University of Padua

Department of Information Engineering

Master's degree in Computer Engineering: Artificial Intelligence and Robotics

Weakly supervised segmentation of polyps on colonoscopy images

Supervisor
Prof. Loris Nanni

Graduating
Riccardo Rampon

Academic Year 2022/2023
Padua, October 24, 2023

A Mamma, Papà, Francesco e nonna Elena

Table of contents

Abstract

English version: Colorectal cancer (CRC) is one of the leading causes of death worldwide and continues to pose a critical public health challenge, demanding precise early detection and intervention. Colonoscopy, the diagnostic examination aimed at exploring the inner walls of the colon to discover any tumour masses, is an effective method to decrease mortality incidence. Emerging techniques, such as advanced image analysis driven by neural networks, hold promise for accurate diagnosis. However, studies have reported that, for various reasons, a certain percentage of polyps are not correctly detected during colonoscopy. One of the most important is the dependency on pixel-level annotations, which requires a lot of computational resources, making necessary innovative solutions. This thesis introduces strategies for improving polyp identification. For this purpose, the main techniques involve the so-called Explainable AI tools for analyzing saliency maps and activation maps, through several state-of-the-art visual saliency detection algorithms and Gradient-weighted Class Activation Mapping (Grad-CAM). In addition, a neural network for segmentation with DeepLabV3+ architecture is used, in which bounding boxes are provided on the training images, within a weakly supervised framework.

Italian version: Il cancro del colon-retto (CRC) è una delle principali cause di morte a livello mondiale e continua a rappresentare una sfida critica per la salute pubblica, richiedendo una precisa e tempestiva diagnosi e un intervento mirato. La colonscopia, ovvero l'esame diagnostico volto a esplorare le pareti interne del colon per scoprire eventuali masse tumorali, ha dimostrato essere un metodo efficace per ridurre l'incidenza di mortalità. Le tecniche emergenti, come l'analisi avanzata delle immagini tramite reti neurali, sono promettenti per una diagnosi accurata. Tuttavia, alcuni studi hanno riportato che, per varie ragioni, una certa percentuale di polipi non viene rilevata correttamente durante la colonscopia. Una delle più importanti è la dipendenza dalle annotazioni a livello di pixel, che richiede molte risorse computazionali; per questo si rendono necessarie soluzioni innovative. Questa tesi introduce alcune strategie per migliorare l'identificazione

dei polipi. A tal fine, le tecniche principali utilizzate coinvolgono i cosiddetti metodi di Explainable AI per l'analisi delle mappe di salienza e di attivazione, attraverso diversi algoritmi di rilevamento della salienza visiva e la Gradient-weighted Class Activation Mapping (Grad-CAM). Inoltre, viene utilizzata una rete neurale per la segmentazione con architettura DeepLabV3+, in cui vengono fornite le bounding box sulle immagini di addestramento, in un contesto debolmente supervisionato.

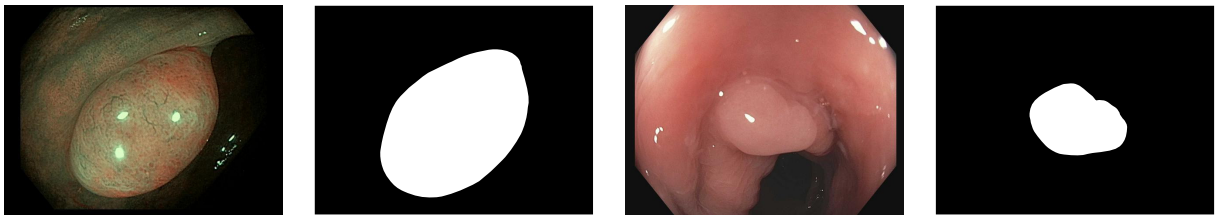
Chapter 1

Introduction

1.1 Contextualization of the Topic

Colon polyps are initial indicators of colorectal cancer, which ranks among the most prevalent forms of cancer [1]. In Italy, in 2021, colorectal cancers were the third neoplasm in men and the second in women [2]. Detecting these precancerous growths during screening holds critical importance, as their early identification and accurate diagnosis are important factors for effective treatment and lowering mortality rates [3]. Research indicates that a simple 1% increase in polyp detection could potentially lead to a 3% reduction in the incidence of colon cancer [4]. Currently, colonoscopy is the established standard diagnostic examination in clinical practice for detecting abnormal tissue within the gastrointestinal tract. However, the precision of this procedure is contingent upon the physician's skill and demands substantial effort. Thus, the clinical prevention of colorectal cancer underscores the need for automated methods capable of pinpointing all existing polyps with a high degree of accuracy. The field of artificial intelligence and machine learning has seen widespread application in semantically segmenting polyps within medical imagery.

Figure 1.1.1 presents two instances of colonoscopy images depicting polyps and their corresponding segmentations. Traditional segmentation methods have historically employed techniques like geometric analysis and frame-based models [5], or a hybrid approach integrating contextual and shape-based strategies [6]. Nevertheless, these methodologies often struggle to capture holistic global context information and are less robust when compared with intricate scenarios, mainly due to their reliance on manually designed features [7].



(a) (b) (c) (d)

Figure 1.1.1: Two examples of the content of the Polyp-Box-Seg [8] dataset for the semantic segmentation of polyps: (a)(c) original images; (b)(d) ground truth.

1.2 Aim and Objectives of the Thesis

The primary purpose of this thesis is to enhance and refine the methodology employed for the segmentation of polyps in colonoscopy images, focusing on a specific weakly supervised approach, typical in the medical field. The fundamental goals of this thesis are the following:

- to examine the foundational principles concerning the segmentation of medical images and the key role assumed by neural networks within this context;
- to analyze the advantages of weakly supervised learning techniques and the help given by methods of Explainable AI such as Saliency Map and Grad-CAM, in terms of improving the accuracy related to polyp segmentation;
- to construct a valid strategy to adapt these specific techniques to the domain of colonoscopy images;
- to perform a set of experiments to evaluate the effectiveness of the created methodologies, with a focus on quantitative and qualitative metrics;
- to perform a comparative analysis that contrasts the results of the proposed new improvement methodologies with the initial approach;
- to understand if there can be further work starting from the methods developed.

1.3 Overview of the Index

This particular section seeks to provide a general preview of the structural composition and content allocation of this thesis. The thesis is structured into several distinct sections and sub-sections, each with a specific purpose that contributes to the overall aims and objectives of the thesis.

The initial sections are dedicated to the creation of a solid basic structure, explaining the fundamental concepts related to the segmentation of medical images, the integration of neural networks in this specific field, and the new methodologies used. This is fundamental for an in-depth exploration of the distinct methodologies of weakly supervised learning, in particular the applications of Saliency Maps and Grad-CAMs, which are seen as the main techniques for improved polyps segmentation.

The core of this thesis focuses on the development and adaptation of these methodologies in the particular context of colonoscopy imaging. This involves the use of iterative learning, the generation of segmentation masks, and their enhancement through the integration of the methodologies mentioned above. The theoretical and technical foundations of each methodology are precisely outlined.

Subsequent sections outline the steps of implementation and experimentation. This includes the explanation of the experimental environment, the implementation of the devised methodologies, and the testing protocols used.

The last sections deal with the experimental results, followed by a comparative analysis that seeks to emphasize how the approaches developed perform concerning the basic approach. The closure of this thesis is punctuated by an enumeration of useful references for further clarification.

Chapter 2

Theoretical Foundations

2.1 Concepts of Medical Segmentation

Medical image segmentation constitutes a crucial element in medical image processing. This process involves the subdivision of an image into anatomical regions of interest, allowing healthcare providers to obtain detailed and accurate information to diagnose medical conditions, plan treatments, and monitor disease progression.

The main goal of segmentation is to precisely delineate anatomical structures within images, thus enabling the identification and localization of tumours, organs, tissues, and other relevant features. This level of detail is particularly essential when it comes to individualized diagnosis and treatment, as it helps to accurately identify anatomical variations among patients and provide a clear view of the sizes and locations of structures within the body.

Traditionally, medical anatomical segmentation was performed manually, requiring the surgeon to carefully trace the contours, slice by slice, through an entire stack of MRI (Magnetic Resonance Imaging) or CT (Computed Tomography) images. Thus, there is a strong need to develop a solution that can automate this difficult process [9].

Medical image segmentation has been revolutionized by the adoption of neural networks, and advanced machine learning algorithms that can recognize complex patterns in images, paving the way for more accurate and efficient segmentation solutions.

2.2 Artificial Neural Networks for Segmentation

2.2.1 Image Segmentation task

Image segmentation plays a critical role in the field of computer vision, aiming to partition an image into meaningful and distinguishable regions or objects. This task holds fundamental importance across various applications, including object recognition, tracking, detection, medical imaging, and robotics. Numerous techniques exist, spanning from traditional methodologies to those driven by deep learning approaches [10]. There exist eight distinct types of segmentation modes applicable in the context of medical imaging [11]:

- Instance segmentation,
- Semantic segmentation,
- Panoptic segmentation,
- Thresholding,
- Region-based segmentation,
- Edge-based segmentation,
- Clustering segmentation,
- Foundation Model segmentation.

We dealt with semantic segmentation. Semantic segmentation involves labelling each pixel within an image. This process generates a densely labelled image, which can then be processed by an artificial intelligence application to create a segmentation mask. In this mask, pixel values $[0, 1, ...255]$ are converted into class labels $[0, 1, ...n]$, clearly delineating different parts of the image.

A myriad of approaches have been developed to address the challenges of semantic segmentation. Notably, Convolutional Neural Networks (CNNs) have risen to prominence due to their capacity to capture complex spatial features [9]. In the realm of medical image segmentation, semantic segmentation techniques contribute to accurate organ delineation, lesion detection, and treatment planning. Subsequent Sections will delve into diverse neural network architectures, including CNNs, and their applications in medical segmentation tasks.

2.2.2 Multi-Layer Perceptron (MLP)

MLPs (Multi-Layer Perceptrons) are artificial neural networks that define a loop-free oriented graph. This kind of neural network is characterized by the presence of multiple layers composed of computational units called artificial neurons. These neurons are arranged in sequence, connected through weights that regulate the influence of transmitted data.

These networks are part of the feed-forward neural networks family: the data flow proceeds from input nodes to output nodes without creating cycles. The initial input passes through neurons in the first layer, known as the input layer. Next, computations proceed through intermediate layers, called hidden layers, and end up in the last layer, known as the output layer. In each neuron, inputs are weighted and summed, and an activation function is applied, according to the formula:

$$y = f(b + \sum_{i=0}^{n} w_i \cdot x_i) \tag{2.2.1}$$

where b is the bias; w_i is the $i{-}th$ weight parameter; x_i is the $i{-}th$ input; f is the activation function and it must respect certain properties, including non-linearity, continuity, and differentiability:

- Non-linearity: the activation function must be non-linear because if not, the network would only be able to represent linear functions (since any combination of linear functions will still be a linear function). Furthermore, if the activation function were linear, several levels of perceptrons could be condensed into one (again by the same property of combination of linear functions),
- Continuity and Differentiability: the activation functions must also exhibit qualities of continuity and differentiability. This is important during the backpropagation process with optimizers using gradient descent (i.e. Stochastic Gradient Descent with Momentum, Adam), where the gradient of the objective function is calculated; here, if the activation function is not linear and differentiable, this operation is not possible.

Figure 2.2.1 visually explains this element of a basic neural network.

This configuration of weighted connections and activation functions gives the MLP the ability to learn complex relationships between inputs and outputs. By training the MLP on a dataset, the weights of connections between neurons can be refined, enabling the network to generate accurate predictions or classifications on new inputs. The MLP is

widely used in multiple domains, including pattern recognition, natural language analysis, computer vision, and others.

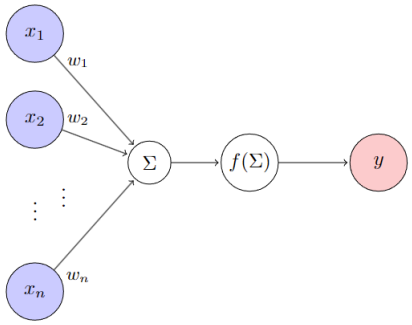


Figure 2.2.1: Representation of an artificial neuron.

2.2.3 Convolutional Neural Network (CNN)

Convolutional neural networks (CNNs) represent a class of deep learning models widely used in the field of image recognition and computer vision. This type of network is inspired by the structure of the human visual system and has proven particularly effective in automatically extracting features from complex images. Convolutional neural networks (CNNs) differ in adopting shared weights and local connections between layers; this means that connection weights are shared among groups of neurons and that each neuron in a subsequent layer is influenced only by a limited subset of neurons in the previous layer. This pattern of local connections allows a significant reduction in the total number of connections in the network, thus reducing training time and facilitating the creation of deeper networks. Shared weights, replicated in different parts of the image, are called filters. Typically, many filters are created for each layer, and each filter is applied to the entire image, allowing specific features to be extracted from it.

CNNs are composed of several layers, each of which has a specific role in processing visual information. Convolutional layers perform the convolution operation, in which filters are applied to detect patterns and features in images:

$$S(x,y) = (I * K)(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} I(x+i, y+j) \cdot K(i,j) \tag{2.2.2}$$

where:

- $(I*K)(x,y)$ represents the value of the resulting pixel at the position of the convolved image,

- I is the image input (or feature map) in the convolutional layer,
- K is the kernel or convolutional filter,
- m and n are the dimensions of the kernel,
- x and y are the coordinates of the pixel in the convolved image.

To perform a convolution it is often necessary to use other steps and other properties typical of the convolution operation such as Stride and Padding to optimize the output:

- Stride is a property of the convolution operation in which the filter is made to run over the input volume not with unit steps (default) but with a larger step (called Stride). Stride reduces the size of the feature maps in the output volume and consequently, the number of connections; small Strides (e.g., 2 or 4) can increase efficiency at the cost of a slight penalty in accuracy.
- Padding is a property of the convolution operation that allows the user to adjust the size of feature maps by adding an edge to the input volume (null values). The Padding parameter denotes the thickness of the edge. Padding is useful for filtering the side pixels of the image; without Padding all edge pixels are analyzed by a very small number of filters, since these filters cannot leave the input matrix, thus leading to a reduction in output size and a loss of information.

Next, the pooling layers reduce the spatial dimension of the extracted features, contributing to the reduction of computational complexity. The extracted features are then linked to the fully connected layers, where classification and label assignment to the images take place.

A key aspect of CNNs is their learning using the backpropagation method, which allows filter weights to be automatically adapted during the training phase. This characteristic allows CNNs to autonomously learn increasingly complex and specific patterns during the training process on a large dataset. CNNs have demonstrated outstanding results in various tasks, such as object recognition, face detection, medical image analysis, and many others, solidifying them as key tool in the development of advanced computer vision applications.

2.3 Training Modalities

2.3.1 Training

In machine learning, ”training” represents the process by which a model is instructed to recognize patterns and relationships in the data. During training, the model is exposed to input data along with the corresponding output values. The model adjusts its internal parameters iteratively to reduce the discrepancy between its predictions and actual output values. This process involves optimization algorithms and loss functions that guide the model toward more accurate predictions.

2.3.2 Supervised and Unsupervised Training

In supervised learning, the training data consists of input and output pairs. The model learns to map inputs into outputs by observing the relationships between them in the training data. The goal is to have a model that can generalize this mapping to make accurate predictions about new, unseen data.

In the case of polyps image segmentation, a set of images (input) is used as a training set to train the model, which also needs an accurate segmentation mask provided by a physician (output). This last factor makes training particularly time-consuming since the creation of accurate segmentation masks takes a considerable amount of time.

Unsupervised learning involves training a model on input data without explicit output labels. The goal is to discover patterns, structures, or relationships within the data. Common tasks include clustering, in which the model groups similar data together, and dimensionality reduction, in which the model reduces data complexity while preserving important information.

2.3.3 Weakly Supervised Training

Weakly supervised learning is an approach that lies between fully supervised and unsupervised learning. This approach is used in scenarios where obtaining accurate labels for training data is difficult or expensive. Instead of using fully labelled data, weakly supervised learning exploits partially labelled or noise-affected data.

Often, this approach employs heuristics, constraints, or probabilistic models to infer approximate labels from the available weak annotations. For example, in the context of polyp detection, if only whole-image-level labels are available (such as ”positive” or ”negative”), with no further details about each polyp, weak supervision techniques could

be used to identify regions within images that are likely to correspond to the labelled classes.

Using this approach, we can teach the model to recognize areas within the images that might represent polyps, even though we do not have detailed information about each individual. In practice, the model could learn to identify common visual features associated with polyps, such as shape or hue, and use them to highlight possible areas of interest. This type of approach allows the model to help identify polyps, even in the absence of precise annotations for each formation.

Learning with weak supervision may be useful in scenarios where labelling large amounts of data is impractical, but a form of supervision is still available, like medical image segmentation.

2.4 Explainable AI (XAI) tools

Explainable AI (XAI) is a set of tools and techniques that allows human users to better understand why an artificial intelligence model generates certain decisions by describing how it works. This kind of approach is extremely useful in the context of neural networks or machine learning models because they are black-box models that have no interpretability and do not reveal any information about their internal workings; this explainability can help developers to ensure that the system is working as expected or, in general, to verifying the system's functionality. In the following Section, two widely used techniques in the XAI's context, namely Saliency Maps and Class Activation Maps (CAMs) will be introduced.

2.4.1 The concept of Interpretability

First of all, it is important to define what Interpretability is and its importance in the context of machine learning models.

Defining mathematically what interpretability is, becomes complex. One interesting explanation of interpretability was proposed in [12], and points out that it concerns the degree to which a person can understand the cause and the reasons behind a decision.

Interpretability and explainability are closely related. Interpretability is used more often in the context of machine learning, instead, Explainability is used more in the context of deep neural networks and deep learning.

Increasing the level of interpretability of a machine learning model makes it easier to

understand the reasons that drive its decisions or predictions. A machine learning model is considered more interpretable than another if the reasons behind its decisions are more easily understood than the other model's decisions.

