POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Politecnico
di Torino

Master's Degree Thesis

Tandem: segmentation of small medical objects exploiting classification

Supervisors

Prof. Daniele APILETTI

Dott. Simone MONACO

Candidate

Guido GENCO

MARCH 2024

Abstract

Deep Learning Medical Image Segmentation is a popular computer vision task, its goal is to provide a precise and accurate representation of target objects with the purpose of disease diagnosis or treatment planning. In this thesis we apply Deep Learning Image Segmentation methods to detect cysts on physiological images of kidneys tissues affected by ADPKD. The collected dataset is characterized by images depicting several sparse and tiny cysts with different sizes and shapes in order to improve segmentation results already computed in previous work over it. Moreover, given images and cysts characteristics we will focus our attention over deep learning methods developed to well perform also with images depicting really small objects. Different solutions will be explored and finally a proposed method consisting of a segmentation model and a classifier trained together, called Tandem method, will be presented and tested. Classification head role inside Tandem solution can be described as a post processing segmentation refinement specifically suitable for small and sparse target objects. Classifier head is aware of segmentation model predictions and consequently refine them by erasing the classified-as-wrong segmented object, in order to adjust and improve segmentation output. Computed tests over our dataset show that this method achieve great results, outperforming standard segmentation models such as U-Net, for example.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter the global context of the thesis is presented, specifically ADPKD topic is introduced to the reader, than a brief description of present work contribution is reported. In chapter 2 background and several related works are described to provide the reader a general vision of the most popular methods and technique used in medical image segmentation. In chapters 3 and 4 respectively selected methods will be deeply analyzed and computed results will be discussed.

1.1 ADPKD

Autosomal dominant polycystic kidney disease (ADPKD) is a life-threatening inherited human disorder and the most common hereditary kidney disease. It's a progressive disease characterized by the formation of multiple cysts that grow out of the renal tubules, kidney enlargement, and also extrarenal organ involvement[1]. It affects up to 12 million individuals and is the 4th most common cause for renal replacement therapy worldwide[2]. The majority of ADPKD patients (approx. 80–85 %) carry a germline mutation in the PKD1 gene on chromosome 16p13, whereas about 15–20 % harbor a mutation in the PKD2 gene on chromosome 4q21[3]. There is an urgent need for developing patient-specific models that can replicate key aspects of polycystic kidneys and therefore be used for drug testing studies. Recently, two new kind of drugs have been used to reduce the growth rate of cysts: Tolvaptan and Octreotide-LAR. In [4], using 3D printing technologies and patients'

cells, a platform has been developed in order to "*study the pathogenesis of the disease directly in human tissues and to identify novel therapeutic targets*". This system is based on engineered kidney tubules and allows evaluation of ADPKD response to drugs, which indeed can be formulated in terms of cysts number and dimension. However, the lack of automated systems to quantify cysts in kidney tubules did not currently allow the platform to be further developed and used at a large scale. In order to properly investigate if these and other future drugs can be effective in ADPKD treatment, the main goal of this thesis is introduced: the development of an artificial intelligence system to perform a fully-automatic segmentation of cysts on kidney tubules to improve and automatize the detection and quantification of cyst number and size. Specifically, given analysis already computed in [4] with U-Net based segmentation models tested on a dataset consisting of images depicting kidney tissue affected by ADPKD cysts, we want to make a step further using the same set of data to evaluate effectiveness of deep learning segmentation models explicitly developed to deal with small medical target objects. The idea is to localize attention over smaller cysts, which represent the main obstacle for obtaining fully satisfying segmentation performances in this task.

1.2 Contributions

The main purpose of this thesis, as anticipated in previous section, is to investigate new deep learning solutions to improve image segmentation of small and scattered medical target objects, which in our case are ADPKD typical kidney cysts. Specifically, all selected methods and techniques are evaluated with a dataset provided by *Istituto di ricerche farmacologiche Mario Negri* (Bergamo, Italy) consisting of RGB immunofluorescence images of tridimensional human tubules engineered from epithelial cyst-lining cells that were isolated from a single donor patient with a mutation in PKD1. Actually, we want to expand and improve results already computed over this dataset in previous work [4] from Monaco et al. Moreover, considering peculiar dataset characteristics that will be described extensively in following sections, our idea is to explore and test effectiveness of deep learning segmentation solutions specifically designed to well perform with small medical objects, rather than more generic and popular segmentation models such as U-Net[5]

and similar. Although those architectures obtain usually good overall segmentation results, they are less sensitive at detecting small objects which in some fields are quite frequent, especially in medical one. Indeed, the final goal of this thesis is the development of a method able to identify correctly the highest number of cysts possible, simultaneously trying to keep the amount of false negatives and false positives as lower as possible, aiming at providing reliable predictions to physicians and enable measurement of ADPKD response to drugs in a fast way and, consequently, also on a large scale. For the purpose of moving towards described goal we firstly investigate state-of-art literature, aiming to find segmentation architectures suitable for our needs. As a result of that search we identify two promising models: CE-Net[6] and CaraNet[7]. We test them for the first time with before mentioned dataset in order to compare their performances and also to understand if their declared capabilities are useful in our context. Finally, we present and explain in depth our newly cysts segmentation solution called Tandem, consisting of a classifier working together with a segmentation model. The first acts as a refinement post-segmentation process for the latter, but the main characteristic is that classifier and segmentation model are not used in sequence only at inference time but instead they are trained together, using as training loss a sum of the two models losses. We asses Tandem performances with cysts dataset following a cross validation strategy called Leave-One-Tubule-Out (LOTO) introduced in [4], therefore we compare Tandem also with several models already evaluated in Monaco et al. work with the same strategy. LOTO results show how good our proposed method is since it reaches great positioning in different evaluation metrics among all considered models in ADPKD cysts segmentation task.

Chapter 2

Background

In this chapter Image Segmentation topic is introduced, specifically its role in medical field is discussed. Moreover, also typical challenges of medical image segmentation are described. Finally, notable deep learning architectures and techniques are presented.

2.1 Image Segmentation

Image segmentation is the process of partitioning digital images in groups of pixels based on some criteria. The division is performed assigning a label to each pixel inside the image in order to build a so called "segmentation mask" which makes possible to highlight some areas of the original image that share a specific characteristic, in other words is the process of dividing an image in non-overlapping regions (segments). The criteria according to which the separation of an image in different portion is performed are task dependent [8], in fact image segmentation has a central role in a broad range of digital image processing tasks since it can be exploited in a plethora of applications belongings to several fields like image processing, image analysis, image understanding, and pattern recognition. The key feature of image segmentation is the possibility of underlining some crucial and interesting aspects inside an image, which alone is just a scene representation, while computing a segmentation mask global information are extracted from it and specific properties of the scene are highlighted throughout a description of the scene

in terms of simple elements like shapes, regions and boundaries. This feature can be crucial in various fields, and this is the reason why image segmentation has been implemented in so many ways and received so much attention during recent years, especially from the moment it is carried out by deep learning methods. Moreover, since image segmentation is used in different applications there is no standard way of segment a specific image, segmentation result goodness is quantified by the reason why a specific image have been segmented. Thus the segmentation problem has not a unique result [8]. Different methods can be more or less effective in segmenting a specific image, depending on the type of image considered, for example some methods can be more efficient on real images while others on synthetic ones. Additionally, even with the same kind of images various methods can be effective in different measures considering the final goal of segmentation, some can segment better specific parts of objects like boundaries rather than other more precise at segmenting shapes. The two main types of segmentation are *semantic segmentation* and *instance segmentation*: In the first case, final goal is classify each single pixel belonging to an image into a fixed set of categories without differentiating object instances, while in the latter case the final goal is to detect all objects in an image while also precisely segmenting each instance [9]. Mentioned differences are visually presented in figure 2.1. Numerous image segmentation algorithms and methods

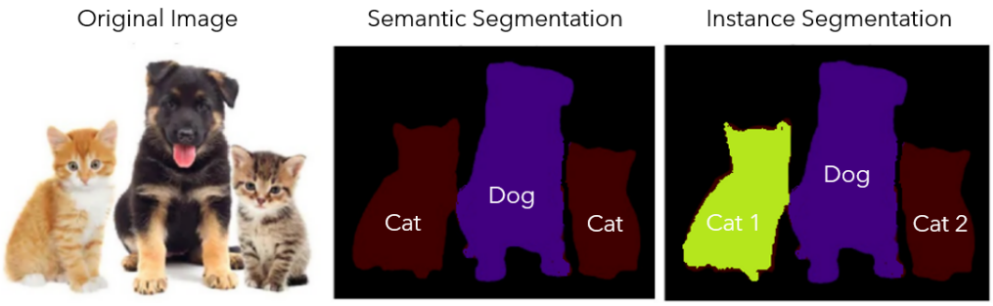


Figure 2.1: Semantic and instance segmentation comparison, from `https://www.v7labs.com/blog/instance-segmentation-guide`

have been developed, from earliest ones such as thresholding and k-means clustering to more complex and recent like Markov random fields. More recently also deep learning models have been exploited to perform image segmentation, nowadays they are the standard approach.

2.2 Image Segmentation in medical domain

Image segmentation, as mentioned before, is a great tool for highlighting in a fast way interesting and specific parts of an image. This feature make it ideal for several medical field applications in which noisy and complex images are scanned from doctors in order to detect and locate specific objects or regions of interest. The utilization of automated segmentation methods makes possible to provide an accurate representation of target objects and analyze their characteristics for diagnosis purpose in an automated, and consequently less time consuming, manner. Examples of these applications, just to name a few, are:

- retinal vessel segmentation
- nuclei segmentation in microscopy images
- identification of organs and lesions from CT or MRI images
- liver segmentation in abdominal CT scans
- early detection and diagnosis of breast, lung, glaucoma, and skin cancer

Image segmentation has a crucial role in medical fields because its usage can help doctors, making the diagnosis process faster and more efficient.

