.02696. URL: https://arxiv.org/abs/1612.02696.

[17] Y. Lecun et al. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[18] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. DOI: 10.48550/ARXIV.1310.4546. URL: https://arxiv.org/abs/1310.4546.

[19] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[20] David Molony, Hossein Hosseini, and Habib Samady. «TCT-2 Deep IVUS: A machine learning framework for fully automatic IVUS segmentation». In: *Journal of the American College of Cardiology* 72.13S (2018), B1–B1.

[21] Alex Nichol, Joshua Achiam, and John Schulman. *On First-Order Meta-Learning Algorithms*. 2018. DOI: 10.48550/ARXIV.1803.02999. URL: https://arxiv.org/abs/1803.02999.

[22] Takeshi Nishi et al. «Deep learning-based intravascular ultrasound segmentation for the assessment of coronary artery disease». In: *International journal of cardiology* 333 (2021), pp. 55–59.

[23] Sidney I. Resnick. *A probability path*. Birkhäuser, 2014.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[25] Albert Sallés. *Advanced Semantic Segmentation using Deep Learning*. 2022.

[26] P.Y. Simard, D. Steinkraus, and J.C. Platt. «Best practices for convolutional neural networks applied to visual document analysis». In: *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* Aug. 2003, pp. 958–963. DOI: 10.1109/ICDAR.2003.1227801.

[27] Jake Snell, Kevin Swersky, and Richard Zemel. «Prototypical Networks for Few-shot Learning». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

[28] Flood Sung et al. «Learning to Compare: Relation Network for Few-Shot Learning». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 1199–1208. DOI: 10.1109/CVPR.2018.00131.

[29] Zhuotao Tian et al. «Prior Guided Feature Enrichment Network for Few-Shot Segmentation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.2 (Feb. 2022), pp. 1050–1065. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.3013717.

[30] Lucas Lo Vercio, Mariana Del Fresno, and Ignacio Larrabide. «Lumen-intima and media-adventitia segmentation in IVUS images using supervised classifications of arterial layers and morphological structures». In: *Computer Methods and Programs in Biomedicine* 177 (2019), pp. 113–121.

[31] Menghua Xia et al. «Extracting Membrane Borders in IVUS Images Using a Multi-Scale Feature Aggregated U-Net». In: *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. 2020, pp. 1650–1653. DOI: 10.1109/EMBC44109.2020.9175970.

[32] Ji Yang, Mehdi Faraji, and Anup Basu. «Robust segmentation of arterial walls in intravascular ultrasound images using Dual Path U-Net». In: *Ultrasonics* 96 (2019), pp. 24–33.

[33] Ji Yang et al. «IVUS-Net: an intravascular ultrasound segmentation network». In: *International Conference on Smart Multimedia*. Springer. 2018, pp. 367–377.

[34] Mehrdad Yazdani. *Example architecture of U-Net for producing k 256-by-256 image masks for a 256-by-256 RGB image*. 2019. URL: https://upload.wikimedia.org/wikipedia/commons/2/2b/Example_architecture_of_U-Net_for_producing_k_256-by-256_image_masks_for_a_256-by-256_RGB_image.png.

[35] Chi Zhang et al. *CANet: Class-Agnostic Segmentation Networks with Iterative Refinement and Attentive Few-Shot Learning*. 2019. DOI: 10.48550/ARXIV.1903.02351. URL: https://arxiv.org/abs/1903.02351.

[36] Chi Zhang et al. «DeepEMD: Few-Shot Image Classification With Differentiable Earth Mover,Äôs Distance and Structured Classifiers». In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020, pp. 12200–12210. DOI: 10.1109/CVPR42600.2020.01222.