Now let's try to understand more about why interpretability is important. In predictive modeling, you have to make a choice: do you want to know what is predicted, or to understand why the prediction was made? In some cases, you don't care about the reason behind a decision; it is enough to know that predictive performances on a test dataset are good. But sometimes, in other situations, knowing the 'why' can help you learn more about the problem, the data, and the reason why a model might fail. The need for interpretability arises from a lack of formalization of the problem [13], which means that for some problems it is not enough to get the prediction ("what"). The model must also explain how it arrived at that prediction ("why") because a correct prediction only partially solves the original problem [14]. Practitioners seek interpretability for three main reasons [15]:

- Debugging: understanding where or why predictions go wrong and running "what-if" scenarios can improve model robustness and eliminate bias;
- Guidelines: black-box models may violate corporate technology best practices and personal preference;
- Regulations: some government regulations require interpretability for sensitive applications such as in finance, public health, and transportation.

Model interpretability addresses these concerns and increases trust in the models in situations where explanations for predictions are important or required by regulation. Interpretability is typically applied at two levels [15]:

- Global Methods: these interpretability methods provide an overview of the most influential variables in the model based on input data and predicted output. In contrast to local methods, global methods can explain how a model works even without probing its functionality on a specific set of inputs;
- Local Methods: these interpretability methods explain a single prediction result.

The methods that will be presented in the following Sections are all local.

2.4.2 Theory of Saliency Detection and Saliency Maps

Saliency maps are visual representations or images that highlight the most important or relevant regions within an image: the brightness of a pixel represents how salient that

pixel is (the brightness of a pixel is directly proportional to its saliency). These regions are typically those that most attract the attention of humans or are crucial to a particular task, such as object recognition or segmentation [16]. It is generally a grayscale image. An example of a saliency map is shown in Figure 2.4.1.

The concept of "saliency" in images refers to distinctive features, such as pixels of special interest or regions of high resolution, that capture attention in visual processing. These unique features help identify visually appealing areas within an image.

Saliency maps were initially proposed by neuroscientists Laurent Itti, Christof Koch, and Ernst Niebur in their study of a visual attention system, inspired by the behaviour and the neuronal architecture of the early primate visual system. These visual attention models are a part of the methodologies used in this work and mainly concern the use of the Spectral Saliency Map [17] and the Itti-Koch Saliency Map [18].

The computation of a Saliency Map first goes through the application of Saliency Detection algorithms, which aim at finding salient objects in an image.

The approaches used in this thesis are based entirely on the research work of Boris Schauerte [19], who has made available a Matlab toolbox that allows direct calculation of a spectral saliency map by implementing several state-of-art saliency detection algorithms. His work is based on other research studies including [17] [20] [21] [22] [23].

Figure 2.4.1: Example of Saliency Map [24].

Spectral Saliency Map

A Spectral Saliency Map is a type of saliency map that highlights regions in an image according to their frequency content. It uses the concept that regions with higher frequency content tend to attract more attention from human observers. For example, edges, textures, colours, and fine details in an image often have higher frequency components.

By analyzing the spectral characteristics of an image, a Spectral Saliency Map can be generated to emphasize these areas.

In this context is important to explain the work performed by Xiaodi Hou and Liqing Zhang [17], whose aim was to present an innovative method for saliency detection based on a technique called "Spectral Residual". The method assumes that salient regions in images have higher residual frequency spectra than their surroundings. The authors propose to calculate the Fourier transform of the image and to compute the difference between the frequency spectrum and a filtered version of it. This spectral residual is then used to generate a saliency map.

More precisely, let's suppose to have an image $\mathcal{I}(x)$. The Spectral Saliency Map is obtained in a few steps:

$$\begin{aligned} \mathcal{A}(f) &= \Re\left(\mathcal{F}\left[\mathcal{I}(x)\right]\right) \\ \mathcal{P}(f) &= \Im\left(\mathcal{F}\left[\mathcal{I}(x)\right]\right) \\ \mathcal{L}(f) &= \log\left(\mathcal{A}(f)\right) \\ \mathcal{R}(f) &= \mathcal{L}(f) - h_n(f) \cdot \mathcal{L}(f) \\ \mathcal{S}(x) &= g(x) \cdot \mathcal{F}^{-1}\left[\exp(\mathcal{R}(f) + \mathcal{P}(f))\right]^2 \end{aligned} \tag{2.4.1}$$

where:

- $\mathcal{A}(f)$ is the amplitude of the Fourier transform $\mathcal{F}[\mathcal{I}(x)]$ of the image,
- $\mathcal{P}(f)$ is the phase of the Fourier transform $\mathcal{F}[\mathcal{I}(x)]$ of the image,
- $\mathcal{L}(f)$ is the log spectrum representation of the image,
- $\mathcal{R}(f)$ is the spectral residual ($h_n(f)$ is a local filter of size n), and
- $\mathcal{F}$, $\mathcal{F}^{-1}$ denote the Fourier Transform and Inverse Fourier Transform, respectively.

As we can see, the calculated spectral residual is used to generate the Spectral Saliency Map. In this map, the highest values correspond to the regions of the image that have significant discrepancies from the surrounding context in terms of frequency spectrum. Figure 2.4.2 shows an example of a Spectral Saliency Map with Discrete Cosine Transform (DCT) algorithms.

Itti-Koch Saliency Map

The Itti-Koch Saliency Map is a method for generating saliency maps. It is based on a biological architecture, shown in Figure 2.4.3, proposed by Koch and Ullman [18]. This

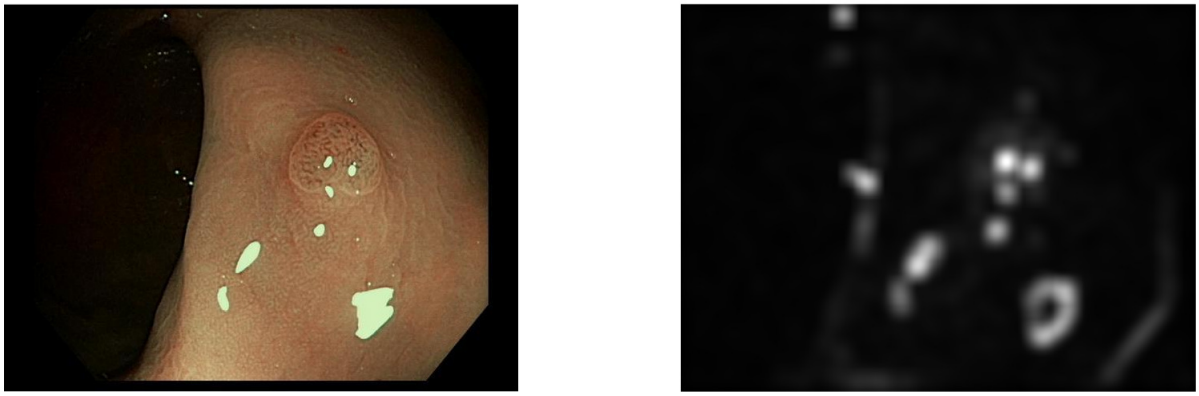


Figure 2.4.2: Example of Spectral Saliency Map with DCT algorithms, computed into an image of the dataset Polyp-Box-Seg [8].

approach takes into account various factors such as intensity, colour, and orientation of pixels to determine which areas in an image are most salient. The Itti-Koch method attempts to simulate the early stages of visual attention in the human brain.

It is based on the so-called "feature integration theory", explaining human visual search strategies [25]. As we can see from Figure 2.4.3, the input image is first decomposed into a set of topographic feature maps, specifically colours, intensity, and orientations maps, that will be described in detail later. After that, nine spatial scales are created using dyadic Gaussian pyramids [26] which progressively apply a low-pass filter and sub-sample the input image, obtaining image-reduction factors ranging from zero to eight octaves.

Next, Center-surround differences (defined as $\ominus$) between a "center" fine scale c and a "surround" coarser scale s is computed: the center is a pixel at scale $c \in 2, 3, 4$, and the surround is the corresponding pixel at scale $s = c + \delta$, with $\delta \in 3, 4$. The across-scale difference between two maps is obtained by interpolation to the finer scale and point-by-point subtraction. Instead, the across-scale addition between two maps consists of the reduction of each map to a specific scale and point-by-point addition. Using multiple scales not only for c but also for $\delta = s - c$, enables a sort of multi-scale feature extraction [18].

Given the red (r), green (g), and blue (b) channels of the input image, an intensity image I is obtained as $I = (r + g + b)/3$ (I is used to create a Gaussian pyramid $I(s)$, where $s \in [0..8]$ is the scale). The first set of feature maps regards the intensity contrast. Here, they generate a set of six maps (c, s), with $c \in 2, 3, 4$ and $s = c + \delta$, with $\delta \in 3, 4$:

$$(c, s) = |I(c) \ominus I(s)| \tag{2.4.2}$$

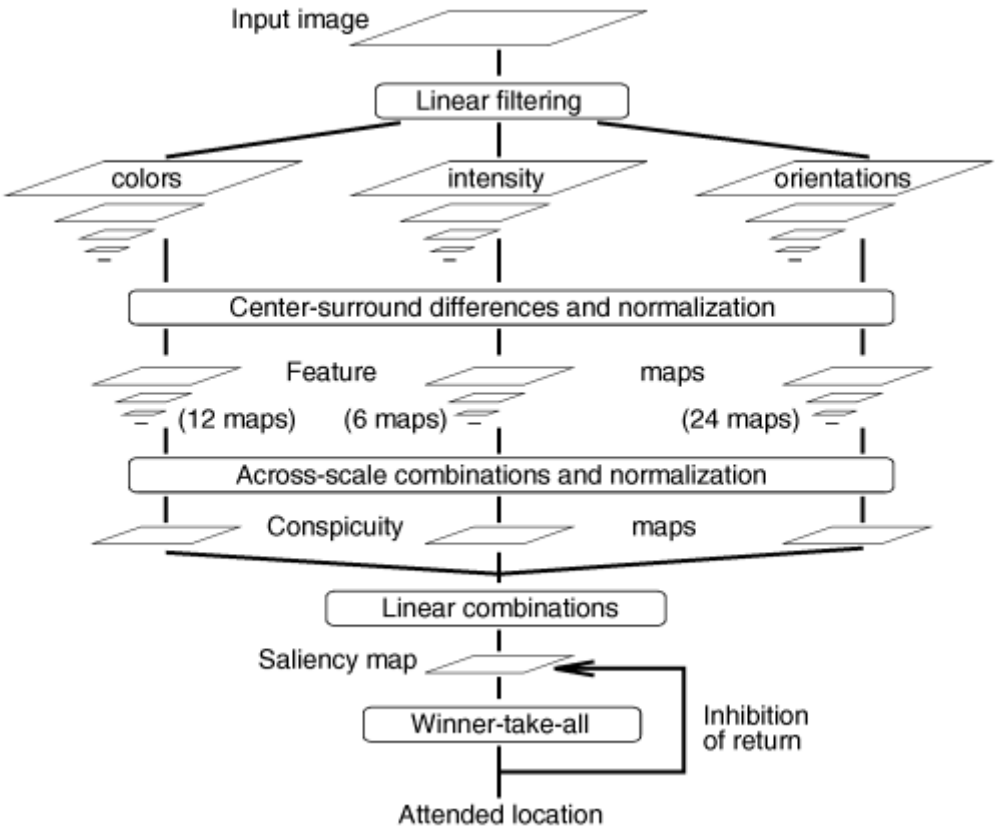


Figure 2.4.3: General architecture of the Itti-Koch model. The Figure is taken from [18].

After that, another four colour channels are created:

$$\begin{aligned} R &= r - (g+b)/2 \quad \text{for red,} \\ G &= g - (r+b)/2 \quad \text{for green,} \\ B &= b - (r+g)/2 \quad \text{for blue, and} \\ Y &= (r+g)/2 - |r-g|/2 - b \quad \text{for yellow} \end{aligned} \tag{2.4.3}$$

A second set of feature maps is similarly constructed for the colour channels:

$$\mathcal{RY}(c,s) = |(R(c) - G(c)) \ominus (G(s) - R(s))| \tag{2.4.4}$$

$$\mathcal{BY}(c,s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))| \tag{2.4.5}$$

where $\mathcal{RY}(c,s)$ are designed to account for both red/green and green/red double opponency at the same time, instead $\mathcal{BY}(c,s)$ for blue/yellow and yellow/blue double opponency.

Finally, the third set of feature maps regards the orientations: local orientation information is obtained from I using oriented Gabor pyramids $O(\sigma, \theta)$, where $\sigma \in [0..8]$ represents the scale and $\theta \in [0°, 45°, 90°, 135°]$ is the orientation. Orientation feature maps, $O(c, s, \theta)$, encode local orientation contrast between the centre and surround scales:

$$O(c, s, \theta) = |O(c, \theta) \ominus O(c, s, \theta)|. \tag{2.4.6}$$

In total, 42 feature maps are created: six for intensity, 12 for colour, and 24 for orientation [18].

Next, there is a normalization process that involves a normalization operator $\mathcal{N}(.)$, which favours maps in which is present a small number of strong peaks (salience locations), while suppressing maps that contain a high number of equal peaks. Specifically, the normalization operator $\mathcal{N}(.)$ consist of:

- normalizing the values in the map to a fixed range $[0..M]$, to eliminate modality-dependent amplitude differences,
- finding the location of the global maximum M in the map and computing the average m of all other local maxima,
- globally, multiplying the map by $(M - \overline{m})^2$.

In this way, only local maxima of activity (meaningful/salience activations regions in the map) are considered.

After that, feature maps are combined into three "conspicuity"/"saliency" maps, $\overline{\mathcal{I}}$ for intensity, $\overline{\mathcal{C}}$ for colour, and $\overline{\mathcal{O}}$ for orientation, at the scale ($\sigma = 4$) of the saliency map. These conspicuity maps are obtained through across-scale addition (defined as $\oplus$):

$$\begin{aligned}
\overline{\mathcal{I}} &= \oplus_{c=2}^{4} \oplus_{s=c+3}^{c=4} \mathcal{N}(\mathcal{I}(c, s)) \\
\overline{\mathcal{C}} &= \oplus_{c=2}^{4} \oplus_{s=c+3}^{c+4} [\mathcal{N}(\mathcal{RG}(c, s)) + \mathcal{N}(\mathcal{BY}(c, s))] \\
\overline{\mathcal{O}} &= \sum_{\theta \in [0°, 45°, 90°, 135°]} \mathcal{N}(\oplus_{c=2}^{4} \oplus_{s=c+3}^{c+4} \mathcal{N}(\mathcal{O}(c, s, \theta)))
\end{aligned} \tag{2.4.7}$$

Finally, these three saliency maps are normalized and summed into the final output

saliency map:

$$S = \frac{1}{3}(\mathcal{N}(\overline{I}) + \mathcal{N}(\overline{C}) + \mathcal{N}(\overline{O})) \tag{2.4.8}$$

Figure 2.4.5 shows the difference between the spectral saliency map and the Itti-Koch approach.

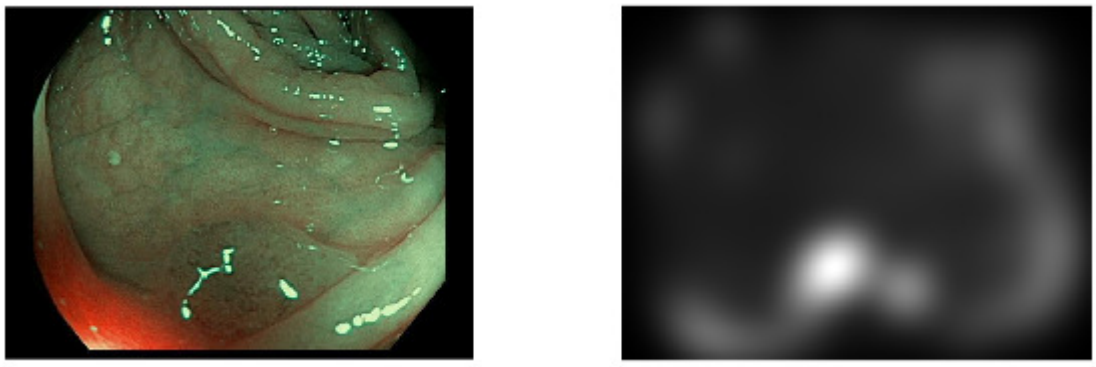


Figure 2.4.4: Example of saliency map with Itti-Koch approach, computed into an image of the dataset Polyp-Box-Seg [8].

2.4.3 Theory of Class Activation Maps (CAMs)

Class Activation Map (CAM) is an Explainable AI approach used for CNNs. It is an interpretation method for neural networks that can visualize features and concepts learned, and explain the individual predictions [14].

It was introduced by Zhou et al. [27] and their work involved the analysis and evaluations based on neural networks structured in a similar way to the 'Network in Network' (NIN) architecture [28]. In these networks, the usual stack of fully connected layers at the end of the model was replaced by a distinct layer called Global Average Pooling (GAP). This Global Average Pooling (GAP) layer essentially averages the activations of each feature map and aggregates these averages into a vector, which is then output. A weighted sum of this vector is then fed to the final softmax layer.

This architecture allows highlighting the important regions of the image by projecting back the weights of the output on the convolutional feature maps. Figure 2.4.6 shows this process.

Instead, Gradient-weighted Class Activation Map (Grad-CAM) is a more versatile explainable technique than CAM, that can be used to help understand the predictions made by a deep neural network [29] and therefore to produce visual explanations for any

Figure 2.4.5: Comparison between Spectral and Itti's approach. In each group we have 1) the input image, 2) the saliency map generated by spectral residual, 3) the saliency map generated by Itti's approach, and 4) the labelled map (we are not interested in this map at the moment). The Figure is taken from [17].

arbitrary CNN.

Grad-CAM determines the importance of each neuron in a network prediction by considering the gradients of the target flowing through the deep network. Grad-CAM computes the gradient of a differentiable output, for example, class score, concerning the convolutional features in the chosen layer. The gradients are pooled over space and time dimensions to find the neuron importance weights. These weights are then averaged across each activation map to determine which features are most important in the prediction and to give us an importance score [30].