2.3 Image Segmentation in medical field

In the 2000s deep learning approaches started to demonstrate their considerable capabilities in image processing tasks [10], including image segmentation. Deep learning methods achieved results never seen before with previous segmentation methods and algorithms, it is correct to declare that deep learning has been a game changer in image segmentation providing optimal results and great performances on popular benchmarks [11].Probably the benchmark that allowed deep learning to shine for the first time has been ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[12] in 2012 when one of the first and most famous deep learning model AlexNet[13] won the challenge, showing the superiority of deep learning methods over other solutions. Owing the great capabilities of deep learning methods, they

have been exploited also in medical image segmentation. Before diving into deep learning image segmentation methods, is crucial to have an overview of what kind of models have been involved in medical image segmentation and what challenges characterize this specific field.

2.3.1 Data availability issues

Despite the huge attention paid to deep learning based methods for medical image segmentation in recent years, publicly available datasets suitable for training segmentation models are limited both in number and size. This scarcity is probably the main problem that modern image segmentation methods have to face in medical field. Acquiring enough data to properly train a segmentation model for medical field is a not trivial operation for several reasons: medical images are private and sensible patients data, this means that sharing them, but even just collecting them, can be a challenging task because of obvious privacy and legal related aspects. In fact, a lot of in literature presented methods are trained with private datasets which for these reasons cannot be shared publicly. Moreover, collecting medical data suitable for deep learning image segmentation it's a relatively recent concern, meaning that the few publicly available datasets are usually small if compared in terms of images number with the ones publicly available in other fields. Another issue consists of the high cost associated with both data collection and annotation: medical images are usually collected from expensive machines, also annotating this kind of images require efforts and time from experienced doctors. Last but not least, is really hard to obtain balanced datasets for rare diseases because positive cases are very infrequent rather than negative ones. All this issues lead to small, and most of the times inadequate, public datasets. It's a fact that deep learning models need great amount of data to effectively learn and, in general, bigger datasets result in better deep learning models but unfortunately most medical datasets are composed of just few thousands or even hundreds of medical images [14].

2.3.2 CNNs: Convolutional Neural Networks

Convolutional Neural Networks are the most widely used architectures in deep learning, specifically in computer vision tasks. The first architecture defined as

CNN was presented by LeCun et al. in [15]. A CNN has three basic building blocks:

- convolutional layers
- pooling layers
- fully-connected layers

These layers are stacked and each intermediate layer receives as input the output of the previous one [16]. The first layer, called input layer, is directly connected to the input image and has a number of neuron equal to the number of pixels in the input. The next set of layers are convolutional layers, they operate over the input performing convolution with a set of filters, known as kernels, in order to perform feature extraction. The output of a convolutional layer, called activation map, become input of activation layer which apply non-linearity in order to enable the modeling of non-linear functions by the network. After that, usually a pooling layer is positioned, these kind of layers perform pooling operations which help to reduce dimensionality of the convolution's output. There are different pooling options, the most common are max and average pooling. Finally, high level abstractions are extracted by fully connected layers. The weights of neural connections and the kernels are continuously optimized during the procedure of back propagation in the training phase [10].

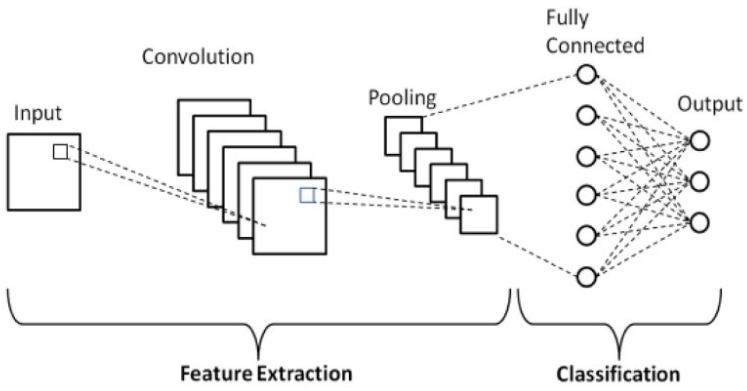


Figure 2.2: CNN diagram from [17]

2.3.3 FCNs: Fully Convolutional Networks

Fully Convolutional Networks, as the name suggests, are a version of CNN in which only convolutional layers are involved and just convolution operations are performed beyond subsampling and upsampling. Fully connected layers are substituted with convolutional ones, this allows to have a dense pixel-wise prediction and to achieve better localization performances [10]. Moreover, this makes possible also to take an image of arbitrary size and produce a segmentation map of the same size in output instead of a simple classification score. A FCN has also less parameters, which means that training can be less time-consuming. Long et al. introduced one of the first FCN architectures in [18], it is considered a milestone for deep learning image segmentation because authors demonstrated that deep networks can be trained for semantic segmentation in an end-to-end manner on variable sized images, achieving a new state-of-the-art segmentation performance [11] testing their work over famous benchmark datasets. They modified some already well known CNN models as VGG16 and GoogleNet substituting fully connected layers with convolutional ones and introduced skip connections in which feature maps from the final layers of the model are up-sampled and fused with feature maps of earlier layers, through this connections the model is able to combine semantic with appearance information, computing more accurate segmentation masks. FCN are not perfect though, their major limitations are related to:

- real-time usage; FCNs are faster than CNNs, but not enough to use them in real-time applications where computation time is crucial.
- global context information; FCNs do not exploit this kind of information efficiently.

Despite these limitations, since when presented achieved best segmentation performances, they have been exploited in a variety of segmentation tasks including medical field related ones such as segmentation of brain tumor or skin lesions.

2.3.4 Encoder-Decoder Based Models

The encoder-decoder structure is a popular kind of architecture in image segmentation. It is based on 2 main macro components: the encoder, a contraction part that

extract information and compress the input image in a reduced dimension feature vector, and the decoder which is the symmetric corresponding part of the network that gradually upscales the encoded features back to the original image dimension [19]. One of the first architecture of this family has been SegNet[20], where Badrinarayanan et al. introduced a convolutional encoder-decoder architecture for image segmentation in which the decoder upsamples its lower resolution feature maps using pooling indices computed from the encoder in its max-pooling steps. Usually the two parts of these networks combine information at different levels exploiting skip connections, a concept firstly introduced in [21], to combine high level spatial features extracted in the encoder with fine-grained ones computed during decoder steps to improve segmentation results.

U-Net

The most worth mentioning deep learning segmentation model based on encoder-decoder structure is probably U-Net[5], which is also the most popular segmentation model specifically designed for medical applications. The primary purpose of this architecture was to address the problem of limited annotated data in medical field, in fact authors stress out the importance of augmentation techniques role to obtain good results with relatively small datasets, as the usual case of medical ones. The model has a characteristic shape known as U-shape, see figure 2.3, consisting of a contracting path (encoder) and an expansive path (decoder). The first has encoder layers in it and is capable of identify relevant features of input image. The encoder layers perform convolutional operations in order to reduce the spatial resolution of the feature maps and at the same time increasing their depth, meaning that will capture increasingly abstract representations of the input image. Just like the contracting path has encoder layers in it, the expansive path is based on decoder layers, its role is to decode encoded and compressed features computed from previous architecture section, upsampling them while also performing convolutional operations. Skip connections make possible to preserve, and consequently recover when needed, spatial information lost during downsampling. These direct paths that connect encoder and decoder have a central role in U-Net behavior because help the latter to better locate features accurately, combining larger and smaller convolutional results to extract features at various

resolutions. Without skip connections would be really hard upsampling back the compressed representation computed from the encoder to the input image resolution. Moreover, skip connections in general help stabilizing training and model convergence. Since U-Net performed so well in medical image segmentation tasks, it gained popularity among developers and researchers, consequently it was deployed also in different segmentation tasks, achieving also there great results. Another consequence of U-Net popularity has been the development of modified versions built with the goal of making the already good architecture presented by Ronneberger et al. even better. An example of this tendency is UNet++ [22] which basically is a U-Net architecture that use a series of nested and dense skip pathways to connect the encoder and decoder sub-networks instead of simple skip connections used in the original U-shaped model. These re-designed skip pathways aim at reducing the semantic gap between the feature maps of the encoder and decoder sub-networks, making easier for the model to learn, authors declare that their modified version of U-Net achieve better performances than the original in several medical image segmentation tasks. Another approach designed to improve the already good

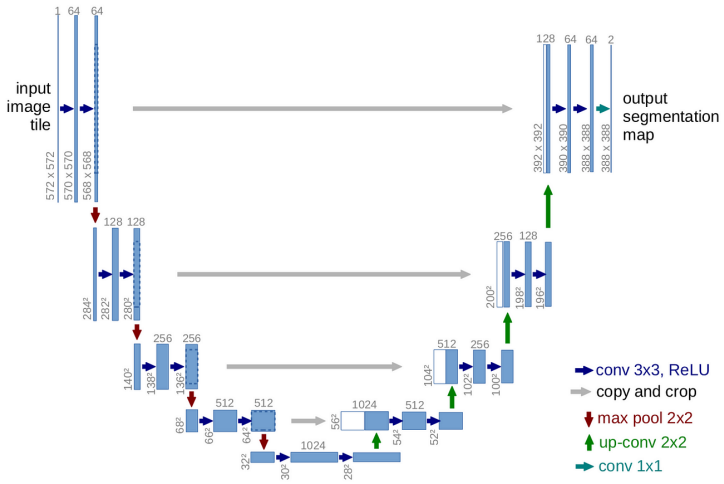


Figure 2.3: U-Net architecture, image from [5]

U-Net segmentation performances is based on adding blocks to the architecture, specifically between encoder and decoder paths, or also changing entire parts of it like the encoder or decoder branch. The goal of the mentioned blocks is usually preserving as much information as possible from features extracted with encoder to

better exploit them in the corresponding decoder. An example of this approach is CE-Net[6]. CE stands for *context encoder*, the main contribution to the standard U-shaped architecture of this specific model is the addition of a context extractor module positioned between encoder and decoder, this module extracts context semantic information and generates more high-level feature maps. Authors report in the paper that this module has been developed to get more abstract features and preserve more spatial information to specifically boost performances of medical image segmentation. Another variation to the basic U-Net architecture that they made is replacing the standard U-Net backbone (encoder) with a ResNet-34 block pretrained on ImageNet. This choice is justified by ResNet shortcut mechanism to avoid the gradient vanishing and accelerate the network convergence i.e. encoder only skip connections in addition to the ones between it and the decoder. Reported results effectively show that this modified version of U-Net brings improvements in different medical image segmentation tasks respect to the original version. Moreover, U-Net has also inspired V-Net[23], which performs segmentation over 3D images with a volumetric, fully convolutional, neural network having a structure inspired by U-Net encoder-decoder characteristic shape. V-Net was introduced to deal with MRI volumes depicting prostate, aiming to translate U-shape from 2D images to 3D volumes. Another popular segmentation methods in medical field is HarDNet-MSEG[24] which uses an encoder-decoder structure but only high-level features extracted in the encoder path are used in the corresponding decoder to compute the final output, for this reason it is called partial-decoder. Specifically, those features are elaborated by a RFB block (Recepive Field Block) during decoding phase which uses multi-branch with different kernel size convolution and also dilated convolution to extract features with different receptive fields. Finally, a 1x1 convolution is applied to merge extracted features from each RFB block and after a dense aggregation of RFBs outputs the final result is computed. This structure of encoder-decoder models with partial decoders and blocks that extract features exploiting dilated convolution and kernels with different sizes to compute multi-scale features is really popular in image segmentation. In fact another relevant model in medical image segmentation is PraNet[25] which also uses a partial decoder but additionally applies also reverse attention to encoder extracted features in order to grant segmentation improvements extracting relationships between boundaries and

areas.

2.3.5 Other relevant segmentation approaches

Other solutions designed to improve medical image segmentation are not based on new architectures but instead on different approaches to the problem, an example is post segmentation refinement where the goal is to use deep learning models in sequence to improve segmentation results exploiting prior knowledge of specific medical tasks. The idea of using models in sequence is quite popular because segmentation results quality is really task specific, and exploiting contribution from different models can be beneficial. The idea of just trying to improve segmentation results only throughout architectural changes can reach diminishing returns sometimes, meaning that popular segmentation models perform already well at segmenting objects and increasing their parameter number or changing structural aspects is not that easy, moreover sometimes those changes lead to really small improvements, making them almost not profitable. On the other hand, acting directly over segmentation results seems to pay off more in some cases. PostDAE[26] is an example of this approach, it is a solution based on post segmentation refinement criteria: using a Denoising Autoencoders (DAE) as a post processing step this model-independent method aims to improve masks computed with arbitrary segmentation models, see figure 2.4 to visualize solution pipeline. At inference time the DAE receives in input masks computed from a generic segmentation model and outputs an anatomical plausible version of it. Since DAEs are neural networks specifically designed to reconstruct a clean input from a corrupted version of it, in this solution the DAE is trained with segmentation mask pairs consisting of an artificially degraded mask and a correct version of the same mask. With this training technique the autoencoder learns target objects usual shape in segmentation masks and also their usual position in it. Consequently, the DAE learns a relation between arbitrary masks and anatomical plausible ones and will be able to impose anatomical priors over generic segmentation masks. This method though, given the idea over which is based, is suitable for medical tasks where target objects occupy usually the same position inside the image and also have a recognizable and common shape such as lung segmentation for example. Another case of segmentation improvement obtained exploiting models

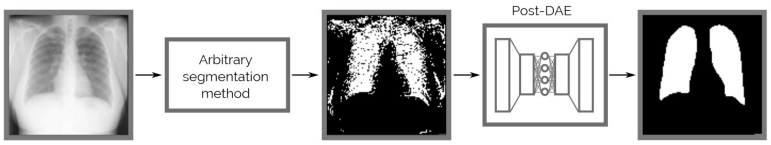


Figure 2.4: PostDAE pipeline. From [26]

in sequence is Stack U-Net[27] where authors build a refinement chain using up to 15 blocks in sequence. The basic building block can be both U-Net standard model or also a modified version of U-Net, called ResU-Net which exploits residual connections. Each model receives in input previous block output combined through skip connections with the original input image in order to refine segmentation result progressively going further in the models chain. Finally the last example

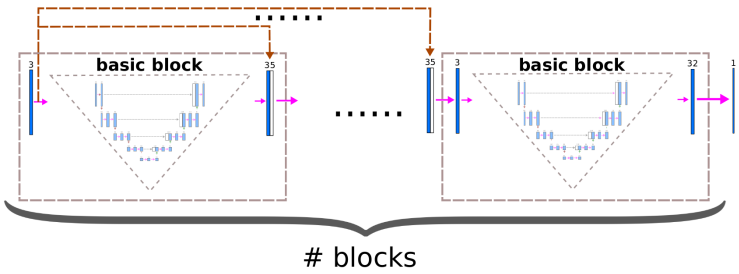


Figure 2.5: Stack-U-Net[27]

of segmentation improvement exploiting several models combined is given by [28], where segmentation models are used together in two different ways, producing consequently two models called respectively TriUNet and DivergentNets. The first is a combination of three U-Net models in which the input image is passed to the first two, having different randomized weights. The output of both models is then concatenated before being passed through a third U-Net model to predict the final segmentation mask. DivergentNets is instead an ensemble of different well performing and popular segmentation models, and also the before introduced TriUNet is used in it. The idea is to improve segmentation results in generic medical tasks exploiting predictions performed by different models separately trained before and finally merged in this ensemble solution at inference time, the idea is to get a precise and generic segmentation architecture able to well-perform in several medical tasks.

Each input image is passed to every model and the final segmentation mask is computed simply averaging computed predictions. Other newly medical image

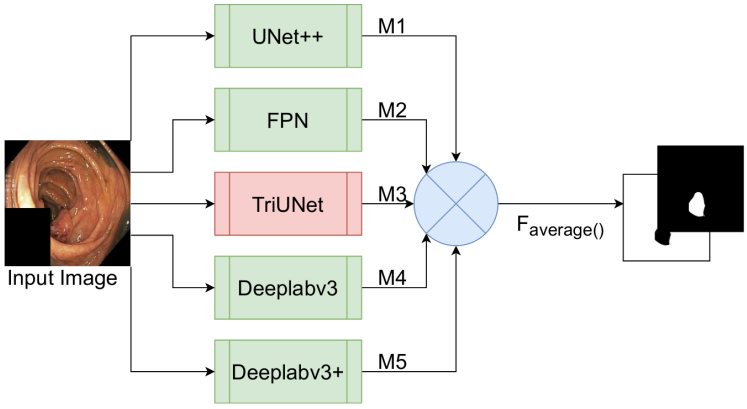


Figure 2.6: DivergentNets structure[28]

segmentation solutions are based on GANs: Generative Adversarial Networks, in which a deep learning model called generator learns a specific task trying to fool another deep learning model, called discriminator, that has to detect if received data is original or has been generated from the generator. GANs are quite useful in image segmentation tasks where available datasets are usually small due to the possibility of generating artificial images offered by this kind of models, therefore can be exploited in segmentation medical related tasks [29]. Rather than using GANs just to generate artificial data to extend training datasets, other solutions based on these kind models use their peculiar adversarial training modality to teach models a relation between RGB images and relative segmentation masks in order to produce a segmentation model to all effects. This approach has been used in medical segmentation tasks [29]. Unfortunately, GANs need really long training time to properly learn if compared with more traditional deep learning methods.

Chapter 3

Methods

In this chapter the utilized dataset is described in depth, specifically task specific segmentation target objects, i.e. cysts, will be analyzed to make the reader aware of the difficulties that a segmentation model can face when dealing with images depicting them. In addition, a brief overview of selected and tested models is reported, followed by a description of Tandem, our newly proposed method, we also explain the motivation that led us to implement this solution. Finally, evaluation metrics and assessment criteria used to analyze results and performances of considered methods are introduced.