Suppose to have an image classification network with y^c as output, representing the score for class c, and want to compute the Grad-CAM map for a convolutional layer $A^k_{i,j}$ with k feature maps (channels), where i,j represent pixels coordinates. The neuron

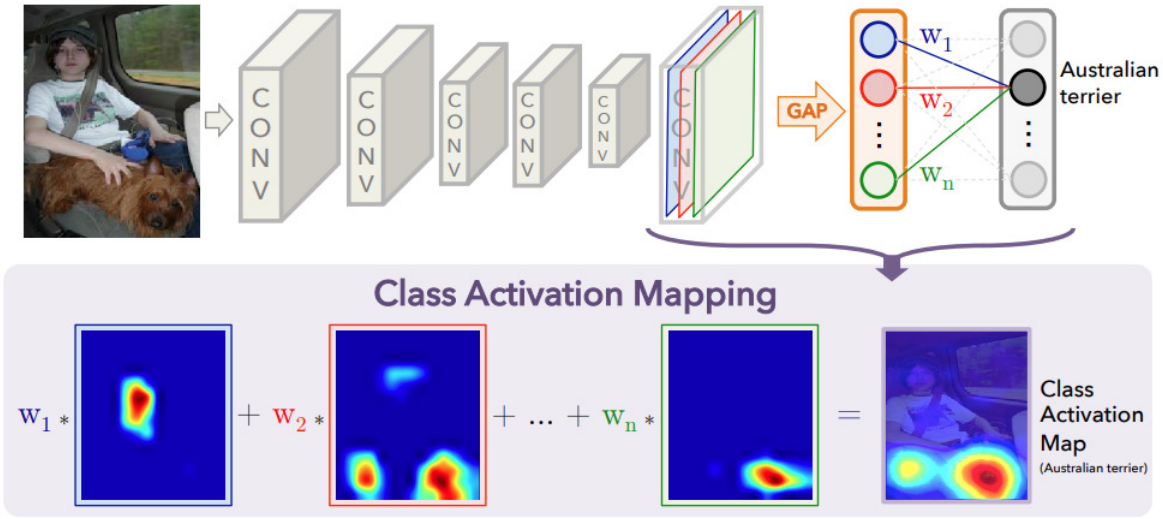


Figure 2.4.6: Class Activation Mapping. The Figure is taken from [27]

importance weight is:

$$\alpha_k^c = \overbrace{\frac{1}{N}\sum_i\sum_j \underbrace{\frac{\partial y^c}{\partial A_{i,j}^k}}_{\substack{\text{Gradients} \\ \text{via backprop}}}}^{\text{Global Average Pooling}} \tag{2.4.9}$$

where N is the total number of pixels in the feature map. Finally, to obtain the Grad-CAM map $L^c_{Grad-CAM}$ we multiply each activation map by its importance score and sum the values, within a ReLU activation function:

$$L^c_{Grad-CAM} = ReLU \underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}} \tag{2.4.10}$$

The ReLU activation function ensures that only features with a positive impact on the target class are considered. The output is thus a heat-map for the specified class, which is the same size as the feature map.

Typically Grad-CAM is a technique most commonly used for classification tasks. However, it is also possible to apply this technique to semantic segmentation tasks: we can compute the Grad-CAM map by replacing y^c with

$$\sum_{(i,j)\in S} y_{ij}^c, \tag{2.4.11}$$

where S is the set of pixels of interest and y_{ij}^c is 1 if pixel (i, j) is predicted to be of class

c, 0 otherwise [30].

Figure 2.4.7 provides a visual comparison between the heat-maps of Class Activation Maps (CAMs) and Gradient-weighted Class Activation Maps (Grad-CAMs).

2.5 Loss functions

This Section will present the main loss functions used to train the final model.

2.5.1 Generalized Dice loss

Generalized Dice Loss is a loss function designed to address the problem of class imbalance in medical segmentation tasks [32]. In this context, the foreground may be much smaller than the background, leading to class imbalance.

The Generalized Dice similarity coefficient measures the overlap between two segmented images, and it is based on Sørensen-Dice similarity:

$$SDS = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.5.1}$$

where X, Y are two sets and $|X|$, $|Y|$ are the cardinalities of the two sets.

Generalized Dice Loss tackles the class imbalance problem by controlling the contribution that each class makes to the similarity, and this is done by weighting each class:

$$GDL = 1 - \frac{2\sum_{k=1}^{K} w_k \sum_{m=1}^{M} Y_{km} T_{km}}{\sum_{k=1}^{K} w_k \sum_{m=1}^{M} Y_{km}^2 + T_{km}^2} \tag{2.5.2}$$

where Y is the image, T is the corresponding ground truth, K is the number of classes, M is the number of elements along the first two dimensions of Y, and w_k is a class-specific weighting factor. When working with imbalanced data sets, class weighting helps to prevent the more prevalent classes from dominating the similarity score [33].

2.5.2 Binary Cross-Entropy (BCE) loss

The Binary Cross-Entropy Loss is a loss function especially used in binary classification problems. The Binary Cross-Entropy (BCE) loss measures the divergence between predicted probabilities and true labels using the logarithmic loss function. It evaluates how

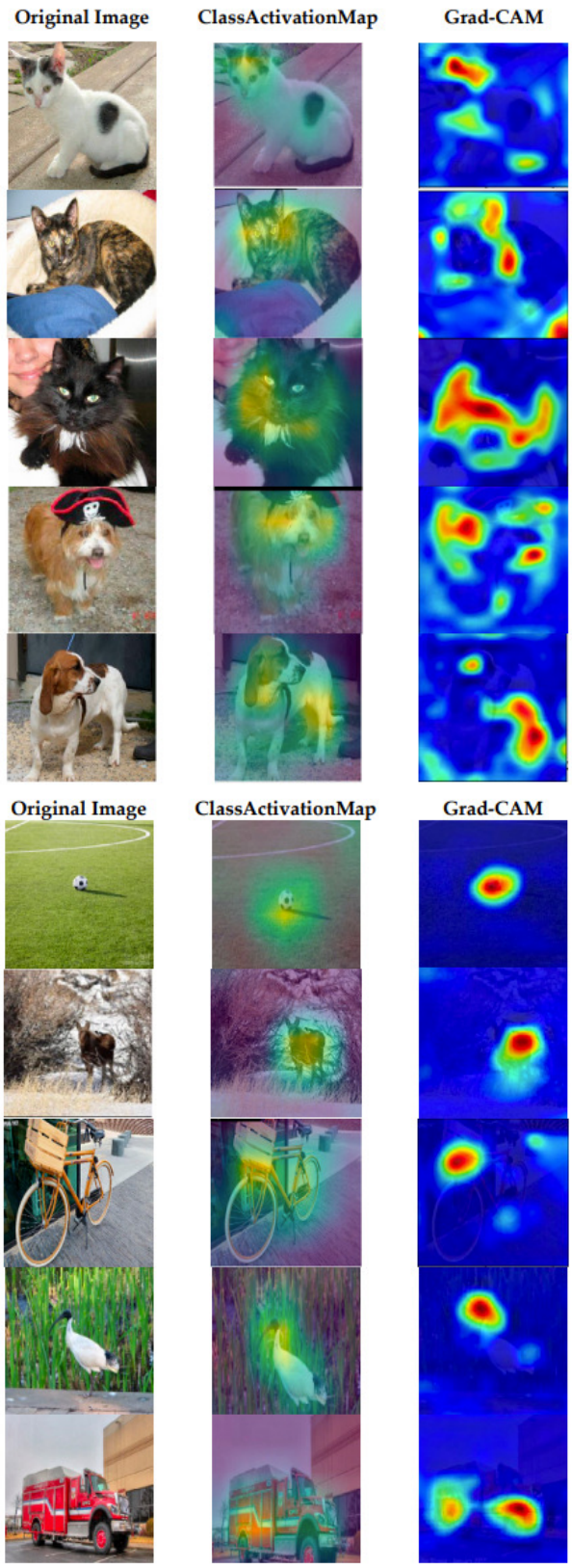


Figure 2.4.7: Comparison between CAM and Grad-CAM concerning the original image. The two Figures are taken and adapted from [31].

close or far the predicted probabilities are from the actual true values. It is computed as:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)] \tag{2.5.3}$$

where y_i is the prediction and p_i is the true one. This makes it an ideal choice for addressing binary classification problems, as it provides a measure of the model's capability to distinguish between the two classes.

2.6 Optimizer methods

This section will present the main optimizer methods used to train the final model.

2.6.1 Stochastic Gradient Descent with Momentum (SGDM)

Stochastic Gradient Descent with Momentum (SGDM) is an optimization algorithm used to train neural networks. It is a variant of the standard Stochastic Gradient Descent (SGD) algorithm, which introduces the concept of "momentum" to accelerate the convergence of the objective function and reduce the oscillation during optimization.

Initially, model parameters, such as weights and bias, are initialized randomly, to zero, or by specific methods such as Xavier/Glorot [34]. The network parameters (weights and bias) are updated using the Gradient Descent algorithm with momentum to minimize the loss function by iteratively moving a small amount in the negative gradient direction (the direction of maximum decrease in the objective function), updating the new momentum:

$$v^{(i+1)} = \alpha v^{(i)} - \frac{\eta^{(i)}}{K}\sum_{k=1}^{K}\nabla J(x_k, t_k; w^{(i)}) \tag{2.6.1}$$

$$w_{i+1} = w^{(i)} + v^{(i+1)} \tag{2.6.2}$$

where $v^{(i+1)}$ is the new momentum, $\eta^{(i)}$ is the learning rate, $\alpha v^{(i)}$ is the previous momentum. Finally, α is a factor that expresses the velocity: the larger α concerning the learning rate, the more previous gradients affect the current update, so we have a stronger momentum.

The key concept of momentum is that, unlike standard Stochastic Gradient Descent (SGD), which takes constant steps along the gradient, momentum accumulates the gradients of previous iterations into an (implicit) weighted sum, keeping track of the directions in which the gradient is headed. This allows the algorithm to overcome small local minima

in the loss function.

2.6.2 Adaptive Moment Estimation (Adam)

Adam (Adaptive Moment Estimation) is an optimization method introduced in [35] that computes the adaptive learning rate for each parameter by combining the concepts of momentum and adaptive gradient. The update rule is based on the value of the gradient at time t and the moving averages of the gradient and its square. More precisely, the Adam method defines the two Exponential Moving Averages (EMAs) m_t (first moment) and u_t (second moment) as:

$$m_t = \rho_1 m_{t-1} + (1 - \rho_1) g_t \tag{2.6.3}$$

$$u_t = \rho_2 u_{t-1} + (1 - \rho_2) g_t^2 \tag{2.6.4}$$

Where g_t is the gradient at time t, g_t^2 is the square of the gradient in terms of the square of its components, ρ_1 and ρ_2 are hyper-parameters representing the exponential decay rate for the first moment and the second moment (usually set at 0.9 and 0.999, respectively); initially the moments are initialized to 0: $m_t = u_t = 0$.

Since the values of the two moving averages may be minimal because of their initialization to zero, especially in the first iterations, the authors of the Adam method proposed a new version that presents a correction to the two moments:

$$\hat{m_t} = \frac{m_t}{(1 - \rho_1^t)} \tag{2.6.5}$$

$$\hat{u_t} = \frac{u_t}{(1 - \rho_2^t)} \tag{2.6.6}$$

Finally, the last update for each parameter θ_t of the network is:

$$\theta_t = \theta_{t-1} - \lambda \frac{\hat{m_t}}{\sqrt{\hat{u_t}} + \epsilon} \tag{2.6.7}$$

where is the learning rate, is a very small positive number to prevent a possible division by 0 (usually $\epsilon = 10 - 8$).

Chapter 3

Baseline Strategy: Network Architecture and Initial Improvement

3.1 Neural Network Architecture

The model tested uses a network with DeepLabV3+ architecture. DeepLabV3+ is an advanced encoder-decoder model that uses sophisticated convolutional neural networks for segmenting images and capturing precise details. This third version of the DeepLab project, developed by Google Research, has proven effective in addressing complex challenges related to image interpretation and producing highly accurate segmentations [36]. Figure 3.1.2 shows a total view of the architecture of DeepLabV3+.

The core of DeepLabV3+ is based on deep convolutional neural networks (CNNs), that work as a backbone, making use of a specific architecture as a feature extractor. Common choices for the basis are ResNet, MobileNet and Xception architecture. This choice allows the model to capture details at different levels of abstraction, greatly improving the accuracy of segmentation.

A key element of DeepLabV3+ is the use of atrous (or dilated) convolutions, which expand the model's field of view without swelling the overall number of parameters. This kind of convolution operation is used in the ASPP (Atrous Spatial Pyramid Pooling) module. Furthermore, to improve the quality of segmentation maps, DeepLabV3+ uses skip connections. These connections combine low-level features from the first layers of the network with high-level features from later layers. This combination helps maintain

fine details while incorporating global context.

The following Sections will go on to talk specifically about the encoder-decoder composition of the DeepLabV3+ model, focusing on the Atrous Spatial Pyramid Pooling (ASPP) module.

3.1.1 Atrous Spatial Pyramid Pooling (ASPP) module

The Atrous Spatial Pyramid Pooling (ASPP) module represents a crucial advancement within DeepLabV3+. This component combines different representations of the same image by using multiple parallel atrous convolutional layers (or dilated convolutional layers), each with different dilation rates; this allows the representation of the same feature using different receptive fields and catches various contextual details at multiple scales [37]. This integration enables the model to encompass objects of different sizes present in the image. The Atrous Spatial Pyramid Pooling (ASPP) module plays a pivotal role in achieving accurate segmentations across different scales of objects.

Within the ASPP module, as we already said, atrous convolutions play the role of extracting features from different areas of the image, but with the focus placed on different scales. The varying dilation rates of atrous convolutions determine the field of view of operations. Higher dilation rates correspond to a wider field of view, while lower rates allow finer details to be captured.

The dilated convolutions are very similar to the standard ones: they differ only by the presence of the dilation rate r, which expresses the space that must be skipped in the input matrix to obtain the desired convolution according to the formula:

$$S(x,y) = (I * K)(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} I(x + r \cdot i, y + r \cdot j) \cdot K(i,j) \tag{3.1.1}$$

where:

- $(I * K)(x,y)$ represents the value of the resulting pixel at the position (x,y) of the convoluted image,
- I is the image input (or feature map) in the convolutional layer,
- K is the kernel or convolutional filter,
- m and n are the dimensions of the kernel K,
- x and y are the coordinates of the pixel in the convolved image,

- r is the dilation rate, which determines the space between kernel pixels during convolution.

As we can see, these convolutions can be dilated to sample input pixels with adjustable spacing. This helps to capture information at different scales without increasing the computational cost too much. Figure 3.1.1 visually shows the dilated convolution operation.

Atrous convolutions are critical for understanding structures and relationships in the image: they maintain spatial resolution and capture local and global context within the image. At lower dilation levels, atrous convolutions capture fine, local details, such as contours or textures. At higher levels of dilation, they capture more general and global information, such as object shapes or backgrounds.

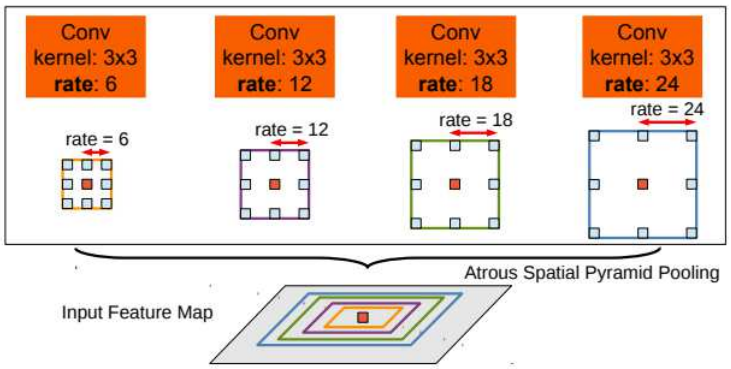


Figure 3.1.1: Atrous Spatial Pyramid Pooling (ASPP). To classify the centre pixel (orange), ASPP exploits multi-scale features by employing multiple parallel filters with different rates. The effective Field-Of-Views are shown in different colors [38].

3.1.2 Encoder-Decoder Architecture

The encoder-decoder architecture of DeepLabV3+ is carefully designed to address the challenges of semantic segmentation in images. This architecture leverages an intelligent combination of components to capture accurate details and improve image understanding at different scales.

The encoder is the starting point of this architecture. Here, the initial image is processed through a backbone neural network, that is composed of a series of convolutional layers, i.e. ResNet, MobileNet, and Xception architecture. Each layer is designed to detect specific features in the image, such as edges and textures. This progressive feature extraction helps the model to understand the image at a higher level by identifying key

elements. During this process, the image also undergoes size reductions through pooling.

After the encoder, the extracted features enter the Atrous Spatial Pyramid Pooling (ASPP) module. This module is crucial because it helps the model capture contextual details at different spatial scales.

In the decoding phase, the architecture focuses on detail reconstruction. The features obtained from the Atrous Spatial Pyramid Pooling (ASPP) module are enlarged through up-sampling operations. In this stage is important to restore details lost in previous steps and recover the original resolution of the image. Meanwhile, skip connections allow the model to carry information back from the encoder to the decoder, helping to maintain a consistent understanding of the image.

At the end of the architecture, we obtain a segmentation map that assigns semantic labels to pixels in the image. Each pixel is classified according to its class, enabling accurate segmentation.