3.1 Dataset

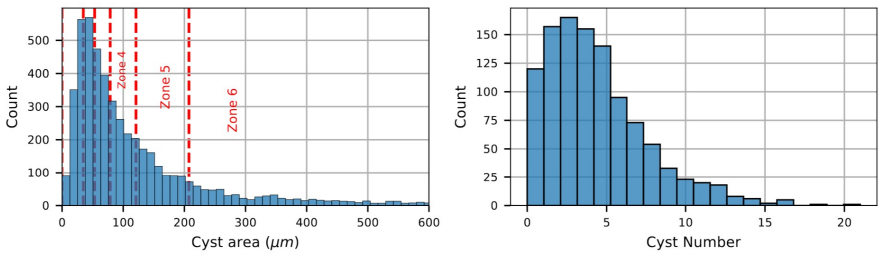


Figure 3.1: Dataset statistics. Statistics for cyst size (left side) and number of cysts per image (right side) on the whole dataset. From [4]

The dataset we used to conduct our experiments was the same used in [4], it has

been provided by Istituto di ricerche farmacologiche Mario Negri (Bergamo, Italy) and consists of RGB immunofluorescence images of tridimensional human tubules engineered from epithelial cyst-lining cells that were isolated from a single donor patient with a mutation in PKD1. Tubules are the result of 4 individual experiments conducted from July 2019 to December 2020 and are classified according to the treatment received, specifically there are images belongings to 4 different *in vitro* experiments with 32 engineered polycystic kidney tubules. The dataset has 1076 images of microscope acquisitions with a fixed scale and fixed size of 1024×1024 pixels. The total number of annotated cysts is 5042. For each image, cysts were manually annotated by human experts through a polygonal segmentation with the opensource annotation tool labelme [30]. The number of cysts depicted in each image is not fixed, a single image can contain a variable number of cysts that goes from zero to twenty in some cases. Moreover, also cysts size is variable: even though they have on average a very small area, their dimensions fluctuate in an ample interval. In [4] dataset annotated cysts were aggregated in 6 zones according to their size (expressed in μm^2), intervals that define these size zones are shown in table 3.1. Analysis computed in Monaco et al. previous work with this dataset

Cysts size	Size zone
$x < 32\mu m^2$	1
$32\mu m^2 \leq x < 53\mu m^2$	2
$53\mu m^2 \leq x < 83\mu m^2$	3
$83\mu m^2 \leq x < 131\mu m^2$	4
$131\mu m^2 \leq x < 232\mu m^2$	5
$x \geq 232\mu m^2$	6

Table 3.1: Cysts size zones, x is the cyst size expressed in μm^2 .

highlight the difficulty of U-Net based segmentation models at computing satisfying results when dealing with cysts belonging to size zone 1, or in other words the smallest ones. Cysts features constitutes the dataset greatest challenge due to the fact that, although cysts share similar visual characteristics, they can have various sizes and shapes which contribute to make this task more complex to learn for a segmentation model. Moreover, segmenting small objects is already a difficult task, and indeed deep learning segmentation models are usually deployed in tasks

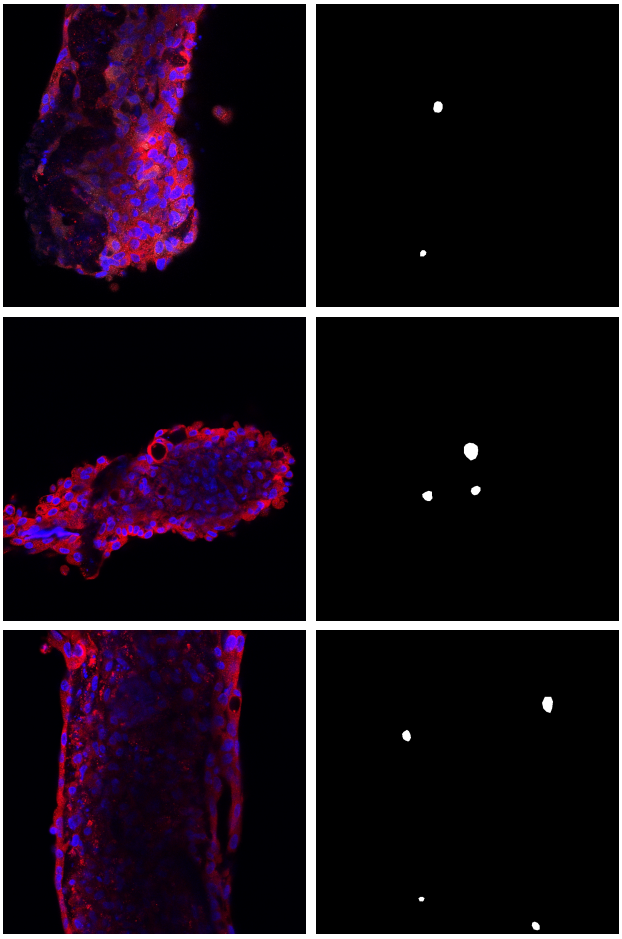


Figure 3.2: Dataset images and relative segmentation masks examples

where target objects have fixed shape and dimensions, net of minor variations, and occupy a not too small part of the image. In this dataset though, average cysts dimension is really small, therefore segmentation target objects occupy only a minimum part of images surface. Another consequence of cysts dimension is that segmentation masks are "unbalanced", meaning that foreground portion occupy always a minimum part of the mask, making the training process harder, see figure 3.2. Another challenge is constituted by the visual similarity between cysts and the tissue pattern, meaning that in addition to the reduced average size of target objects, also a lot of tissue portion have visual features in common with cysts, hence this can easily mislead segmentation models. Mentioned similarities are depicted in image 3.3 to better understand the problem. Yet another aspect to

consider as challenge is the scatter distribution of cysts in images. Target objects, in fact, in several pictures have a scattered positioning, moreover they seems to not share a distribution pattern: in some cases they are really far apart from each other and isolated, in others they are close to each other resembling clusters, and finally in some cases they present a mixed modality, meaning that the same picture can have clusters of cysts with also scattered and isolated ones. To summarize the above, detecting and segmenting the exact position of cysts in these particular images is not an easy task, neither for a human, because target cysts have really small dimensions and variable shape, moreover their identification is made even more difficult due to their general resemblance to the background pattern and their not predictable positioning. In general detecting and segmenting tiny objects is a known task in medical image segmentation because the ability of detecting small areas means usually possible early detection of fatal diseases and consequently prevention of serious and irreversible consequences in many cases. Thus, small objects segmentation in medical domain is a challenging task of crucial importance. For this reasons we focused our solution search on deep learning methods that aim

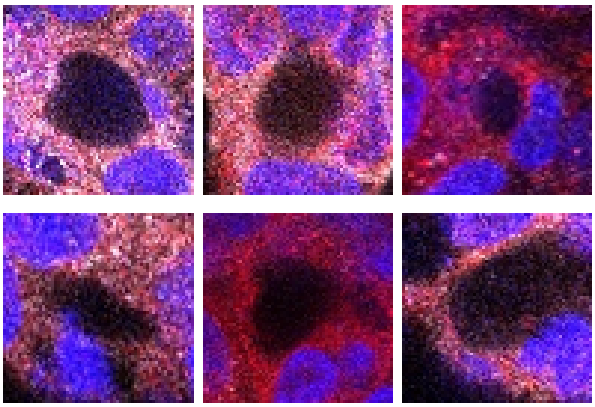


Figure 3.3: Patches extracted from dataset images. True cysts (top row) and portions of tissue resembling cysts (bottom row)

to compute good performances with segmentation of small objects in medical fields, the dataset discussed before is a great benchmark to understand how much a given model is really good at segmenting this kind of difficult objects due to its specific features. Models and methods developed specifically for this kind of tasks are not really frequent due to the challenging goal, so even finding solutions to asses was

quite challenging.

3.2 Preprocessing

As usual in deep learning based solutions, images have been augmented throughout different random transformations before feeding them into models during training in order to gain generalizability. Indeed, augmentations help models to gain invariance properties to image rotations, brightness changes and similar variations, but most of all they are really useful to enlarge the number of samples available in dataset. This concept has already been mentioned in previous sections, specifically when talking about U-Net architecture: Ronnenberg et al. point out in their paper[5] the crucial role of augmentations in deep learning approaches applied to medical fields because of the usual medical datasets small size in terms of samples number. Applied augmentations are reported in table 3.2 with relative parameters and probability rate, the latter is the parameter that, more than others, makes possible showing different transformed versions of the same image to models during training combining different transformations in several ways. Finally, each image has been normalized with ImageNet[12] mean and standard deviation in order to obtain the same ImageNet[12] RGB images distribution, this is also a common practice in computer vision because many pretrained models have ImageNet weights.

Transformation	Parameter	Probability
Horizontal Flip	-	0.5
Random Rotate	$\pm 90°$	0.5
Random Brightness	brightness limit: 0.2	1.0
Random Contrast	contrast limit: 0.2	1.0
Random Gamma Correction	gamma limit: (80, 120)	0.5
Contrast Limited Adaptive Histogram Equalization (CLAHE)	contrast limit: 4.0	0.5
Normalize	ImageNet mean and stdev for each RGB channel	1.0

Table 3.2: Augmentations applied to images

3.3 Selected segmentation models

We decided to focus our attention on segmentation models explicitly developed to deal with small medical objects and for this reason we selected from literature two promising architectures: CE-Net and CaraNet. Both will be detailed described in following sections. Moreover, we compared those solutions with UNet++, a model already tested in [4], therefore we consider it as a "baseline" model because previous work results proved that, when tested with our cysts dataset, is the most precise architecture. Consequently also a brief explanation of this model is reported below.