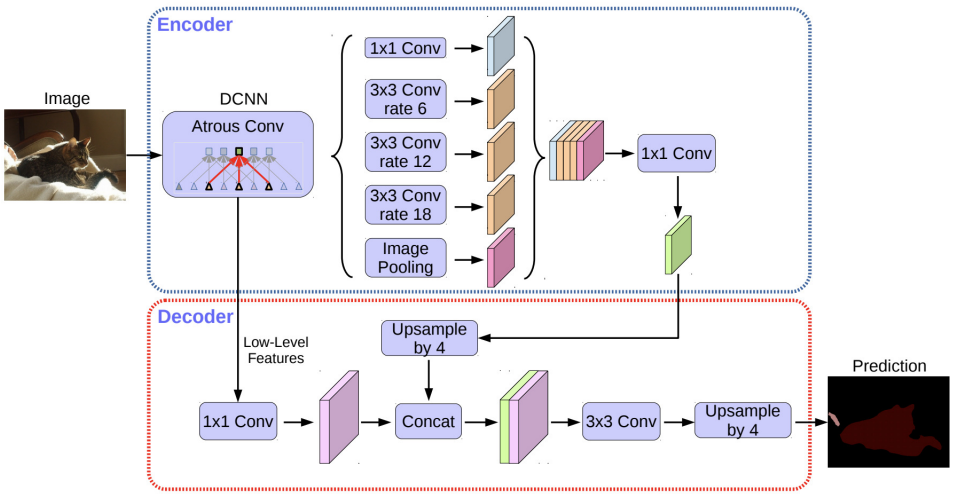


Figure 3.1.2: DeepLabV3+ architecture for semantic segmentation [39].

3.1.3 Details of architecture used

The adopted architecture includes the following components: first we have ResNet50 as the backbone, so the input feature map has a size of $224 \cdot 224 \cdot 3$; next there are various stages of encoder downsampling until we reach the feature map size of $14 \cdot 14 \cdot 2048$. After these steps, we find the Atrous Spatial Pyramid Pooling (ASPP) module section, where several layers extracted from different resolutions are depth-concatenated, resulting in feature maps of size $14 \cdot 14 \cdot 1024$.

In the decoder upsampling section, an initial volume of $56 \cdot 56 \cdot 256$ is obtained, finally arriving at the initial resolution of $224 \cdot 224 \cdot 2$, which represents our binary segmentation mask. The model has a total of about 43.9 million parameters (including weights and biases) that can be trained.

3.2 Initial Improvement of Masks

In this Section, we will provide an overview of the initial method employed to improve the performance of our model.

3.2.1 Iterative learning

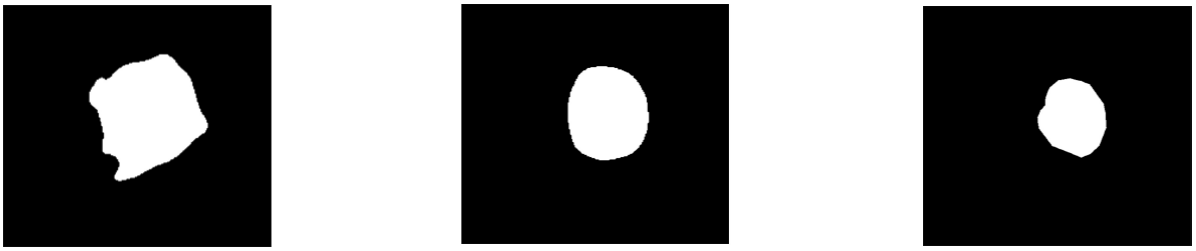
The chosen training approach used is based on weak supervision, where our network is trained on a specific dataset of colonoscopy images called the Polyp-Box-Seg dataset. To be precise, we selected 1040 images, constituting 80% of the total 1300 images (that have ground truth information), plus an additional set of 2770 images (that have only bounding box annotations).

Our training process follows an iterative methodology aimed at gradually refining the model's performance and capabilities. The beginning of each iteration is composed of two phases:

- the generation of the segmentation masks for images that lack them,
- the use of these new masks as targets during the training process of that specific iteration.

This iterative loop, characterized by the alternation between mask generation and training, is preserved for a predefined number of cycles (precisely, eight iterations). The general objective of this iterative process is the incremental enhancement and refinement of the quality and precision of the masks produced by the model.

Three images are shown in Figure 3.2.1: the first is a segmentation mask generated at the first iteration, followed by the one generated at the seventh iteration, ending with the target mask, which is what we aim to generate.



(a) Iteration n.1 **(b)** Iteration n.7 **(c)** Target mask

Figure 3.2.1: Segmentation masks at different stages of training [40].

3.2.2 Masks generation

To start the weakly supervised training process, it is necessary to generate the masks that will serve as targets for iterative learning, even though they initially will not be accurate. The generation of the masks is divided into several steps:

- Initial Mask Predictions: the mask generation process is based on predictions from a pre-trained model, represented by DeepLabV3 (ResNet50 as a backbone). This model has been trained using a fully supervised approach on the Kvasir-SEG dataset. All training details will be presented in the next Sections.
- Bounding Box Alignment: a key step involves the alignment of mask components with the confines of the corresponding bounding box. This procedure is illustrated in Figure 3.2.2, in the second row.
- Accuracy verification: this involves verifying that the generated mask completely covers no less than 30% of the entire area of the bounding box. Alternatively, in images where the mask generated does not meet the accuracy criteria established in the previous step, the mask will be considered inadequate and therefore replaced. This replacement involves the introduction of a circular mask, characterized by a diameter equivalent to 4/5 of the smaller dimension of the bounding box. This kind of mask is presented in Figure 3.2.2, in the third row.

3.2.3 Details of various iterations

First of all, it is necessary to list all the details of the training options used to train the pre-trained model on the Kvasir-SEG dataset: training was carried out with mini-batches of size 50 for a maximum of 25 epochs, with a learning rate of 0.01 characterized by a drop factor of 0.2, using the Stochastic Gradient Descent with Momentum (SGDM) method as an optimizer with momentum equal to 0.9. Afterwards, a Generalized Dice loss function

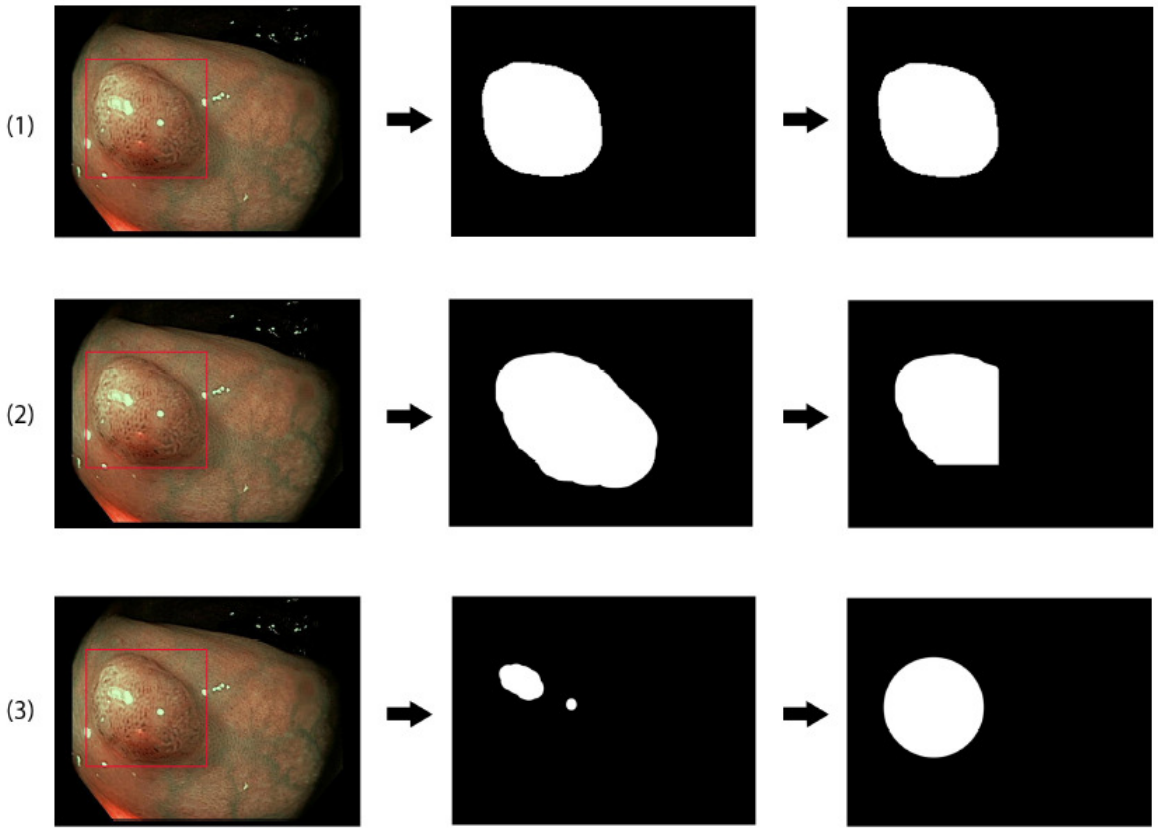


Figure 3.2.2: Example of model-generated masks, starting from the top: (1) the mask generated is accurate, (2) the generated mask needs to be cropped, (3) the mask generated occupies too little space of the bounding box, so it is discarded [40].

with an L2-Regularization factor equal to 0.005 is used.

Regarding the final baseline model, trained with weakly supervised learning, we use iterative training for a total of 8 iterations using Adam as an optimizer and Generalized Dice as a loss function, presented in Section 2.5.1. Specifically, the first 2 iterations use mini-batches size of 127 for a maximum of 20 epochs and a learning rate (LR) fixed at 0.0001. Then we proceed with 6 more iterations of a maximum of 6 epochs each, but in this case, the learning rate will be multiplied by 0.1 in order to reduce oscillations as the model becomes more accurate.

In all iterations, data augmentation has been applied. Generally, this is a strategy that allows us to reduce the phenomenon of over-fitting, and also it allows us to limit the problems consequent to having small datasets [41].

In detail, data augmentation consists of creating new data by applying transformations, generally translations, rotations, clippings, and horizontal and vertical flips to all

training images. In this thesis, we used: translations in a range of -20 and +20 pixels on the x-axis and y-axis, flips on both axes, rotations between -20 and +20 degrees, zoom-in and zoom-out for scale values between 0.9 and 1.2.

Chapter 4

Methods developed: Algorithms for Improving Segmentation Masks

4.1 Saliency Maps

The improvement methods involving saliency maps are based on the use of Matlab's Spectral Visual Saliency Toolbox, which contains the implementations of several state-of-the-art visual saliency detection algorithms.

In particular, we used the *spectral_saliency_multichannel* function, which implements spectral algorithms (FFT, DCT) for multi-channel saliency map computation. A detailed description of the function and its input parameters are as follows:

```
spectral_saliency_multichannel(
    image,
    saliency_map_resolution,
    algorithms,
    smap_smoothing_filter_params,
    cmap_smoothing_filter_params,
    cmap_normalization,
    extended_parameters,
    do_figures
);
```

where:

- *image* is the input image on which we compute the saliency map,

- *saliency_map_resolution* is the target saliency map resolution,
- *algorithms* is a list of spectral algorithms (several algorithms are available: 'fft', 'fft:whitening', 'fft:residual', 'dct', 'quat:fft:pqft', 'quat:fft:eigenpqft', 'quat:fft:eigensr', 'quat:dct', 'quat:dct:fast', 'fft:whitening:multi','fft:residual:multi', 'quat:dct:multi', 'quat:pqft:multi', 'quat:fft:eigensr:multi', 'quat:fft:eigenpqft:multi', 'itti', 'gbvs', 'ldrc', 'ldrccb'),
- *smap_smoothing_filter_params* controls the smoothing filter applied to the saliency map,
- *cmap_smoothing_filter_params* controls the smoothing filter applied to the conspicuity map,
- *cmap_normalization* controls the normalization of the conspicuity map;
- *extended_parameters* allows to specify any additional parameters specific to the algorithm used,
- *do_figures* controls whether or not output figures are displayed.

Specifically, two algorithms were used for this work among all available:

- *'dct'* uses the Discrete Cosine Transform (DCT) algorithms and DCT-based image signatures to calculate the saliency of each channel separately and then averages the result,
- *'itti'* uses the Itti-Koch method, explained in Section 2.4.2, to compute the saliency for the given image.

It should be added that for this type of approach, image pre-processing was done using an anisotropic diffusion filter, which has the advantages of reducing noise in images, improving contrast and emphasizing contours. It is a type of edge-preserving filter. Is widely used in medical image processing to improve the visibility of anatomical structures.

Each improvement method developed in this work was created from the support of the main refinement approach, explained in Section 3.2. No changes were made to the architectural parameters of the network and its training components.

4.1.1 Base Spectral Saliency Map

The Base Spectral Saliency Map approach (called SPECTRAL) is divided into three stages involving, respectively, the computation of the Spectral Saliency Map, its normal-

ization and the identification of false positives and false negatives, to improve the final segmentation mask:

1. The Spectral Saliency Map is calculated by modifying parameters such as *saliency_map_resolution* and *smap_smoothing_filter_params*: the resolution of a saliency map is set to $\frac{1}{4}$ of the initial image size, to have a good balance between a too coarse and a too precise resolution (note that the saliency maps are rescaled to the size of input images after its computation, to facilitate the successive procedures); finally a rotationally symmetric Gaussian lowpass filter of size equal to 9 with a standard deviation of 2.5 is used on the output saliency map;

2. Next, the Saliency Map is normalized. First, we calculated the dynamic range of the values in the saliency map (the difference between the maximum value and the minimum value), then we subtracted the minimum value from each element in the saliency map and finally divided it by the dynamic range; thus the saliency map includes values between 0 and 1:

$$\frac{\text{spectral\_saliency\_map} - \min(\text{spectral\_saliency\_map}(:))}{\max(\text{spectral\_saliency\_map}(:)) - \min(\text{spectral\_saliency\_map}(:))} \tag{4.1.1}$$

3. The final step concerns the identification of false positives and negatives given two specific thresholds t_1 and t_2; this is done by using two multiplicative constants k_1 and k_2 ($k_2 = 1 - k_1$), equal to 0.1 and 0.9 respectively, which are useful in calculating the above-mentioned thresholds:

$$\begin{aligned} t_1 &= k_1 \times max(saliency\_map) \\ t_2 &= k_2 \times max(saliency\_map) \end{aligned} \tag{4.1.2}$$

   where $max(saliency\_map)$ indicates the maximum value included in the saliency map.

   Specifically, false positives are detected if the corresponding value of the pixel in the mask is 1 and if the value of the saliency map in the same pixel is inferior or equal to the threshold t_1, and therefore it is not significantly important at the saliency level.

   On the other hand, false negatives are detected if the corresponding value of the pixel in the mask is 0 and if the value of the saliency map of the same pixel is greater than or equal to the threshold t_2, and so, it is important at the saliency level.

   Finally, we compute the final binary segmentation mask given by the union between

the generated mask (the mask given by the initial refinement) and the pseudo mask computed in this step.

Algorithm 1: Spectral Saliency Map

```
saliency_map_resolution ← [size(img)*0.25, size(img,2)*0.25];
smap_smoothing_filter_params ← 'gaussian', 9, 2.5;
saliency_map ← spectral_saliency_multichannel('dct', ...);
saliency_map ← (saliency_map − min(saliency_map(:)))/(max(saliency_map(:
  )) − min(saliency_map(:)));
saliency_map ← resize(saliency_map);
k_1 ← 0.1;
k_2 ← 1 - k_1;
threshold ← k_1 × max(saliency_map);
false_positives ← (pred_mask == 1) ∧ (saliency_map ≤ threshold);
cleaned_mask ← pred_mask;
cleaned_mask (false_positives) ← 0;
threshold ← k_2 × max(saliency_map);
false_negatives ← (pred_mask == 0) ∧ (saliency_map ≥ threshold);
cleaned_mask (false_negatives) ← 1;
ref_mask ← cleaned_mask ∨ pred_mask;
```

(a) **(b)** **(c)** **(d)**

Figure 4.1.1: Examples of segmentations: (a) input image, taken from the dataset CVC-ClinicDB [42], (b) ground truth respective to the image, (c) mask obtained by the basic method, explained in Section 3.2, (d) mask obtained by Spectral approach.

An example of segmentation masks is shown in Figure 4.1.1 so that comparisons can be made. In addition, Algorithm 1 is a brief pseudocode representing the approach used for improving a segmentation mask by use of the Spectral saliency map.

4.1.2 Superpixels Spectral Saliency Map

The Superpixels Spectral Saliency Map approach (called SPECTRAL_SUPERPIXELS) is divided into four stages involving, respectively, the computation of the N superpixel

segments of the 2-D image, the computation of the Spectral Saliency Map, its normalization and the identification of the most salient superpixel regions, to improve the final segmentation mask:

1. The N superpixels of the input image are computed by using the Simple Linear Iterative Clustering (SLIC) algorithm [43], which groups pixels into regions with similar values; after this step, we will obtain a series of segments of pixels from the input image; N is set to 120;
2. Next, the Spectral Saliency Map is computed as before by modifying *saliency_map_resolution* and *smap_smoothing_filter_params* parameters;
3. After that, the Saliency Map is normalized as before by using Formula 4.2.2;
4. The final step concerns the identification of the most salient superpixel regions; this step is performed by determining a global threshold using Otsu's method [44] and analyzing the average of the pixel values contained in each image segment; the segment is added to the final segmentation mask if this average is greater than or equal to the previously calculated global threshold. Finally, we compute the final binary segmentation mask given by the union between the generated mask (the mask given by the initial refinement) and the pseudo mask computed in this step.

Algorithm 2: Superpixels Spectral Saliency Map

```
saliency_map_resolution ← [size(img,1), size(img,2)];
smap_smoothing_filter_params ← 'gaussian', 9, 2.5;
num_superpixels ← 120;
segments ← SLIC_superpixels(img, num_superpixels);
ref_mask ← zeros(size(img,1), size(img,2));
saliency_map ← spectral_saliency_multichannel('dct', ...);
saliency_map ← (saliency_map − min(saliency_map(:)))/(max(saliency_map(:
  )) − min(saliency_map(:)));
threshold ← OTSU_threshold(saliency_map);
for i = 1 to num_superpixels do
    segment_mask ← (segments == i);
    median_segment_mask ← mean(saliency_map (segment_mask));
    if median_segment_mask ≥ threshold then
        ref_mask ← ref_mask + segment_mask;
ref_mask ← ref_mask ∨ pred_mask;
```

An example of segmentation masks is shown in Figure 4.1.2 so that comparisons can be made. In addition, Algorithm 2 is a brief pseudocode representing the approach used

(a) **(b)** **(c)** **(d)**

Figure 4.1.2: Examples of segmentations: (a) input image, taken from the dataset CVC-ClinicDB [42], (b) ground truth respective to the image, (c) mask obtained by the basic method, explained in Section 3.2, (d) mask obtained by Spectral and Superpixels segmentation approach.

for improving a segmentation mask by use of the Spectral saliency map with Superpixels segmentation.

4.1.3 Itti-Koch Spectral Saliency Map

The Itti-Koch Spectral Saliency Map approach (called SPECTRAL_ITTI) is divided into five stages involving, respectively, the computation of the Itti-Koch Saliency Map, the computation of the Spectral Saliency Map, their normalization, the computation of a saliency map obtained as a weighted sum between them, and the identification of false positives and false negatives to improve the final segmentation mask:

1. The Itti-Koch Saliency Map is computed by modifying parameters such as *saliency_map_resolution* and *smap_smoothing_filter_params*: since this type of map brings with it many features regarding colour, intensity and orientation of pixels, the resolution was set to $\frac{1}{4}$ of the original image size; this was done to balance the large amount of information computed by this method; finally a rotationally symmetric Gaussian lowpass filter of size equal to 9 with a standard deviation of 2.5 was used on the output saliency map, as before;

2. Next, the Spectral Saliency Map is computed as before by modifying *saliency_map_resolution* and *smap_smoothing_filter_params* parameters;

3. After that, the Itti-Koch and Spectral Saliency Map are normalized as before by using Formula 4.2.2;

4. Then takes place the computation of a single and unique saliency map given by the weighted sum of the two previously computed and normalized maps: two weights w_1, w_2 were used, with w_1 set to 0.6 and w_2 to 0.4 ($w_2 = 1 - w_1$); the saliency map

is computed as follows:

$$saliency\_map = w_1 \times itti\_saliency\_map + w_2 \times spectral\_saliency\_map; \quad (4.1.3)$$

It was chosen to give more weight to the Itti-Koch Saliency Map than to the Spectral Saliency Map, because of the information content that the map offers;

5. The final step concerns the identification of false positives and negatives given two specific thresholds t_1 and t_2 as before; this is done by using two multiplicative constants k_1 and k_2, equal to 0.1 and 0.9 ($k_2 = 1 - k_1$); the thresholds are computed as in Formula 4.1.2. Finally, we compute the final binary segmentation mask given by the union between the generated mask (the mask given by the initial refinement) and the pseudo mask computed in this step.

Algorithm 3: Itti-Koch Spectral Saliency Map

```
saliency_map_resolution ← [size(img,1)*0.25, size(img,2)*0.25];
smap_smoothing_filter_params ← 'gaussian', 9, 2.5;
itti_saliency_map ← spectral_saliency_multichannel('itti', ...);
itti_saliency_map ← (itti_saliency_map − min(itti_saliency_map(:
  )))/(max(itti_saliency_map(:)) − min(itti_saliency_map(:)));
spectral_saliency_map ← spectral_saliency_multichannel('dct', ...);
spectral_saliency_map ← (spectral_saliency_map − min(spectral_saliency_map(:
  )))/(max(spectral_saliency_map(:)) − min(spectral_saliency_map(:)));
w1 ← 0.6;
w2 ← 1 - w1;
combined_map ← w1 ×itti_saliency_map + w2 ×spectral_saliency_map;
combined_map ← resize(combined_map, [size(img,1), size(img,2)]);
k1 ← 0.1;
k2 ← 1 - k1;
threshold ← k1 × max(combined_map);
false_positives ← (pred_mask == 1) ∧ (combined_map ≤ threshold);
cleaned_mask ← pred_mask;
cleaned_mask (false_positives) ← 0;
threshold ← k2 × max(combined_map);
false_negatives ← (pred_mask == 0) ∧ (combined_map ≥ threshold);
cleaned_mask (false_negatives) ← 1;
ref_mask ← cleaned_mask ∨ pred_mask;
```

An example of segmentation masks is shown in Figure 4.1.3 so that comparisons can be made. In addition, Algorithm 3 is a brief pseudocode representing the approach used for improving a segmentation mask by use of the Spectral and Itti-Koch saliency map.

(a) (b) (c) (d)

Figure 4.1.3: Examples of segmentations: (a) input image, taken from the dataset CVC-ClinicDB [42], (b) ground truth respective to the image, (c) mask obtained by the basic method, explained in Section 3.2, (d) mask obtained by Spectral and Itti-Koch approach.

4.2 Class Activations Maps (CAMs)

The improvement methods involving Gradient-Class Activations Maps (Grad-CAMs) are based on the use of Matlab's function *gradCAM*, which computes the Grad-CAM importance map, where areas in the map with higher positive values correspond to regions of input data that contribute positively to the prediction. A detailed description of the function and its input parameters are as follows:

```
gradCAM(
    net,
    X,
    classNames,
    featureLayer,
    reductionLayer,
    OutputUpsampling
);
```

where:

- *net* is the trained network;
- X is the input data, specified as a numeric array. For 2-D image data, X must be a h-by-w-by-c array, where h, w, and c are the height, width, and number of channels of the network input layer, respectively;
- *classNames* is a list of the class labels to use for computing the Grad-CAM map;
- $featureLayer$ is the name of the feature layer used to extract the feature map when computing the Grad-CAM map;
- *reductionLayer* is the name of the reduction layer used to extract output activations

when computing the Grad-CAM map;

- *OutputUpsampling*, is the output upsampling method. For these method, the *'bicubic'* interpolation was to produce a smooth map;

Each improvement method developed in this work was created from the support of the main refinement approach, explained in Section 3.2. Consequently, the architectural and training parameters were not changed.

4.2.1 Gradient-Class Activations Maps (Grad-CAMs): first variant

The first variant of the Gradient-Class Activations Maps (Grad-CAMs) approach (called GRADv1) is divided into six stages involving, respectively, the adaptation of the image to the input layer of the pre-trained network, the computation of two Grad-CAM maps based on specific feature layers, the computation of a score map obtained as a weighted sum between the two previous maps, their normalization and, finally, the computation of a binary segmentation mask obtained as the intersection between using a threshold map and the mask predicted by the model:

1. The adaptation of the image to the input layer of the pre-trained model is done by resizing the image using the specific dimensions of the model's input layer;

2. Next, there is the computation of two Grad-CAM maps: the first, which we will call score_map_1 was computed using the 'catAspp' layer of the model as the feature layer. The 'catAspp' layer is a deep concatenation of the outputs of four previous Re-LU layers present in the ASPP module, explained in Section 3.1.1; the choice to use this layer was made because typically the last layer that collects the outputs of the ReLU layers is used as feature layer and also because in this way we provide a map with information about where the network puts most attention in the image. The second, which we will call score_map_2 was computed using the 'dec_relu4' layer as a feature layer, which corresponds to the fourth ReLU output layer of the model's decoder; the choice of this layer was because it gives us information about the shape of the segmentation. As a reduction layer, the 'softmax-out' layer was used for both activation maps: typically, for classification and segmentation tasks, this layer is usually the final soft-max layer;

3. Then takes place the computation of a single and unique activation map given by the weighted sum of the two previous maps: two weights w_1, w_2 were used, with w_1

set to 0.3 and w_2 to 0.7 ($w_2 = 1 - w_1$); the activation map is computed as follows:

$$score\_map = w_1 \times score\_map\_1 + w_2 \times score\_map\_2 \tag{4.2.1}$$

It was chosen to give more weight to the map that contains information regarding the shape of the segmentation because the network attention information is very strong;

4. After that, the activation map is normalized. First, we calculated the dynamic range of the values in the saliency map (the difference between the maximum value and the minimum value), then we subtracted the minimum value from each element in the saliency map and finally divided it by the dynamic range; thus the saliency map includes values between 0 and 1:

$$\frac{score\_map - min\_value}{max\_value - min\_value} \tag{4.2.2}$$

where *score_map* is the activation map, *min_value* is the minor element inside the map and *max_value* the maximum;

5. Next, the threshold activation map is computed. A global threshold such as Otsu's threshold was used to threshold the activation map obtained from the previous step; then the resulting mask was resized with the initial dimensions of the input images;

6. Finally, the final mask is obtained by taking the common pixels between the mask predicted by the model and the mask obtained from the previous step;

(a) **(b)** **(c)** **(d)**

Figure 4.2.1: Examples of segmentations: (a) input image, taken from the dataset CVC-ClinicDB [42], (b) ground truth respective to the image, (c) mask obtained by the basic method, explained in Section 3.2, (d) mask obtained by Grad-CAM (first variant) approach.

An example of segmentation masks is shown in Figure 4.2.1 so that comparisons can be made. In addition, Algorithm 4 is a brief pseudocode representing the approach used for improving a segmentation mask by use of the Grad-CAM.

Algorithm 4: Gradient-Class Activations Maps (Grad-CAMs): first variant

```
input_size ← net.Layers(1).InputSize(1);
resized_img ← resize(img, [input_size, input_size ]);
class_names ← ['one', 'zero'];
feature_layer_1 ← 'catAspp';
reduction_layer ← 'softmax-out';
upsample_type ← 'bicubic';
score_map_1 ← gradCAM(net, resized_img, class_names, reduction_layer,
  feature_layer_1, upsample_type);
class_names ← ['one', 'zero'];
feature_layer_2 ← 'dec_relu4';
score_map_2 ← gradCAM(net, resized_img, class_names, reduction_layer,
  feature_layer_2, upsample_type);
w_1 ← 0.3;
w_2 ← 1 - w_1;
weighted_score_map ← (w_1 × score_map_1 + w_2 × score_map_2)/2;
norm_weighted_score_map ← normalizeGradCAM(weighted_score_map);
threshold ← OTSU_threshold(img);
thresh_score_map ← norm_weighted_score_map ≥ threshold;
thresh_score_map ← resize(thresh_score_map,[size(img,1), size(img,2)]);
final_mask ← thresh_score_map ∧ pred_mask;
```

4.2.2 Gradient-Class Activations Maps (Grad-CAMs): second variant

The second variant of Gradient-Class Activations Maps (Grad-CAMs) approach (called GRADv2) is divided into six stages involving, respectively, the adaptation of the image to the input layer of the pre-trained network, the computation of some Grad-CAM maps based on specific feature layers, the computation of a score map obtained as the average of the previous maps, their normalization and, finally, the computation of a binary segmentation mask obtained as the intersection between a threshold map and the mask predicted by the model:

1. The adaptation of the image to the input layer of the pre-trained model is done as before;

2. Next, there is the computation of some Grad-CAM maps: different encoder activation layers were used, starting from intermediate to deep layers, along with the corresponding decoder ones. In order, the following layers were used as feature layers: some intermediate convolutional layers such as 'res3c_branch2b', 'res4c_branch2b' and corresponding Re-LU output layers such as 'activation_19_relu', 'activation_31_

relu': these layers lie between the first layers, which capture low-level features such as edges and colours, and the deeper layers, which capture high-level features. The middle layers can represent a combination of low-level and high-level features.
Next, deeper convolutional and Re-LU output layers such as 'res5c_branch2b', 'activation_49_relu', and 'catAspp': deeper convolution layers tend to capture higher-level features and more complex image concepts; may contain more abstract and discriminating information for our classes of interest. Generally, activations layers are useful for highlighting regions with strong activations.
Finally, we have 'dec_c2', 'dec_relu2', 'dec_c4', 'dec_relu4', and 'dec_crop2': these layers represent the last stages of your network and are responsible for transposing the convolutional output to achieve higher resolution segmentation. By using these layers, we can have a better localization of discriminating features for the 'one' and 'zero' classes in our image.
As a reduction layer, the 'softmax-out' layer was used for both activation maps, as before;

3. Then takes place the computation of a single and unique activation map given by the average of the previous maps;
4. After that, the activation map is normalized as before;
5. Next, the threshold activation map is computed as before by using Otsu's threshold;
6. Finally, the final mask is obtained by taking the common pixels between the mask predicted by the model and the mask obtained from the previous step;

(a) **(b)** **(c)** **(d)**

Figure 4.2.2: Examples of segmentations: (a) input image, taken from the dataset CVC-ClinicDB [42], (b) ground truth respective to the image, (c) mask obtained by the basic method, explained in Section 3.2, (d) mask obtained by Grad-CAM (second variant) approach.

An example of segmentation masks is shown in Figure 4.2.2 so that comparisons can be made. In addition, Algorithm 5 is a brief pseudocode representing the approach used for improving a segmentation mask by use of the Grad-CAM.

Algorithm 5: Gradient-Class Activations Maps (Grad-CAMs): second variant

```
input_size ← net.Layers(1).InputSize(1);
resized_img ← resize(img, [input_size, input_size ]);
class_names ← ['one', 'zero'];
feature_layers ← [ 'res3c_branch2b', 'activation_19_relu', 'res4c_branch2b',
    'activation_31_relu', 'res5c_branch2'", 'activation_49_relu', 'catAspp', 'dec_c2',
    'dec_relu2', 'dec_c4', 'dec_relu4', 'dec_crop2', ];
reduction_layer ← 'softmax-out';
upsample_type ← 'bicubic';
combine_grad_map ← zeros(size(resized_img, 1), size(resized_img, 2));
for i ← 1 to numel(feature_layers) do
    feature_layer ← feature_layers (i);
    score_map ← gradCAM(net, resized_img, class_names, reduction_layer,
     feature_layer, upsample_type);
    combine_grad_map ← combine_grad_map + score_map;
combine_grad_map ← combine_grad_map/numel(feature_layers);
norm_score_map ← normalizeGradCAM(combine_grad_map);
threshold ← 0.4 × max(norm_score_map);
thresh_score_map ← norm_score_map ≥ threshold;
thresh_score_map ← resize(thresh_score_map, [size(img,1), size(img,2)]);
final_mask ← thresh_score_map ∧ pred_mask;
```

4.2.3 Gradient-Class Activations Maps (Grad-CAMs) and Adaptive Region Growing (ARG)

The variant of Gradient-Class Activations Maps (Grad-CAMs) approach that uses also the Adaptive Region Growing algorithm (called GRAD_ARG) takes inspiration from the work of Yuhan Xie et al. [45] and it is divided into three parts: detection, growth and segmentation. In the detection part, we exploit the information given by the weak labels, i.e., the bounding boxes; then we carry out the computation of Grad-CAM on the image given the bounding box to generate a meaningful heat-map representing the network's attention in the image box. In the growth part, we use a highly thresholded heat-map as initial seed points; according to the grayscale information in the bounding box, an adaptive regional growth is performed to obtain pseudo-labels. In the segmentation part, we compute the final binary segmentation mask given by the intersection between the generated mask and the pseudo mask after post-processing:

1. In the detection part, we exploited all the information given by the weak labels, i.e., the bounding boxes, by cropping the input image and considering only the part contained in the bounding boxes. In this image, we calculated an activation map

using Grad-CAM, making use of the 'dec_relu4' layer as the feature layer, 'softmax-out' as the reduction layer and 'bicubic' interpolation as the output upsampling method. The obtained heat-map was then normalized using the Formula in 4.2.2 and thresholded with a high confidence threshold of about 80%. Considering that there may be positioning errors in the heat-map, we exploit grayscale image information to remove some seed points: we first compute the Otsu's threshold θ for the image box, and then we eliminate all seed points below this threshold, to guarantee the accuracy of seed points. These can be described with the following Formula:

$$rect\_heatmap(x,y)' = \begin{cases} 1, rect\_heatmap(x,y) \geq \theta \\ 0, rect\_heatmap(x,y) \leq \theta \end{cases} \tag{4.2.3}$$

The resulting binary map represents the initial seed points used as input to the Adaptive Region Growing algorithm.

2. After that, there is the growth part, in which the Adaptive Region Growing (ARG) algorithm is involved. The idea behind this algorithm is the concept of connectivity: starting from initial seed points, group pixels or sub-regions into larger regions based on a merging rule (or predicate) until a stopping rule is reached.
   We set the stopping condition of adaptive region growth as:

$$\begin{aligned} |P_{new} - P_{\bar{S}}| &\leq \epsilon \\ P_{new} &\geq \frac{P_{\bar{f}} + P_{\bar{b}}}{2} \\ \epsilon &= (P_{\bar{f}} - P_{\bar{b}}) \times ratio \end{aligned} \tag{4.2.4}$$

   where P_{new} denotes the grayscale of new point in growing, $P_{\bar{S}}$ denotes the average grayscale of seeds, $P_{\bar{f}}$ and $P_{\bar{b}}$ denote the average grayscale of foreground and background and *ratio* denotes a parameter used to include in the growth pixels of background (we set *ratio* to 0.2). The union of Formulas 4.2.4 constitutes the boundary conditions for growth. The pseudo-code of the Adaptive Region Growing algorithm for pseudo mask generation is shown in Algorithm 6;

3. Next we have the segmentation part where the final binary segmentation mask is generated: after generating the pseudo mask we decided to apply post-processing to it, which consists of identifying the connected components (blobs), among them considering only the largest ones and apply a morphological dilate operation on the final mask. As a last step we performed an intersection operation between the mask predicted and refined by the approach explained in the Section 3.2 and the pseudo

mask generated in this step.

This approach differs from the others because a slight change was made in the training components: the Binary Cross-Entropy (BCE) loss function, explained in Section 2.5.2, was summed to the Generalized Dice loss function, explained in Section 2.5.1.

Algorithm 6: Adaptive Region Growing algorithm for pseudo mask generation

Input : Initial seed points P_{seeds}
Output : Pseudo mask P_{grown}
Initialization: grown points set $P_{\text{grown}} = P_{\text{seeds}}$; directions set $D = \{(-1,-1),(0,-1),(1,-1),(1,0),(1,1),(0,1),(-1,1),(-1,0)\}$;

```
for row in P_seeds do
    for col in P_seeds do
        P_seed = P_seeds[0];
        foreach direction in D do
            P_new = P_seed + direction;
            if (|P_new − P_S̄| ≤ ϵ or P_new ≥ (P_f̄ + P_b̄)/2) then
                P_seeds = P_seeds ∪ P_new;
                P_grown = P_grown ∪ P_new;
return P_grown
```

Algorithm 7: Gradient-Class Activations Maps (Grad-CAMs) and Adaptive Region Growing (ARG)

```
pred_mask ← get_mask(net, img);
generated_mask ← initial_refinement(pred_mask);
img_box ← rect_bb(img);
seed_pints ← get_seed_points(img_box);
pseudo_mask ← adaptive_region_growing(seed_pints);
final_mask ← compute_mask_union(pseudo_mask, generated_mask);
```

An example of segmentation masks is shown in Figure 4.2.3 so that comparisons can be made. In addition, Algorithm 7 is a brief pseudocode representing the approach used for improving a segmentation mask by use of the Grad-CAM and Adaptive Region Growing.