3.3.1 Unet++

UNet++[22] is a segmentation model based on U-Net, as the name suggests. What distinguish this version from the basic one is the dense skip pathways between encoder and decoder, which replace simple skip connections. It enhances extracted feature processing and was reported by its authors to outperform U-Net on several datasets. Encoder and decoder are connected through a series of

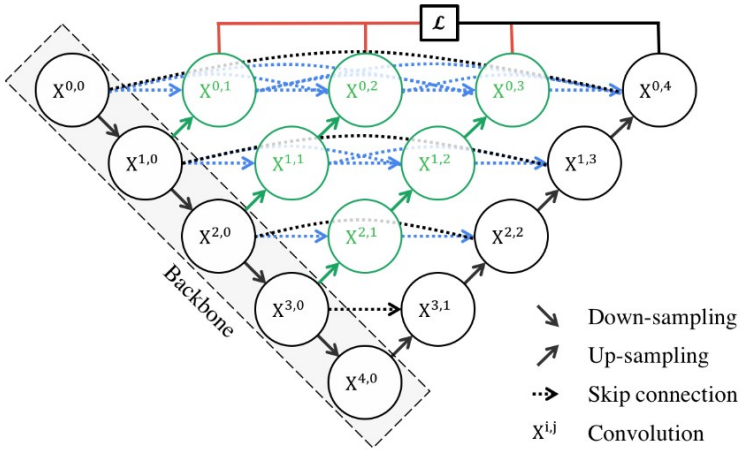


Figure 3.4: Unet++[22] architecture

nested dense convolutional blocks. The main idea is to bridge the semantic gap between the feature maps of the encoder and decoder prior to fusion, making them semantically similar to each other. This is done to make optimization process easier

because of the similarity between received features from encoder and awaiting ones in corresponding decoder depth. The encoder features, before arriving to the

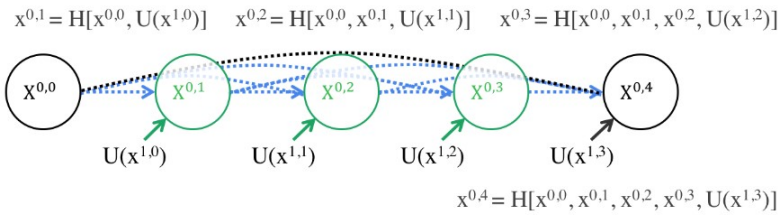


Figure 3.5: Unet++[22] first skip path details

decoder, pass through a dense convolutional block as shown in figure 3.5 where H represents a function that combines Batch Normalization, ReLU activation, and a 3x3 convolution while U an upsampling operation and finally [] concatenation.

3.3.2 CE-Net

CE-Net[6], briefly introduced in section 2, is called Context Encoder Network because the main contribution of this architecture is adding to a classic U-Net encoder-decoder structure a Context Extractor module, beyond changing the encoder with a ResNet-34 pretrained on Imagenet. Authors expose a common U-net limitation: consecutive pooling layers reduce feature resolution to compute increasingly abstract feature representations, sacrificing spatial information. The goal of CE-Net model is to boost segmentation maintaining high-level semantic features at the middle stage (between encoder and decoder) without increasing parameter number and size of feature maps. Motivated by this, authors introduce a Context Extractor block to capture more high-level features and preserve spatial information in an efficient way, figure 3.6 show architecture schema. The Context Extractor module extracts context semantic information and generates more high-level feature maps, convolution operations performed in this module are dilated convolutions (also known as "atrous" convolution), this because even though standard convolution is widely used in segmentation to extract feature representations of images, the pooling layers usually placed after lead to loss of useful semantic information. In order to overcome this limitation, dilated convolution is adopted which basically increase the kernel receptive field dilating it with a rate r without increasing

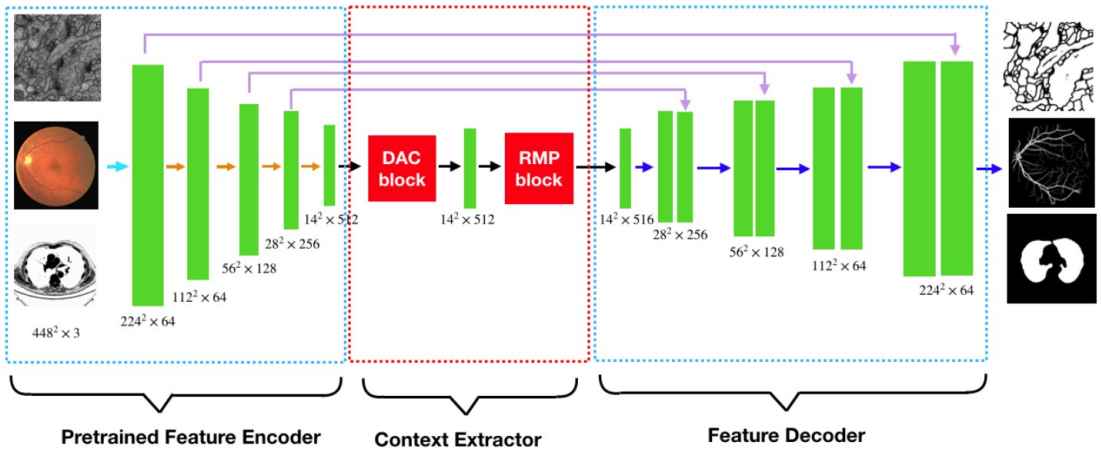


Figure 3.6: CE-Net[6] architecture

parameter number or computational cost. A regular convolution is a specific case of dilated one, to perform it in fact r is set to 1. Given a 3x3 kernel in dilated convolution and $r = 1$, for example, we have a receptive field of 3, while if $r = 3$ the receptive field become 7 without any additional computation cost. The Context Extractor module consists of two main sub-blocks: the DAC block and the RMP block, both are explained in details below.

DAC block

Dense Atrous Convolution (DAC) block is used to encode the high-level semantic features maps, which in other words are the features computed from encoder module. Authors were inspired by Inception[31] and ResNet[21] basic blocks. In the first different convolutions are performed in parallel with kernels having several dimensions in order to compute multi-scale features while in the latter skip connections are exploited to avoid gradient vanishing. A DAC block merge these two techniques, an illustration from [6] is reported below:

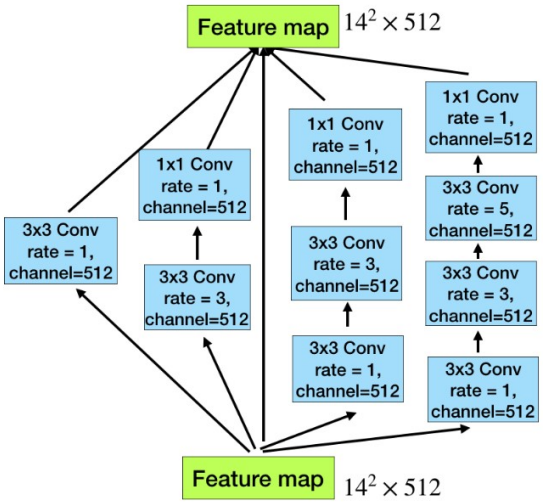


Figure 3.7: DAC module

In DAC block there are 4 branches, each one with a different dilation rate for dilated convolution: from 1 of the first branch up to 1,3 and 5 of the last, thus receptive fields of the four branches are different (3,7,9,19) as in Inception, and finally as in ResNet the original features are added to computed ones via skip connection. At the end of each branch we have a 1x1 convolution for rectified linear activation. The convolution of large reception field extract and generate more abstract features for large objects, while the convolution of small reception field is better for small object. Authors declare that by combining the dilated convolution of different rates, the DAC block is able to extract features for objects with various sizes and consequently also small ones, this is also the main reason why we selected CE-Net for our experiments given cysts variable size.

RMP block

Residual Multi-kernel Pooling is CE-Net authors solution to classic segmentation challenge in medical images: variation of objects sizes. RMP relies on multiple effective field-of-views to detect object at different sizes. A general max pooling operation usually just employs a single pooling kernel. As illustrated in figure 3.8, the RMP block encodes global context information with four different-size receptive fields. The four-level outputs contain the feature maps with various sizes. A 1×1

convolution is performed after each level of pooling to reduce the dimension of weights and computational cost. Then computed feature maps are upsampled to get the same size features as the original feature map via bilinear interpolation. Finally, the original features are concatenated with upsampled feature maps.

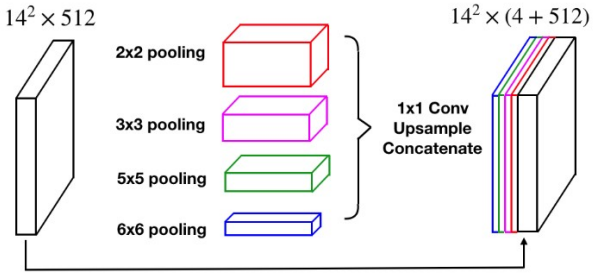


Figure 3.8: RMP module

Dice Loss

CE-Net authors tested their model over dataset with highly imbalanced images, where segmentation target objects occupied only a small area of the mask. In order to overcome the limitations that Binary Cross Entropy has with this kind of imbalanced data they used Dice Loss to train CE-Net. Dice coefficient D is a number among 0 and 1 which describes the overlap between two binary masks, specifically predicted and ground-truth masks. The goal is to maximize that number to have masks really similar to each other. Dice loss is based on Dice coefficient D and was developed to explicitly deal with imbalanced segmentation masks[32][23]. Dice loss formulation from [6] is reported below:

$$\mathcal{L}_{dice} = 1 - \sum_{k}^{K} \frac{2\omega_k \sum_i^N p_{(k,i)} g_{(k,i)}}{\sum_i^N p_{(k,i)}^2 + \sum_i^N g_{(k,i)}^2}$$

where N is the pixel number, $p(k,i) \in [0,1]$ and $g(k,i) \in \{0,1\}$ denote predicted probability and ground truth label for class k, respectively. K is the class number (which is two: background and foreground), and $\sum_k \omega_k = 1$ are the class weights. In CE-Net paper, authors set $\omega_k = \frac{1}{K}$.

3.3.3 CaraNet

CaraNet[7] is a model developed to solve segmentation of small objects problem, specifically in medical fields. Authors motivate their goal with a sharp observation: detecting small objects means finding early stages diseases and consequently can be crucial for early detection and diagnosis. Given network purpose we decided to test it with our dataset since, as already mentioned, it is characterized by small cysts which cause serious difficulties to standard segmentation models. CaraNet uses Res2Net[33] as backbone for features extraction. Given an input RGB image, a backbone network compute increasingly higher representations of it throughout its layers: the deeper is the layer, the higher representation channels number grows, meanwhile its spatial dimension is reduced. In CaraNet only high-level features extracted from deep backbone layers are passed to the decoder, which in this architecture is a Partial Decoder (PD) precisely because only a subset of backbone computed features are utilized.