(a) **(b)** **(c)** **(d)**

Figure 4.2.3: Examples of segmentations: (a) input image, taken from the dataset CVC-ClinicDB [42], (b) ground truth respective to the image, (c) mask obtained by the basic method, explained in Section 3.2, (d) mask obtained by Grad-CAM and Adaptive Region Growing approach.

Chapter 5

Experiments and Results

5.1 Datasets

Pixel-wise image segmentation is a highly demanding task in medical image analysis. In practice, it is difficult to find annotated medical images with corresponding segmentation masks. For this reason, this kind of problem requires high-quality training datasets. The types of data available for colorectal polyp detectors include, for the most part, static photos extracted from live video streams.

This Section analyzes the standard datasets typically used to test polyp detection models, and the dataset used to train our model. In Table 5.1.1, a summary of all the datasets is presented. In order to be able to test our developed models and understand their performance, we used the following datasets:

- Kvasir-SEG: The Kvasir-SEG dataset (size 46.2 MB) is an open-access dataset of 1000 gastrointestinal polyp images and their corresponding ground truth segmentation mask, manually annotated by a medical doctor and then verified by an experienced gastroenterologist [46]. The resolution of the images contained in Kvasir-SEG varies from 332x487 to 1920x1072 pixels.
- CVC-ClinicDB: The CVC-ClinicDB (size 5.22 MB) is an open-access dataset containing 612 images of polyps associated segmentation mask manually produced by experts with a resolution of 384×288, taken from 31 colonoscopy sequences [42]. Typically used for testing due to the limited number of samples.
- CVC-ColonDB: The CVC-ColonDB (size 91.2 MB) is another polyps dataset containing 380 images with the respective segmentation mask, with a resolution of

574x500 [47]. Typically used for testing due to the limited number of samples.

- CVC-300: The CVC-300 (size 31.7 MB) is a dataset consisting of 300 colorectal endoscopy images, a subset of CVC-ColonDB [47]. Typically used for testing due to the limited number of samples.
- ETIS-LaribPolypDB: The ETIS-LaribPolypDB (size 176 MB) dataset contains 196 images with a fixed resolution of 1225x966, with polyps extracted from 34 sequences with 44 different polyps [48]. Typically used for testing due to the limited number of samples.

In order to train our developed models, we used the following dataset:

- Polyp-Box-Seg: The Polyp-Box-Seg dataset (size 46.2 MB) contains 4070 colonoscopy images with 640x480 resolution from over 2000 patients. A subset of 1300 elements is equipped with the corresponding manually annotated mask, while the remaining component has a bounding box. The images in this dataset were hand-selected from videos of colonoscopies so that each image contains a unique polyp. This is to reduce the correlation between images and thus to decrease the possible intra-patient polyp similarity bias. The images in this dataset contain polyps of different sizes and morphologies and come from each portion of the colorectum [49].

Table 5.1.1: Summary of all Colon-rectal Polyp Datasets used.

Dataset	Number of Samples	Resolution	Size (MB)
Kvasir-SEG	1000	332x487 to 1920x1072	46.2
Polyp-Box-Seg	4070	640x480	46.2
CVC-ClinicDB	612	384x288	5.22
CVC-ColonDB	380	574x500	91.2
CVC-300	300	574x500	31.7
ETIS-LaribPolypDB	196	1225x966	176

5.2 Performance Indicators

This Section will present the performance indicators used to assess the models created.

All the metrics are based on the computation of a confusion matrix for a binary segmentation mask, which contains the number of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values. The kind of prediction ranges span from zero (worst) to one (best). Our goal is to score the similarity between the predicted segmentation (actual prediction) and the labelled segmentation (actual ground truth).

First of all, the confusion matrix is a tabular representation used to evaluate the performance of a classification model, especially in binary classification problems (where there are two classes). The matrix shows the relationship between the predictions made by the model and the true class labels of the data, as we can see from Figure 5.2.1.

The confusion matrix is composed of 4 key elements:

- True Positive (TP): represents the number of times in which the model correctly predicts the positive class,
- False Positive (FP): represents the number of times in which the model correctly predicted the positive class when it was negative,
- True Negative (TN): represents the number of cases in which the model correctly predicted the negative class,
- False Negative (FN): represents the number of times in which the model did not correctly predict the negative class when it was positive.

Now we can define the evaluation metrics used for our models in the testing phase:

- Intersection over Union (IoU, also called Jaccard coefficient): is the area of the intersection over the union of the predicted segmentation and the ground truth:

$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}} = \frac{TP}{TP + FP + FN} \tag{5.2.1}$$

- Dice: is calculated by dividing twice the area of the intersection between the predicted mask and the ground truth mask by the sum of the areas of the two masks:

$$Dice = \frac{2 \times \text{Area of overlap}}{\text{Total Area}} = \frac{2TP}{2TP + FP + FN} \tag{5.2.2}$$

- Precision: is the number of True Positive samples divided by all positive samples:

$$Precision = \frac{TP}{TP + FP} \tag{5.2.3}$$

- Recall: is the number of True Positive samples divided by the number of all samples that should have been identified as positive:

$$Recall = \frac{TP}{TP + FN} \tag{5.2.4}$$

		Predicted	
		Positive	**Negative**
Actual	**Positive**	TP	FN
	Negative	FP	TN

Figure 5.2.1: The Confusion Matrix.

- F2 Score: is a measure that assigns greater weight to accuracy than to recall:

$$F2 = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{5.2.5}$$

  where β is a factor that controls the importance of precision versus recall. A larger β value assigns greater weight to precision.

- Accuracy: is the number of correct predictions, that is the number of correct positive and negative predictions divided by the total number of predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2.6}$$

5.3 Training and Testing Protocol

This Section will briefly present the protocols used for training and testing the created models, including the backbone neural network.

First, the backbone was trained on the Kvasir-SEG dataset. To evaluate the goodness, and thus the various metrics, of the models we created, was used a specific training and testing protocol. In the training protocol, the model underwent a training phase using the Polyp-Box-Seg dataset, consisting of the images specified in Section 3.2.1 (80% of the total images). Next, all models have been tested in the same Polyp-Box-Seg dataset (20% of the total images).

The testing protocol involved the preliminary test phase of the models on the five datasets typically used in the medical field of colorectal polyp segmentation, called CVC-ClinicDB, CVC-ColonDB, CVC-300, ETIS-LaribDB; here, to accelerate the development of the various models, the most promising ones in the preliminary test phase were then used in the iterative learning, with the procedure previously described.

5.4 Working Environment

For the actual implementation of the models, MATLAB 9.14 (R2023a) was used, equipped with Computer Vision Toolbox, Deep Learning Toolbox, Image Processing Toolbox, Parallel Computing Toolbox, Statistics and Machine Learning Toolbox, Image Processing Toolbox, Spectral Saliency Toolbox. Along with these Toolboxes, the Graph-Based Visual Saliency (GBVS) package was installed on MATLAB to take advantage of the Itti-Koch Saliency Map computation.

The machine used for training is the 'blade' server installed at our Department, configured to satisfy computing resource requests from users; specifically, the train and test protocol was performed on an Nvidia Titan RTX GPU or Nvidia RTX 3090 GPU, with 10G of maximum memory.

The training process took a different number of total hours, depending substantially on the type of model trained. The baseline took 5 hours for full training; methods relying on a saliency map took 10-12 hours for full training; methods relying on Grad-CAM took 36-38 hours, except Adaptive Region Growing, which instead took 10-11 hours to complete the training.

5.5 Results

In this Section, we report the experimental results carried out to assess the various approaches developed, explained in the previous chapter. Before going into the details of the specific results obtained from each model developed and the related tables and graphs, it is important to highlight again the general context of the work, including the work environment, explained in Section 5.4, and the training and testing protocols, explained in Section 5.3.

We will begin with an overall presentation of the performances of the developed models, including the baseline, and the main trends observed in the results followed by a brief examination of them. Then, we will move on to a discussion of the results obtained, making an analysis based on metrics listed in Section 5.2 and a comparison with the initial baseline refinement method.

5.5.1 Results presentation

We will now present the results obtained from the models developed. For a clear understanding, we will present graphs showing the trend of the metrics used during the various

re-trainings of iterative learning. This presentation of results will allow us to assess any benefits given by iterative learning, to understand how each model performed in specific situations, and to understand the differences between the various approaches. Graph 5.5.1 shows the trend of the BASELINE model's metrics. Instead, Table 5.5.1 shows a summary of the performance of the metrics respectively to the BASELINE model. As we can see, the values are very stable, undergoing little change, but no improvement, except for the Recall metric and consequently F2-Score where the improvement is slightly more substantial.

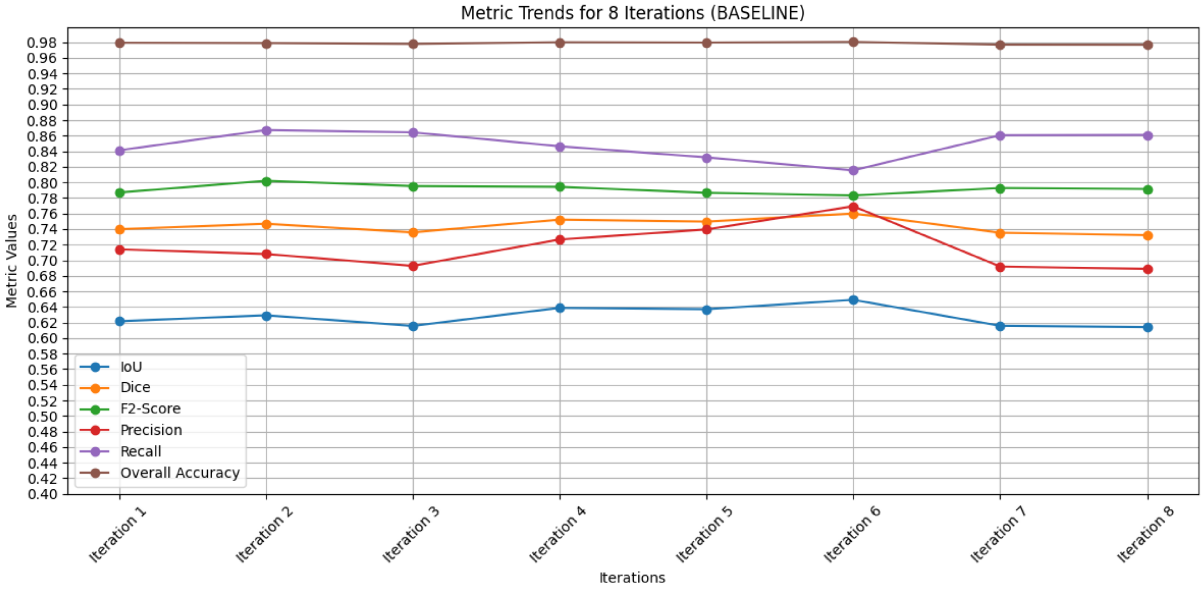


Figure 5.5.1

Table 5.5.1: Summary of BASELINE model's metrics.

Metric	Initial Value	Final Value	Average Value
IoU	**0.621608**	0.614180	0.617894
Dice	**0.739918**	0.732251	0.736084
F2-Score	0.787054	**0.791596**	0.789325
Precision	**0.713988**	0.688870	0.701429
Recall	0.841075	**0.860933**	0.853004
Overall Accuracy	**0.979315**	0.976722	0.978018

Next, Graph 5.5.2 shows the trend of the SPECTRAL model's metrics. Instead, Table 5.5.2 shows a summary of the performance of the metrics concerning the SPECTRAL model. As we can see, overlap metrics such as IoU, and Dice, improve only a little during iterative learning. However, the F2-Score at the eighth iteration decreases by 2% compared to the first iteration. This indicates that the model may have started to identify more false positives than true positives toward the end of training. Overall accuracy represents the model's overall percentage of correct predictions and a slight increase is noted. It is possible to see that, in the fifth iteration, there is a decrease in performance that affects all metrics, except for Recall. This behaviour could be due to variations in the dataset.

In summary, the SPECTRAL method is shown to improve the overlap between model predictions and ground truth throughout iterative training, increasing accuracy but decreasing Recall slightly toward the end.

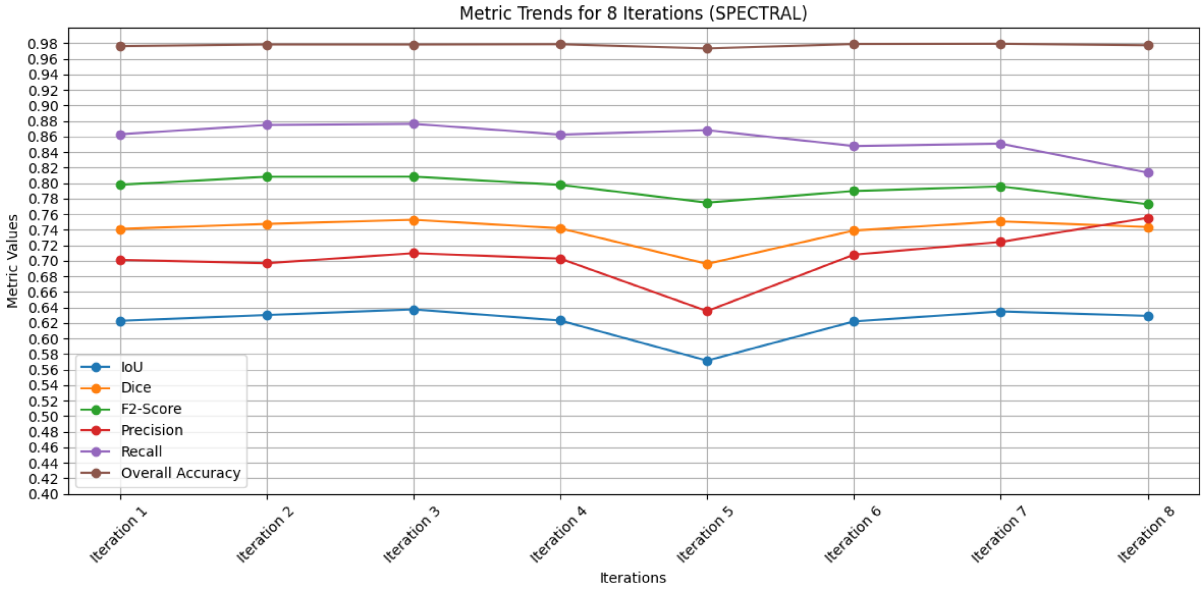


Figure 5.5.2

Table 5.5.2: Summary of SPECTRAL model's metrics.

Metric	Initial Value	Final Value	Average Value
IoU	0.622839	**0.629200**	0.625818
Dice	0.741085	**0.743589**	0.742337
F2-Score	**0.797949**	0.772832	0.785391
Precision	0.701238	**0.755416**	0.728327
Recall	**0.862838**	0.813646	0.838242
Overall Accuracy	0.976349	**0.977425**	0.976883

Graph 5.5.3 shows the trend of the SPECTRAL_SUPERPIXELS model's metrics. Instead, Table 5.5.3 shows a summary of the performance of the metrics respectively to the SPECTRAL_SUPERPIXELS model. As we can see, the initial IoU and Dice scores are relatively low, indicating poor segmentation quality at the beginning of training. However, both metrics improve throughout training, with higher final values. The F2-Score also shows improvement from the initial value to the final value, indicating that the model improved at balancing Precision and Recall as training progresses. Overall accuracy starts with a relatively high value and remains constant throughout training, indicating that the model maintains a high level of Overall accuracy.

In summary, the SPECTRAL_SUPERPIXELS method demonstrates a gradual improvement in segmentation quality, especially in the last iterations, achieving comparable or even better performance than the BASELINE approach.

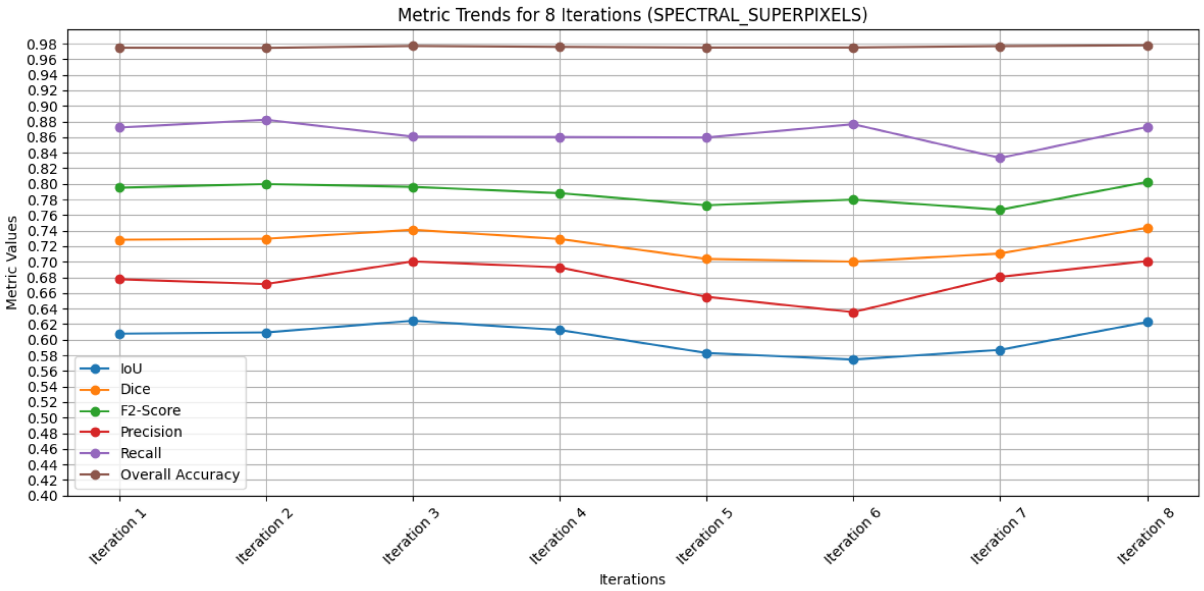


Figure 5.5.3

Table 5.5.3: Summary of SPECTRAL_SUPERPIXELS model's metrics.

Metric	Initial Value	Final Value	Average Value
IoU	0.607796	**0.622640**	0.611699
Dice	0.728318	**0.743620**	0.735990
F2-Score	0.795135	**0.802456**	0.798109
Precision	0.677546	**0.700883**	0.688606
Recall	0.872280	**0.873003**	0.872642
Overall Accuracy	0.974704	**0.977738**	0.975905

Graph 5.5.4 shows the trend of the SPECTRAL_ITTI model's metrics. Table 5.5.4 shows a summary of the performance of the metrics respectively to the SPECTRAL_ITTI model. The initial and final values of the various metrics are fairly close for most metrics, indicating that the model does not change significantly during iterative learning. As we can see, the initial IoU and Dice scores are relatively low, indicating poor segmentation quality at the beginning of training. The F2-Score also shows a similar trend, with a slight improvement at the end of training. Overall accuracy is relatively high at both the beginning and end of training.

In summary, the SPECTRAL_ITTI method shows relatively stable performance during training, with modest improvements in segmentation accuracy.

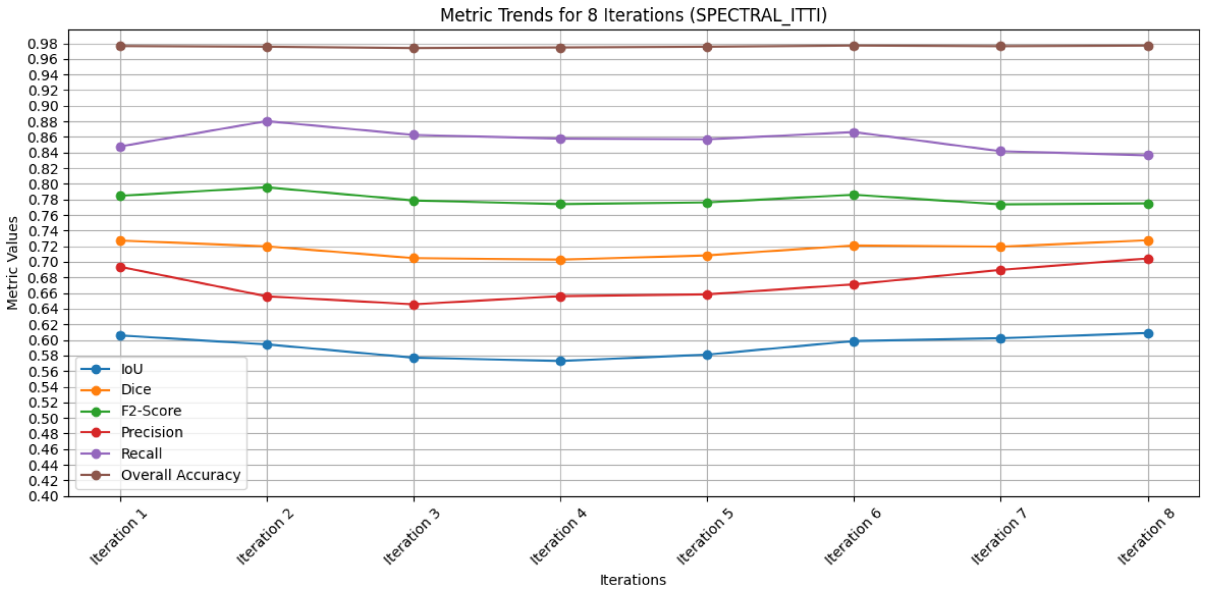


Figure 5.5.4

Table 5.5.4: Summary of SPECTRAL_ITTI model's metrics.

Metric	Initial Value	Final Value	Average Value
IoU	0.605766	**0.608854**	0.606690
Dice	0.727273	**0.727625**	0.727449
F2-Score	**0.784558**	0.774823	0.779690
Precision	0.693765	**0.704323**	0.699044
Recall	**0.847759**	0.836467	0.842113
Overall Accuracy	0.976447	**0.976975**	0.976711

Graph 5.5.5 shows the trend of the GRADv1 model's metrics. Table 5.5.5 shows a summary of the performance of the metrics respectively to the GRADv1 model. As we can see, this method undergoes slight variations in the metrics, resulting in a few individual improvements in the Precision and Overall Accuracy metrics.

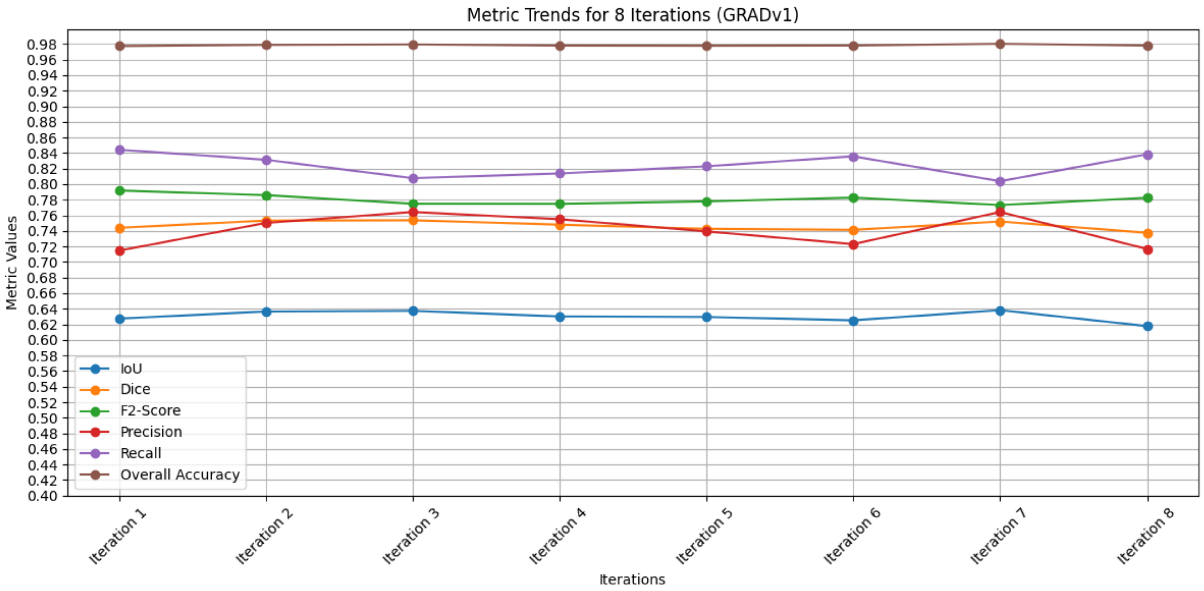


Figure 5.5.5

Table 5.5.5: Summary of GRADv1 model's metrics.

Metric	Initial Value	Final Value	Average Value
IoU	**0.627390**	0.617706	0.623152
Dice	**0.743969**	0.737625	0.740797
F2-Score	**0.791876**	0.782535	0.787206
Precision	0.714716	**0.716855**	0.715785
Recall	**0.844174**	0.838336	0.841255
Overall Accuracy	0.977355	**0.977965**	0.977660

Graph 5.5.6 shows the trend of the GRADv2 model's metrics. Table 5.5.6 shows a summary of the performance of the metrics respectively to the GRADv2 model. As we can see, the IoU values start from about 0.600 at the first iteration to 0.616 at the last, ranging from 0.576 to 0.643; while Dice values start from 0.723 to reach 0.739, ranging from 0.702 to 0.757. Since these are overlapping metrics, we can say that the initial segmentation is not precise, but improves as the re-training iterations continue. The F2 score starts with 0.766 and shows slight fluctuations, but generally improves, ending with about 0.770. The overall accuracy of the model starts with a value of 0.976 and remains consistently high for all iterations, ending with a value of about 0.978. This indicates that the model has a good ability to identify the classes of interest and correctly classify the input image pixels.

In summary, the GRADv2 method shows iterative improvements in segmentation performance, making it an effective approach for the assigned task.

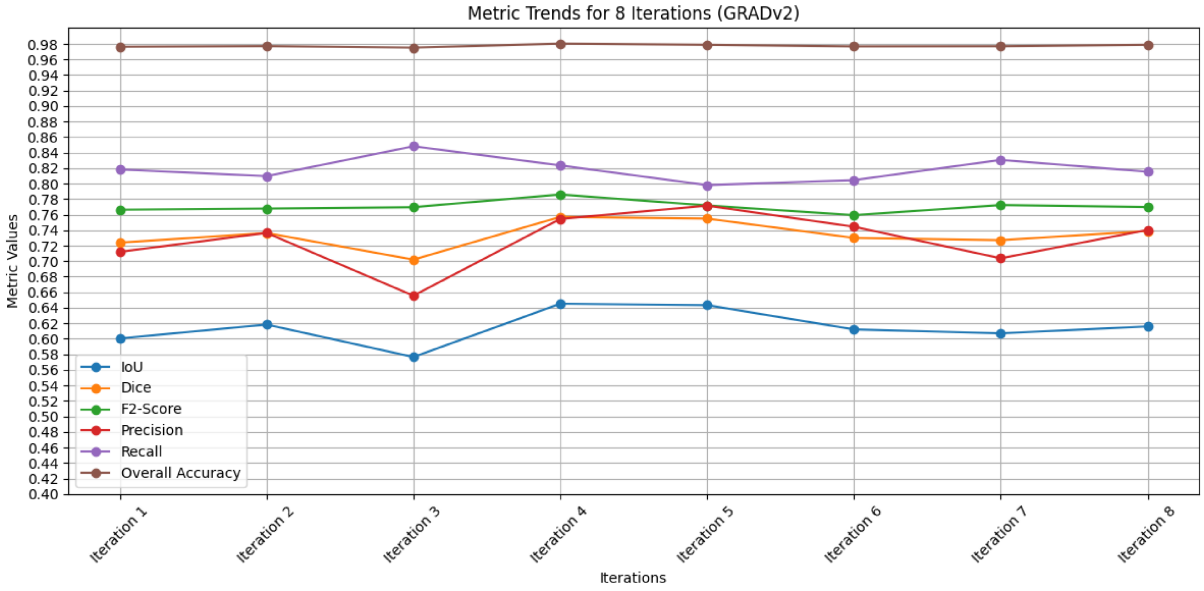


Figure 5.5.6

Table 5.5.6: Summary of GRADv2 model's metrics.

Metric	Initial Value	Final Value	Average Value
IoU	0.600463	**0.616015**	0.609100
Dice	0.723934	**0.739020**	0.731477
F2-Score	0.766402	**0.769815**	0.768108
Precision	0.712023	**0.740330**	0.726677
Recall	**0.818348**	0.815428	0.816777
Overall Accuracy	0.976417	**0.978923**	0.977534

Graph 5.5.7 shows the trend of the GRAD_ARG model's metrics. Table 5.5.7 shows a summary of the performance of the metrics respectively to the GRAD_ARG model. As we can see, the IoU values range from 0.623 to 0.650, starting at 0.623 and reaching 0.632 at the last iteration. The Dice score starts at 0.734 and goes up to 0.748, ranging in values between 0.734 and 0.764. Overall of the 8 re-trainings, we can say that the average values obtained concerning these overlap metrics are the best, given the fact of having a constant improvement in each iteration. The F2-score range from 0.733 to 0.781, while the initial value is 0.753 and the final value is 0.764. Here we can see a wider fluctuation than in the other methods, given by changes in the Precision and Recall metrics. The overall accuracy remains consistently high. This indicates that the model maintains a high level of correct predictions throughout the training process.

In conclusion, the GRAD_ARG method shows a positive trend of improvement in segmentation performance metrics.

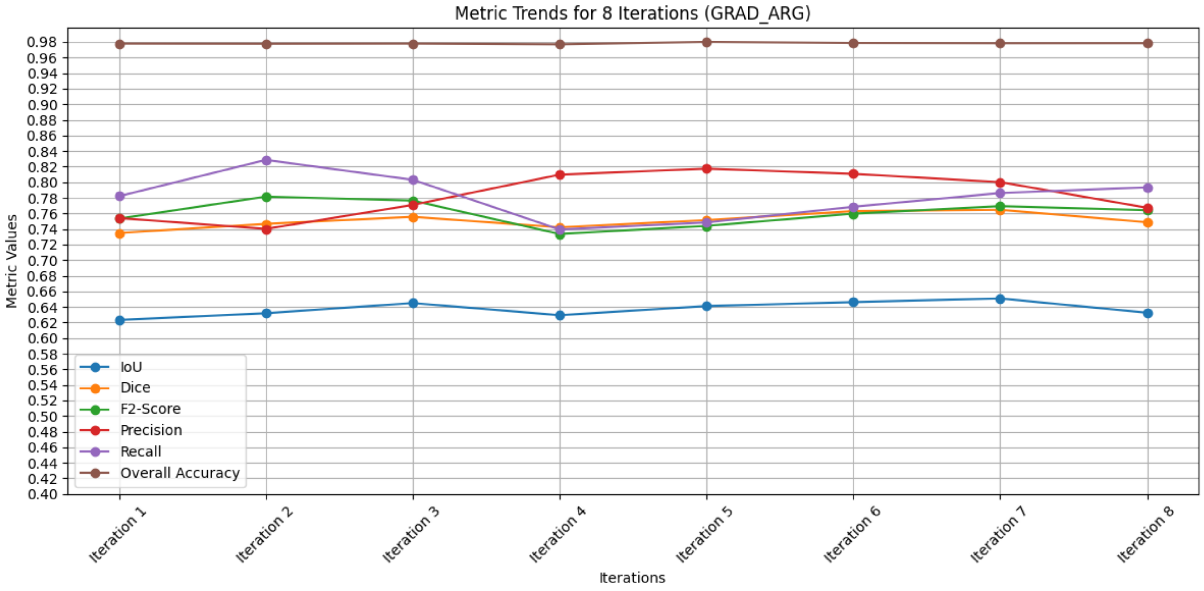


Figure 5.5.7

Table 5.5.7: Summary of GRAD_ARG model's metrics.

Metric	Initial Value	Final Value	Average Value
IoU	0.623321	**0.632517**	0.629762
Dice	0.734847	**0.748677**	0.741762
F2-Score	0.753288	**0.764094**	0.758691
Precision	0.754054	**0.767179**	0.760616
Recall	0.781986	**0.793370**	0.787678
Overall Accuracy	0.977870	**0.978382**	0.978126

5.5.2 Results Discussion

We will now discuss the results presented in the previous Sub-Sections. Initially, we are going to compare methods using Saliency Maps with methods using Grad-CAMs and see how the performance indicators change. Then, we will compare each model with the BASELINE approach, trying to see if there are any promising methods among those developed. We will use the average value of each metric for each method developed so that we can also define stability, convergence and if there has been an overall improvement. Figure 5.5.8 can be useful to have a visual understanding of the performance of each metric respectively the methods developed.

First, let's start by comparing methods that use the Saliency Map approach. For the results shown in Tables 5.5.2, 5.5.3 and 5.5.4 we can note that the SPECTRAL_SUPERPIXELS model seems to have generally high performance among the three methods in the last iteration. The Dice score, Recall and F2-Score metrics are among the highest, indicating a good ability of the model to capture relevant features in the data. Overall accuracy is constant, and in the last 3 iterations, it increases a lot, overtaking the other two models. Instead, the SPECTRAL model has better results in terms of IoU and Precision. On the other hand, the SPECTRAL_ITTI model shows significant variation in metrics between runs, with some performance that is significantly better than others. For example, the Recall seems to reach a high value in the fourth iteration and stay stable until the end. Overall accuracy is generally high, but there may be some sensitivity to specific data features or parameter configurations.

It is important to note that the spectral algorithms used to calculate saliency maps highlight more the parts of the input image that contain an abrupt change in colour or brightness. This is typical of colorectal images, which may have light reflections in the colon wall due to the probe's light used to perform the colonoscopy and illuminate the affected area. This may be the reason for no excellent performances, because these areas may not have polyps.

After that, we analyze the GRADv1 and GRADv2 models. From the results shown in Tables 5.5.5 and 5.5.6, we can note that, in terms of Precision and Recall, GRADv2 has slightly higher values than GRADv1. However, GRADv1 has a higher F2-Score than GRADv2, indicating a better ability to balance Precision and Recall. Overall accuracy is similar between the two methods, with GRADv2 slightly higher. In summary, the two methods seem to have similar performance, with GRADv2 showing slight superiority. However, these differences may not be statistically significant to determine a model's behaviour.

Certainly, we can say that the slight improvement we had with GRADv2 comes from the use of more useful information in generating the activation map, given by the layers taken at the initial, intermediate and deep levels of our base model, as explained in Section 4.2.2.

Table 5.5.1 presents a summary of the performances concerning the BASELINE model. We use this data as a basis for a brief comparison with metrics from the other methods developed:

- **SPECTRAL**: In Table 5.5.2 is presented a summary of the performances concerning the SPECTRAL model. The results are similar to those of the BASELINE, with higher Overall accuracy. IoU and Dice are also good, indicating a good overlap between predictions and labels. F2-Score is not so good: although SPECTRAL is better in terms of Precision, it is not better in terms of Recall, so this weighs a lot on the F2-Score computation.

- **SPECTRAL_SUPERPIXELS:** In Table 5.5.3 is presented a summary of the performances concerning the SPECTRAL_SUPERPIXELS model. This model seems to perform slightly better concerning BASELINE, in general; in fact, we have small improvements in each metric.

- **SPECTRAL_ITTI:** In Table 5.5.4 is presented a summary of the performances for the SPECTRAL_ITTI model. This model seems to perform slightly worse concerning BASELINE, in general; in fact, we have small improvements only in terms of Precision and Overall accuracy. IoU, Dice, F2-score and Recall are still good, but worse respectively to BASELINE.