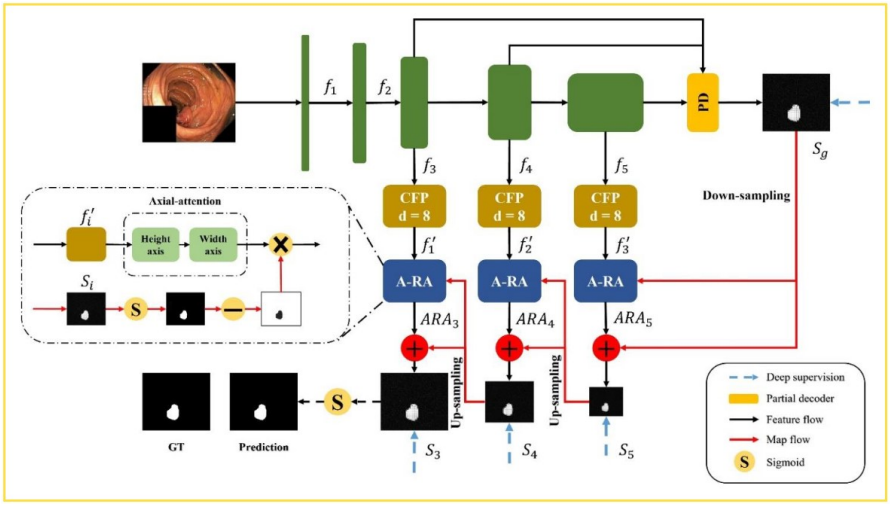


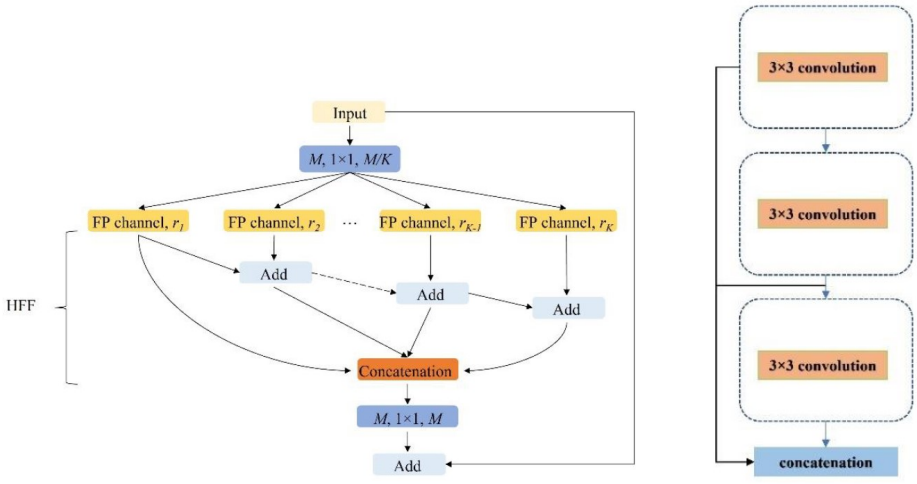
Figure 3.9: CaraNet[7] architecture

Actually, from Res2Net can be obtained features at five different depth levels $f_i, \{i = 1, ...,5\}$, only high-level features are aggregated with the Partial Decoder to get a global map $S_g = \text{PD}(f_3, f_4, f_5)$. Authors motivate the structural choice of completely discard backbone extracted low-level features reporting that these contribute less to performance but have higher computational cost because of their

larger spatial resolution. CaraNet has two main modules: CFP and A-RA.

CFP module

CFP stands for Channel-wise Feature Pyramid, has been introduced in [34] and is a module developed in order to perform complex convolution operations efficiently, therefore its paper goal was to present a segmentation model for real-time usage. Authors were inspired by Inception module and its reformulation presented in [31]. Specifically, in Inception multi-scale feature maps are computed in each module throughout parallel convolutions with different kernel sizes and than concatenated. In the original Inception formulation, three parallel convolutions with 1x1, 3x3 and 5x5 kernels were performed, but computational cost of the last two was high. Consequently, to reduce parameters and make the module more efficient, factorization was introduced in [31]: the expensive 5x5 convolutional filter was replaced with two 3x3 in sequence. A CFP module is a factorized form of convolution operator that decompose large kernel into smaller convolutions consisting of K channels called *Feature Pyramid channels* (FP).



(a) (b)

Figure 3.10: Diagrams of (a): CFP module; (b): FP channel

In a single FP channel multi-scale features are extracted with factorized 5x5 and 7x7 convolutions, replaced respectively with two and three 3x3 filters similarly to what happens in the last Inception module version. Finally, results are concatenated

exploiting skip connections to compute multi-scale features and also to make network trainable, avoiding vanishing gradient. Each FP channel has a specific dilation rate, indeed dilated convolution rather than regular one is performed, meaning that convolutional filter is dilated with a dilation rate r in order to enlarge the receptive field without increasing parameters number. The dilation rate is the number of pixel gaps between adjacent convolution elements and contribute to multi-scale features extraction. In CaraNet each CFP module has $K = 4$ FP channels having dilation rate $r_k = \{1,2,4,8\}$. CFP module input is reduced from dimension M to $M/4$ with a 1x1 convolution, which projects high-dimension feature maps to low-dimension, before being passed to each FP channel. Exploiting skip connections, channels output features are combined in a Hierarchical Feature Fusion (HFF) style:

$$\begin{cases} level_1 = out_{FP1} \\ level_2 = level_1 + out_{FP2} \\ level_3 = level_2 + out_{FP3} \\ level_4 = level_3 + out_{FP4} \end{cases}$$

Final FP contains a stack of features computed as:

$$\sum_i level_i$$

The final output is obtained adding to HFF result, with a skip connection, the original input. Complete CFP architecture is reported in figure **??**.

A-RA module

A-RA is the abbreviation of Axial-Reverse Attention, this module apply attention, in other words assign a weight to each portion of the frame to focus training over specific images areas. As visible in figure 3.9 these modules are placed after each CFP module, this because they receive in input relative CFP output and a global map S_i computed from previous A-RA module, except for the first which receive S_g global map directly from Partial Decoder. Two kind of attention mechanism are applied respectively to both inputs. Axial attention, which is based on self-attention, is the operation of assigning to a single context (image) some weights, specifically a

weight to each portion of it. This is factorized along height and weight dimensions to improve performances. Result of axial-attention is then combined, through an element-wise multiplication, with reverse attention[35] computed over S_i map in order to erase salient regions from side-outputs and consequently force the network to explore portion of the image with potentially missing objects. Combination of the two attention mechanism leads to A-RA output (ARA_i). Finally ARA_i is combined with global map S computed from previous steps obtaining S_5, S_4, S_3 which are the previous mentioned side-outputs. CaraNet final segmentation mask is computed performing a Sigmoid over S_3.

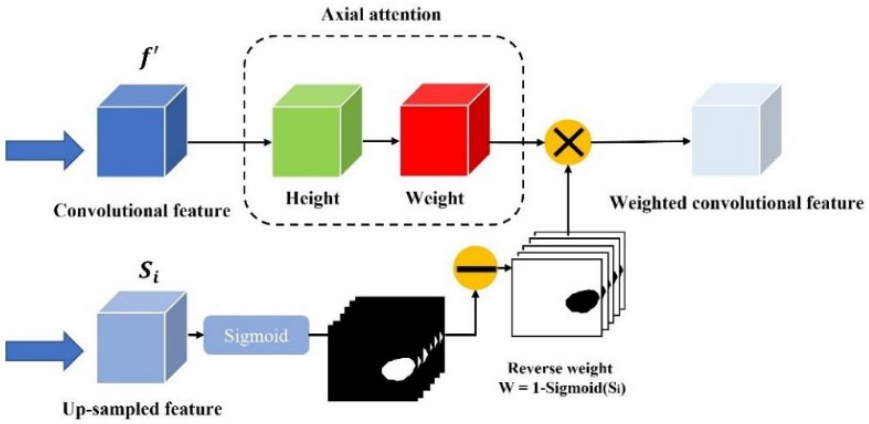


Figure 3.11: Structure in details of A-RA module

Structure Loss

Structure Loss is a loss formulation developed by CaraNet authors to train specifically their model. This loss is a combination of weighted Intersection over Union (IoU) and weighted Binary Cross-Entropy (BCE). The first ensure a loss computation to globally supervise the segmentation results while the latter is used to supervise results at local pixel-level. Consequently to train CaraNet, a deep supervision for the three side-outputs ($S1$, $S2$, $S3$) and the global map Sg is applied. Thus, given ground-truth segmentation mask G, the total loss is:

$$\mathcal{L}_{total} = \mathcal{L}^w_{IoU} + \mathcal{L}^w_{BCE} = \mathcal{L}(G, S^{up}_g) + \sum_{i=3}^{5} \mathcal{L}(G, S^{up}_i) \tag{3.1}$$

We used this loss to train CaraNet during out experiments and also for supervise it within our proposed method described in following sections, furthermore we experimented also other loss formulations, such as weighted Binary Cross Entropy between ground-truth mask and final segmentation output, but computed results showed worse network performances, hence an intermediate results supervision through Structure Loss is really effective for CaraNet.