- **GRADv1, GRADv2**: In Table 5.5.5 and 5.5.6 are presented a summary of the performances for the GRADv1 and GRADv2 model. Concerning the GRADv1 model, the results are similar to those of BASELINE, with higher Overall accuracy and Precision. IoU and Dice are also good. Only F2-Score and Recall perform worse. Instead, GRADv2 has better performance in terms of Iou, Dice, Precision and Overall accuracy, but it maintains the same trend as 'GRADv1' in terms of F2-Score and Recall.

- **GRAD_ARG**: In Table 5.5.7 is presented a summary of the performances for the GRAD_ARG model. As we can see from the graphs 5.5.8, the IoU and Dice scores are better than any other method, indicating a good overlap between predictions and labels. This model has better performances also in Precision and Overall accuracy for BASELINE, but it seems to have difficulty improving Recall without

compromising Precision, and this behaviour spoils also the F2-Score.

We can say that most methods can maintain or improve performance compared to BASELINE in some of the metrics evaluated. All methods seem to produce fairly good results in terms of IoU, Dice, and Overall Accuracy. Recall and Precision metrics vary among the methods, with some methods scoring higher in these metrics than others. Analyzing other metrics than Overall accuracy, we can see that GRAD_ARG gets higher scores than all the other methods on IoU and Dice metrics, respectively, which are important as overlap metrics as well as Precision. The other method that overtakes the others is SPECTRAL_SUPERPIXELS on F2-Score and Recall metrics, respectively, which is important for balancing false negatives against false positives in the binary segmentation mask. Finally, the highest Overall accuracy is given by the GRADv2 method, which is closest to 0.98 at the last iteration of iterative learning.

There are differences between the results obtained with the different methods and this could be due to various factors, including the specific approach developed, weight management, optimizer, maximum number of epochs and other training parameters. The results of some methods, like SPECTRAL, GRADv2 and GRAD_ARG appear to be not so stable across iterations. For these methods, it might be useful to increase the number of re-training to achieve some sort of convergence.

In summary, iterative learning with different interpretation techniques can lead to significant improvements in model performance, but the choice of methods must be guided by the specific nature of the problem, as segmentation of colonoscopy images is a very complicated task by its nature.

To summarize, using various explainable techniques in iterative learning can enhance model performance. However, it's important to select the appropriate methods based on the complexity of the problem at hand, and its nature: the segmentation of colonoscopy images is a very complicated task by its nature.

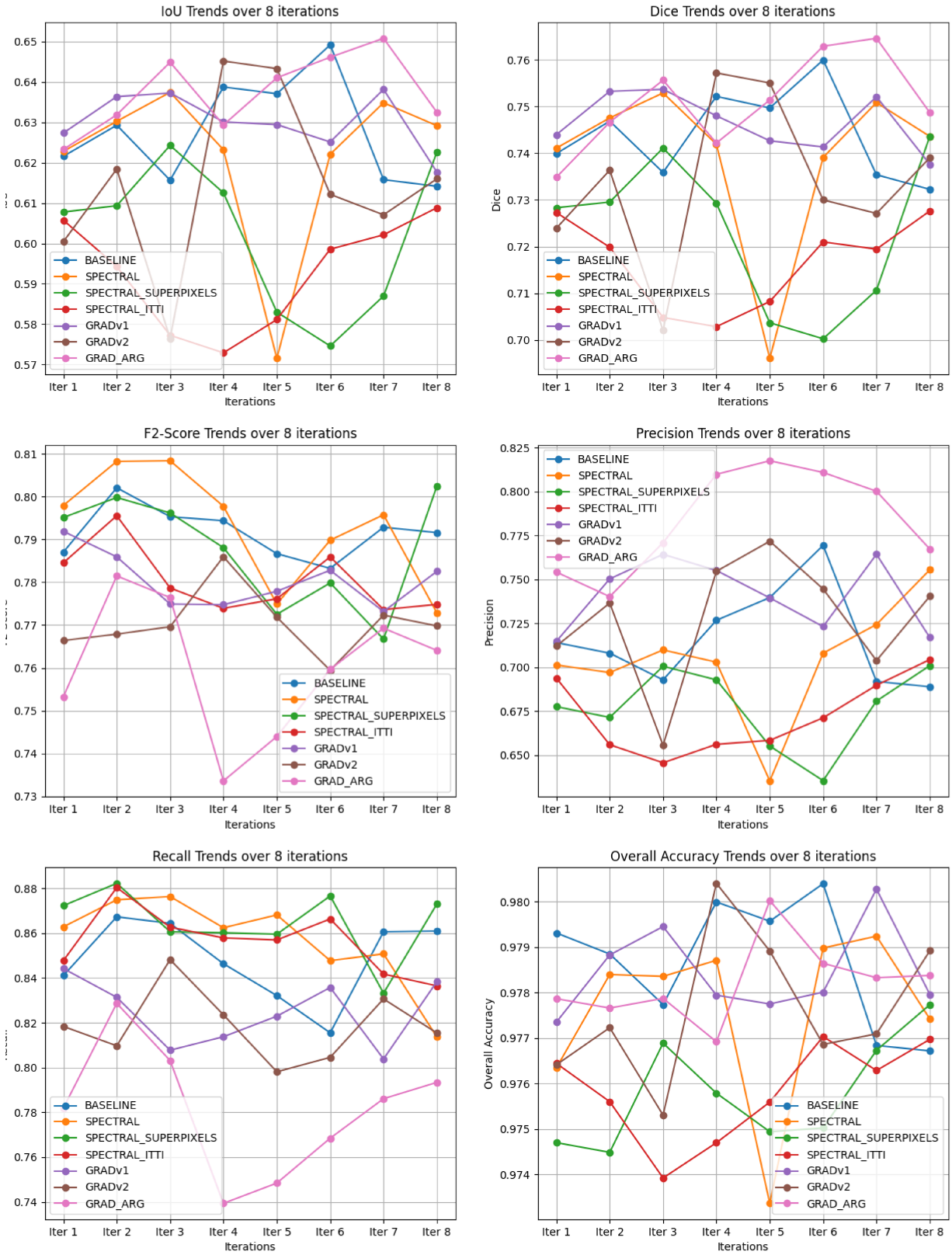


Figure 5.5.8: Trend of metrics in the 8 iterations.

Chapter 6

Conclusions

The conclusions of this thesis are now drawn, having come to the end of the presentation and discussion of the results. In this thesis, we conducted an in-depth investigation in the field of medical image segmentation, a crucial element in the field of medical image processing, particularly when used in the context of colon polyps and colorectal cancer, which ranks among the most prevalent forms of cancer nowadays. Precise segmentation of anatomical structures in medical images plays a key role in guiding clinical interventions, treatment planning and patient assessment.

This thesis is focused on the examination of two of the most important explainable AI-based methodologies that aim to improve image segmentation in our specific domain, exploiting weak supervised semantic segmentation. To address this complex challenge, we developed several methods, including SPECTRAL, SPECTRAL_SUPERPIXELS, SPECTRAL_ITTI, GRADv1, GRADv2 and GRAD_ARG. These methods exploited both saliency maps and Grad-CAM maps, attempting to capture relevant image features and salient image regions to improve segmentation accuracy. Through a series of experiments, we evaluated the performance of each method in terms of IoU, Dice score, F2-Score, Precision, Recall and Overall accuracy.

The results obtained revealed several interesting trends. For example, methods based on Grad-CAM, such as GRADv2 or GRAD_ARG showed great potential in improving segmentation, with high performance on several metrics. On the other hand, Saliency Map-based methods, such as SPECTRAL_SUPERPIXELS, showed a slight advantage in terms of F2-Score and Recall. However, it is important to note that the differences between these methods may not be statistically significant in determining model behaviour, due to variability in metrics during re-training iterations. The robustness of each method

depends on its ability to maintain consistent performance on different images. The results indicate that no method is consistently superior to BASELINE in all metrics. This represents a lack of robustness in the results.

6.1 Future Works

One of the possible future paths could focus on exploring other methods of Explainable AI, such as Occlusion Sensitivity, LIME, etc., or focus on approaches involving a fusion of saliency map and Grad-CAM-based methods. An example would be to try to combine what good has been achieved with SPECTRAL_SUPERPIXELS and GRAD_ARG and observe the results, noting whether there is an improvement or not.

In addition, considering the complex nature of colonoscopy images, another possible way could be to explore even more deeply the scientific literature in this context and observe new methods that can take advantage of the weak labels.

In conclusion, this thesis sought to bring improvements to already well-tested and high-performing methodologies in the field of medical image segmentation of colorectal polyps by offering explainable AI-based approaches. The work remains open to possible interesting developments.

Bibliography

[1] Rebecca L. Siegel et al. "Colorectal cancer statistics, 2020". In: *CA: A Cancer Journal for Clinicians* (2020). URL: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21601.

[2] Prof. Orazio Schillaci. "I numeri del cancro in Italia, 2022". In: *salute.gov.it* (2022). URL: https://www.aiom.it/wp-content/uploads/2022/12/2022_AIOM_NDC-web.pdf.

[3] Juan José Granados-Romero et al. "Colorectal cancer: a review". In: *International Journal of Research in Medical Sciences* (2017). URL: https://www.msjonline.org/index.php/ijrms/article/view/3905.

[4] P. Wieszczy, J. Regula, and M.F. Kaminski. "Adenoma detection rate and risk of colorectal cancer". In: *Best Practice & Research Clinical Gastroenterology* (2017). URL: https://www.sciencedirect.com/science/article/pii/S1521691817300707.

[5] Alexander V. Mamonov et al. "Automated Polyp Detection in Colon Capsule Endoscopy". In: *IEEE Transactions on Medical Imaging* (2014). URL: https://doi.org/10.1109%2Ftmi.2014.2314959.

[6] Nima Tajbakhsh, Suryakanth R. Gurudu, and Jianming Liang. "Automated Polyp Detection in Colonoscopy Videos Using Shape and Context Information". In: *IEEE Transactions on Medical Imaging* (2016).

[7] J. Bernal, J. Sánchez, and F. Vilariño. "Towards automatic polyp detection with a polyp appearance model". In: *Pattern Recognition* (2012). URL: https://www.sciencedirect.com/science/article/pii/S0031320312001185.

[8] Gregor Urban Siwei Chen and Pierre Baldi. "Weakly Supervised Polyp Segmentation in Colonoscopy Images using Deep Neural Networks". In: (2022). URL: https://www.igb.uci.edu/colonoscopy-ai-for-gi2/.

[9] Justin Ker et al. "Deep Learning Applications in Medical Image Analysis". In: *IEEE Access* (2017). DOI: 10.1109/ACCESS.2017.2788044.

[10] Nikolaj Buhl. “Guide to Image Segmentation in Computer Vision: Best Practices”. In: *encord.com* (2023). URL: https://encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide/.

[11] Nikolaj Buhl. “Medical Image Segmentation: A Complete Guide”. In: *encord.com* (2023). URL: https://encord.com/blog/medical-image-segmentation/#:~:text=Medical%20image%20segmentation%20is%20used,to%20improve%20accuracy%20and%20outputs..

[12] Tim Miller. *Explanation in Artificial Intelligence: Insights from the Social Sciences*. 2018. arXiv: 1706.07269 [cs.AI].

[13] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. arXiv: 1702.08608 [stat.ML].

[14] Christoph Molnar. *A Guide for Making Black Box Models Explainable*. 2023.

[15] MathWorks. *What Is Interpretability?* 2023. URL: https://it.mathworks.com/discovery/interpretability.html.

[16] akshaysingh98088 Taemin. “What is Saliency Map?” In: *GeeksForGeeks* (2021). URL: https://www.geeksforgeeks.org/what-is-saliency-map/.

[17] Xiaodi Hou and Liqing Zhang. “Saliency Detection: A Spectral Residual Approach”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383267.

[18] L. Itti, C. Koch, and E. Niebur. “A model of saliency-based visual attention for rapid scene analysis”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.11 (1998), pp. 1254–1259. DOI: 10.1109/34.730558.

[19] Boris Schauerte. “Spectral Visual Saliency Toolbox”. In: 2023. URL: https://www.mathworks.com/matlabcentral/fileexchange/32455-spectral-visual-saliency-toolbox.

[20] Boris Schauerte and Rainer Stiefelhagen. “Predicting human gaze using quaternion DCT image signature saliency and face detection”. In: *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*. 2012, pp. 137–144. DOI: 10.1109/WACV.2012.6163035.

[21] Boris Schauerte and Rainer Stiefelhagen. “Quaternion-Based Spectral Saliency Detection for Eye Fixation Prediction”. In: Oct. 2012, pp. 116–129. ISBN: 978-3-642-33708-6. DOI: 10.1007/978-3-642-33709-3_9.

[22] Chenlei Guo, Qi Ma, and Liming Zhang. "Spatio-temporal Saliency detection using phase spectrum of quaternion fourier transform". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587715.

[23] Antonino Furnari, Giovanni Maria Farinella, and Sebastiano Battiato. "An Experimental Analysis of Saliency Detection with Respect to Three Saliency Levels". In: *Computer Vision - ECCV 2014 Workshops*. Ed. by Lourdes Agapito, Michael M. Bronstein, and Carsten Rother. Springer International Publishing, 2015.

[24] Mariusthart. "A view of the fort of Marburg (Germany) and the Saliency Map of the image using color, intensity and orientation. Color had a weight of 1.5 and the other maps had a weight of 1." In: *Wikimedia Commons* (2008). URL: https://upload.wikimedia.org/wikipedia/commons/1/1a/Saliencymap_example.jpg.

[25] Anne M. Treisman and Garry Gelade. "A feature-integration theory of attention". In: *Cognitive Psychology* 12.1 (1980), pp. 97–136. ISSN: 0010-0285. DOI: https://doi.org/10.1016/0010-0285(80)90005-5. URL: https://www.sciencedirect.com/science/article/pii/0010028580900055.

[26] H. Greenspan et al. "Overcomplete steerable pyramid filters and rotation invariance". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Publ by IEEE, 1994. DOI: 10.1109/cvpr.1994.323833.

[27] Bolei Zhou et al. *Learning Deep Features for Discriminative Localization*. 2015. arXiv: 1512.04150 [cs.CV].

[28] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. 2014. arXiv: 1312.4400 [cs.NE].

[29] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. DOI: 10.1007/s11263-019-01228-7. URL: https://doi.org/10.1007%2Fs11263-019-01228-7.

[30] MathWorks. *Grad-CAM Reveals the Why Behind Deep Learning Decisions*. 2023. URL: https://it.mathworks.com/help/deeplearning/ref/gradcam.html?searchHighlight=Grad-CAM&s_tid=srchtitle_support_results_1_Grad-CAM.

[31] Xavier Alphonse Inbaraj et al. “Object Identification and Localization Using Grad-CAM++ with Mask Regional Convolution Neural Network”. In: *Electronics* (2021). DOI: 10.3390/electronics10131541. URL: https://www.mdpi.com/2079-9292/10/13/1541.

[32] Tom Vercauteren Carole H. Sudre Wenqi Li. “Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations.” In: *Deep Learn Med Image Anal Multimodal Learn Clin Decis Support* (2017). DOI: 10.1007/978-3-319-67558-9_28.

[33] MathWorks. *GeneralizedDice Function - MATLAB Documentation*. 2023. URL: https://it.mathworks.com/help/vision/ref/generalizeddice.html.

[34] Xavier Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Journal of Machine Learning Research - Proceedings Track* (2010).

[35] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[36] Liang-Chieh Chen et al. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. 2018. arXiv: 1802.02611 [cs.CV].

[37] Liang-Chieh Chen et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017. arXiv: 1706.05587 [cs.CV].

[38] Liang-Chieh Chen et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. arXiv: 1606.00915 [cs.CV].

[39] Liang-Chieh Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *ECCV*. 2018.

[40] Chiereghin Francesco. “SEGMENTAZIONE DI POLIPI DEBOLMENTE SUPERVISIONATA SU IMMAGINI DI COLONSCOPIE.” In: *Padua Thesis and Dissertation Archive* (2022). URL: https://thesis.unipd.it/retrieve/40cfd29a-6405-4a38-8a32-e02d4d4f4ffb/Chiereghin_Francesco.pdf.

[41] Khoshgoftaar T.M. Shorten C. “A survey on Image Data Augmentation for Deep Learning.” In: *J Big Data 6, 60* (2019). DOI: https://doi.org/10.1186/s40537-019-0197-0.

[42] Jorge Bernal et al. “WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians”. In: *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society* (2015). DOI: 10.1016/j.compmedimag.2015.02.007. URL: https://doi.org/10.1016/j.compmedimag.2015.02.007.

[43] Radhakrishna Achanta et al. “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/TPAMI.2012.120.

[44] Nobuyuki Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: 10.1109/TSMC.1979.4310076.

[45] Yuhan Xie et al. “Detect, Grow, Seg: A weakly supervision method for medical image segmentation based on bounding box”. In: *Biomedical Signal Processing and Control* (2023). DOI: https://doi.org/10.1016/j.bspc.2023.105158. URL: https://www.sciencedirect.com/science/article/pii/S1746809423005918.

[46] Debesh Jha et al. “Kvasir-seg: A segmented polyp dataset”. In: *International Conference on Multimedia Modeling*. Springer. 2020, pp. 451–462.

[47] D. et al Vázquez. “A benchmark for endoluminal scene segmentation of colonoscopy images.” In: *J. Healthc. Eng.* (2017). DOI: https://doi.org/10.1155/2017/4037190.

[48] Silva J.S.; Histace A.; Romain O.; Dray X.; Granado B. “Toward embedded detection of polyps in WCE images for early diagnosis of colorectal cancer.” In: *Int. J. Comput. Assist. Radiol. Surg.* (2014). DOI: https://doi.org/10.1007/s11548-013-0926-3.

[49] Gregor Urban Siwei Chen and Pierre Baldi. “Weakly Supervised Polyp Segmentation in Colonoscopy Images using Deep Neural Networks”. In: 2022. URL: https://www.igb.uci.edu/colonoscopy-ai-for-gi2/.