3.4 Tandem solution

The main contribution of this thesis is the development of a newly solution based on a segmentation model and a classifier working together called *Tandem*. This strategy was developed because, as comparisons between different models will show in next chapter, among the tested architectures CaraNet[7] is the one that outperform others in terms of detected cysts, but at the same time is also the model that produces the highest number of wrong predictions. We believe this happens because CaraNet is really good at detect small cysts but is fooled by tissue structures that resemble them, which have been illustrated in section 3.1, consequently producing several wrong predictions. Aiming to improve CaraNet performances over our dataset reducing the number of wrong predicted cysts and, at the same time preserving as much as possible correctly detected ones, we introduced a classifier which act as a post-segmentation refinement component. The idea is to unfold each RGB image that have to be segmented in a fixed number of patches and use the classifier to predict if there are cysts or not in each obtained patch, assigning respectively label *True* or *False*. In the second case, corresponding areas of *False* classified patches in the segmentation prediction are erased, meaning that basically they are set to black. In this way wrong cysts number should be reduced from CaraNet computed mask. Our solution is not a simple ensemble, classification head is not trained alone but instead at the same time with CaraNet. This choice was made to make classifier aware of segmentation model prediction and exploit it to precisely predict labels. In fact, it receives in input patches extracted from original RGB image concatenated with corresponding patches of CaraNet segmented mask. Using classifier computed labels for patches, a *refinement mask* is built and finally it is multiplied with CaraNet prediction in order to refine it, deleting sections

of predictions corresponding to patches classified as "not containing cysts" from classifier. We registered evident improvement in terms of wrong cysts number using this strategy with a truly negligible reduction of properly identified cysts. Moreover, computed results show that CaraNet benefit from classifier presence: training process indeed improved more epoch after epoch when Tandem solution was applied, meanwhile during CaraNet solo training improvements stopped after just few epochs. More details about training strategy, schedulers and chosen classifier are reported in sections below.

3.4.1 Pipeline

As already anticipated in previous section, our idea is exploiting classification to improve segmentation results. Classifier role can be described as a sliding window that scans the segmentation model RGB input image to predict cysts presence, in order to finally refine segmentation output merging with it its predictions and thus contributing to the final segmentation result. In order to scan the input image with the classifier we unfold it in non-overlapping squared patches. Unfolding an image, or more specifically its tensor, means extracting sliding local blocks from it that we refer to as patches. Given a fixed patch dimension, is possible to extract overlapping or non-overlapping patches specifying a stride, but since we are interested in a classifier prediction of each input image section we chose the second option, setting the stride equal to patch width and height. Actually, the classifier scans RGB image not before, but only after the segmentation model has predicted a binary mask for it. This because both RGB input image and corresponding segmentation predicted mask are unfolded equally therefore their patches are coherent, i.e. the i-th segmentation output patch is the segmented version of i-th RGB input image patch, hence we concatenate corresponding patches before passing them to the classifier. In this way computed patches are not just RGB images but instead they have 4 channels: three color channels plus the additional fourth consisting of their binary predicted mask from segmentation model. We will refer to 4-dimensional patches as *enriched patches*. Figure 3.12 depict graphically the described process. Enriched patches are finally passed to classifier which will consequently predict a label for each one of them, being aware of segmentation prediction. Obviously, in order to train the classifier performing

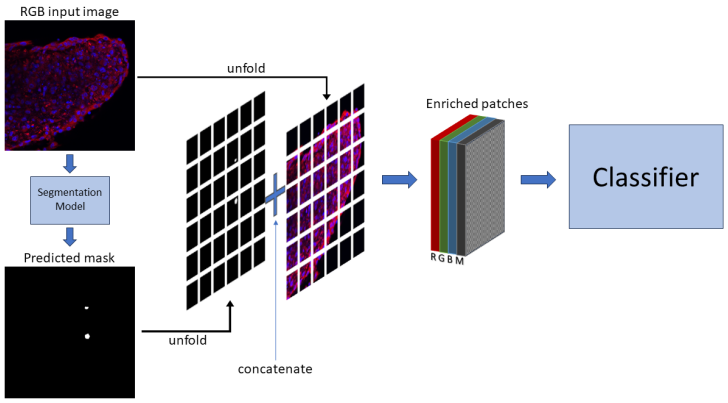


Figure 3.12: Patches enrichment process in proposed Tandem solution

back-propagation to adjust its weights, patches ground-truth labels are required for classifier loss computation. We assign, during training, a ground-truth label to each enriched patch exploiting the ground-truth segmentation mask associated with RGB input image from which they are extracted: this means that, other than input RGB image and predicted segmentation mask, also ground-truth mask associated with input image is unfolded in patches, specifically in the exact same way as the fists two. Therefore patches in unfolded RGB image match coherently the ones extracted from ground-truth mask. Exploiting this coherence we can assign to each enriched patch the ground-truth label:

- *True* if corresponding ground-truth mask patch contains cysts
- *False* if corresponding ground-truth mask patch does not contain cysts

A ground-truth mask patch has a fixed number of pixels, we use a criteria based on threshold t to define if it contains cysts: if the total number of pixels set to 1 (area corresponding to segmentation target object) is higher or equal to t we consider that patch as containing cysts, otherwise not. In our final configuration patches have dimension 128x128 and threshold t is set to 200, meaning that a patch will be labeled as *True* even if only a small portion of total pixels number is set to 1, this decision was made to correctly label as *True* also patches containing really small cysts. The ground-truth labeling process computed during training is depicted in figure 3.13. Once all enriched patches are labeled from the classifier, a refinement

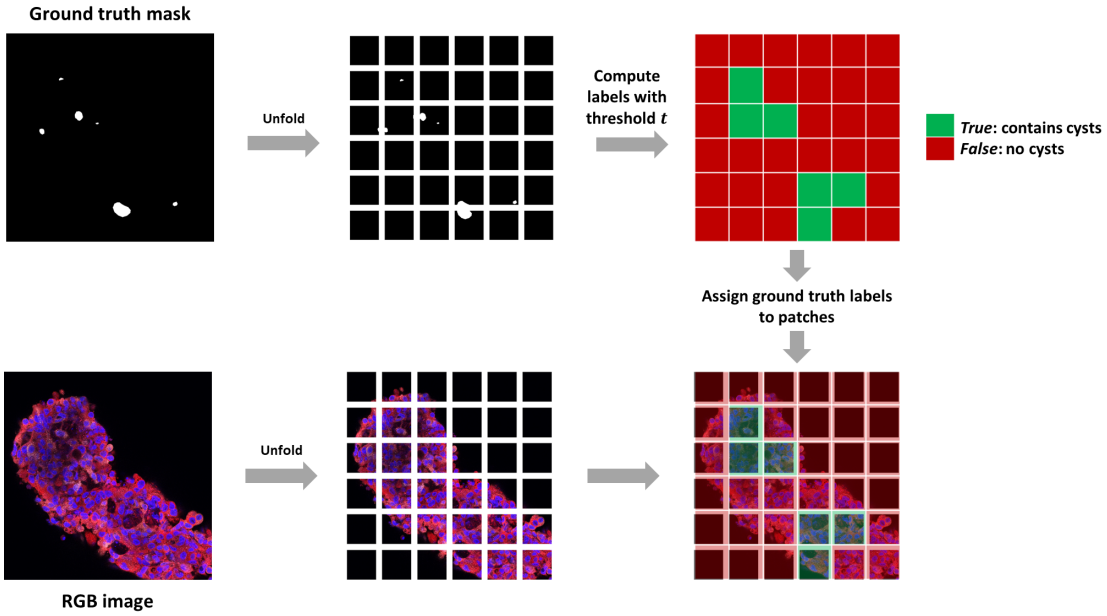


Figure 3.13: How ground-truth labels are assigned to each patch of a single RGB image during training

mask (R) is computed in order to use it later in the *refinement process* to improve the segmentation model predicted mask (P). The refinement mask is a matrix having the same dimension of the segmentation mask and consists of ones and zero values. At each patch classified as *False* corresponds a zero-value matrix of the same dimension, while to patches classified as *True* one-value matrices correspond. Finally, an element-wise multiplication between refinement mask and segmentation model output($R \cdot P$) is performed in order to erase all segmentation predictions inside areas which corresponds to *False* classified patches while keeping unaltered the predictions in areas matching *True* classified patches. The complete Tandem pipeline at inference time is graphically reported in figure 3.14. Of course this method is not perfect, for example when in a given patch there are both correct and wrong segmented cysts, if the classifier labels that patch as *Negative* the correctly predicted ones will be erased in the final output after the refinement, even if the segmentation model had detected them correctly. On the other hand is true also the opposite, because when a wrong predicted cyst lays in the same patch of correctly predicted ones, it will be kept in the final segmentation output if the patch is classified as *True*. Moreover, in case of big cysts that do not appear in a single

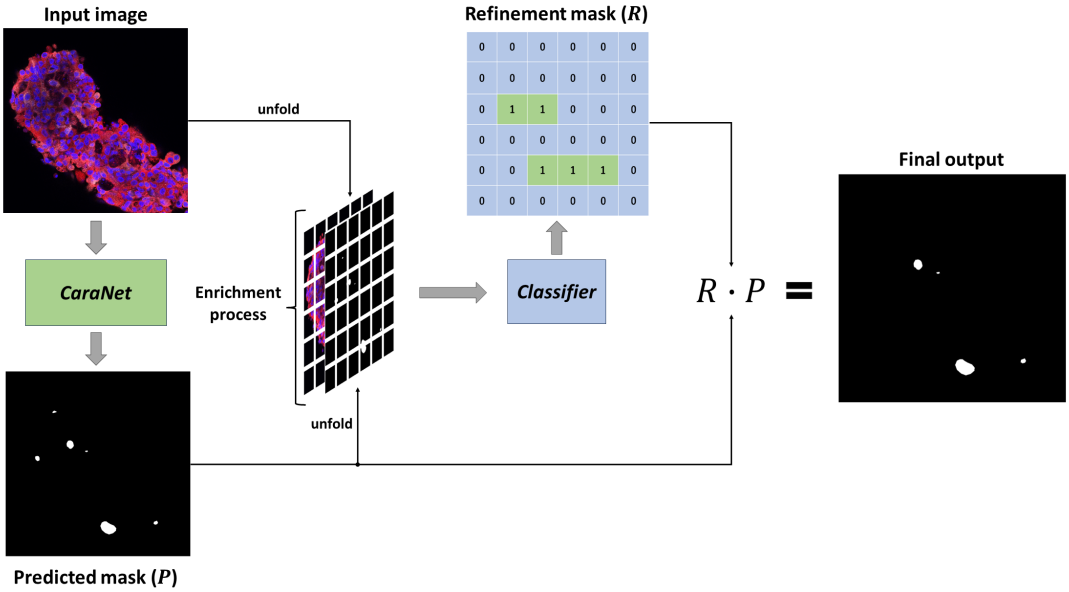


Figure 3.14: Tandem pipeline

patch but instead in more than one, is possible that in the final segmentation output those will appear trimmed because patches containing only small slices of them and nothing else could be classified as *False*, and consequently during refinement their corresponding portion in the segmentation prediction will be erased.

3.4.2 Classifier choice

Since we had to train both a segmentation model and a classification model at the same time we decided to focus our classifier choice on not too complex models in order to add only a relatively small number of parameters to tune other than the ones from segmentation model. A standard choice would have been a classic ResNet[21] solution, but we selected Res2Net[33] which is a modified version of the first. The main contribution of Res2Net to the original ResNet architecture is a novel residual block structure, specifically hierarchical residual-like connections are inserted inside each single residual block in order to improve the multi-scale representation ability of the network maintaining a similar computational load. We preferred Res2Net over ResNet because authors report in the paper better performances of the first over the latter utilizing the same number of layers. Res2Net obtains slightly better

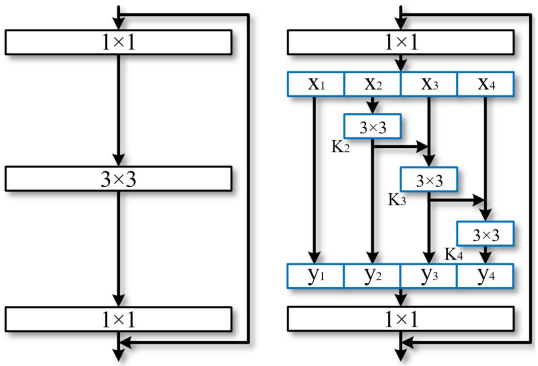


Figure 3.15: Res2Net residual block compared with standard version

results with several popular datasets, as for example ImageNet, even if compared with other popular models.

3.4.3 Classifier class imbalance problem

As already mentioned in 3.1, since cysts are really small objects images belonging to our dataset are extremely imbalanced because target objects occupy only a minimum portion of the entire picture. Consequently, unfolding RGB images and assigning a label to each patch according to relative patches extracted unfolding associated ground-truth mask, leads to unbalanced training data also for the classifier. From a single 768x768 RGB image divided in 36 patches of 128x128 pixels, on average only 10% of patches were labeled as positives (class 1). Class imbalanced data makes model training really hard because samples from the less frequent class will be fed into the classifier in only a few cases, this leads to training batches in which less frequent class samples will be only a minimum part, furthermore in some cases could also happen that no sample belonging to minority class would be present at all. Hence a generic loss function, such as Binary Cross Entropy (BCE), would not be able to correctly supervise the classifier training because will not consider the imbalance problem, moreover its value will be low even if the model predict always the label associated with preponderant class, thus the model will learn that predicting always the label associated with a specific class (the more frequent one) is a good solution while actually it is not. In other words, class imbalance can overwhelm training and lead to degenerate and not useful models. For this reason

the first loss function selected for classifier training was the balanced version of BCE which can be defined starting from the standard BCE formulation:

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases}$$

where $y \in \{0,1\}$ specify ground-truth class and $p \in [0,1]$ is the model estimated probability for class with label $y = 1$. Defining p_t as:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases}$$

we can rewrite the Binary Cross Entropy Loss as

$$\mathrm{CE}(p, y) = \mathrm{CE}(p_t) = -\log(p_t).$$

Weighted BCE version can be defined introducing the parameter $\alpha \in [0,1]$ for class 1 and $1-\alpha$ for class 0. Defining α_t as already done with p_t the balanced version of Cross Entropy Loss can be written as:

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t).$$

Unfortunately, balanced BCE has not been able to manage our data class imbalance. During our tests we noticed that classifier was learning to just predict always label 0, associated with samples from the most frequent class. We tested different values of α, computing also class weights from classifier training dataset analysis: starting from the entire dataset we excluded experiment 3 images from it, because those pictures were used as test set in our experiments, and divided each image in patches, computing associated labels from ground-truth masks in order to have a precise idea about the number of samples in both classes. We did this for each tested patch dimension. Using resized images of 768x768 pixels and patches size set to 128x128 pixels a total number of 35028 patches is obtained, class distribution is depicted in figure 3.16 where class 0 is associated with patches not containing cysts while class 1 with patches containing them. Indeed, we have an extremely imbalanced dataset

for classifier training. We computed class weights using different formulas but

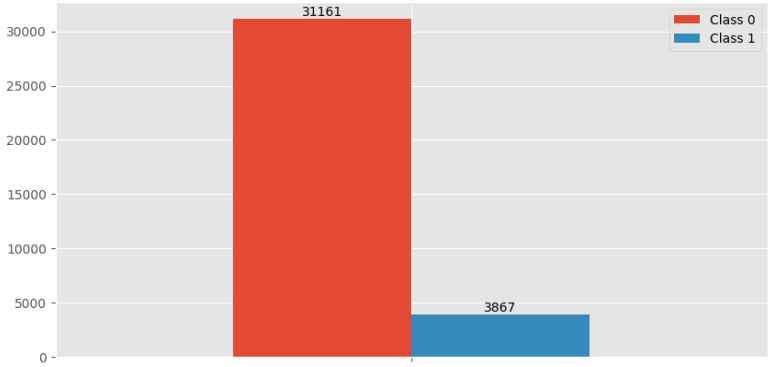


Figure 3.16: Classifier training dataset class distribution

classifier was not affected at all, it was struggling to learn in any case with balanced BCE loss. For this reasons we shifted our attention to another loss function able to manage highly imbalanced training dataset: Focal loss[36], which was explicitly developed for this purpose.

$$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{3.2}$$

Focal loss formulation is reported in equation 3.2, basically it's a modified version of the Cross Entropy loss, specifically a modulating term is applied to standard cross entropy formula aiming to focus learning on "hard" usually misclassified examples. The main idea behind this loss is that, during training, the model will confidently predict samples belonging to predominant class since their number is high enough to make the model aware of their characteristics and consequently it will properly learn to recognize them, on contrary it is not true for the less frequent class samples. Model predictions with a high probability are called *easy examples* and usually are associated with predominant class samples. On the other hand, predictions with not so high probability are called *hard examples*, which given the imbalanced dataset nature, happen to be associated with samples belonging to minority class. Focal loss force the model to focus on hard examples exploiting the modulating factor $(1-p_t)^{\gamma}$ where γ is a tunable *focusing parameter*, when it is set to 0 Focal loss acts basically as Cross Entropy loss. Using this formulation the loss contribution of highly confident classified samples ("easy" examples, where $p_t >> 0.5$) is reduced

in order to focus training on hard examples. As for Cross Entropy, also Focal loss has a weighted version, which is the one that we used in our experiments, where is possible to define class weights via α_t parameter. Weighted Focal loss is reported in equation 3.3. We tested different combination of α_t and γ values, obtaining best classifier response with $\alpha_t = 0.1$ and $\gamma = 2.0$.

$$\mathrm{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{3.3}$$

In our case the minority class was the one associated with patches containing cysts. Using Focal loss we were able to train properly the classifier and obtain satisfactory refinement results.

3.4.4 Loss computation

As described in previous sections, we use Structure Loss to supervise CaraNet training and Focal Loss for Res2Net classifier. Since we train those models together in Tandem, we sum those losses before performing back-propagation, thus the total loss formulation for Tandem ($\mathcal{L}_T$) will be the sum of both Focal loss and Structure loss contribution, as reported in equation 3.4.

$$\mathcal{L}_T = \mathcal{L}_{Structure} + \mathcal{L}_{Focal} \tag{3.4}$$

3.5 Evaluation metrics and methods

Image Segmentation results are always evaluated using pixel-wise metrics as IoU, Precision and Recall. In this section the aforementioned metrics will be explained, moreover also other kind of evaluation metrics introduced previously in [4] and specifically based on cysts will be described since also in this thesis cysts-wise IoU, Precision and Recall are used to evaluate and compare results obtained with different models. In standard pixel-wise metrics a True Positive (TP) pixel is a correctly predicted one, meaning that in both segmentation predicted mask and ground-truth mask the given pixel is labeled as "foreground", consequently a True Negative (TN) is a pixel labeled as "background" in both masks. Of course, a False Positive (FP) and a False Negative (FN) pixel are respectively a background

pixel labeled as "foreground" and vice versa. In our specific case, since our mask are binary, a "background" pixel is a pixel not belonging to a cyst, otherwise a "foreground" pixel is a pixel belonging to a cyst. Based on those information we can report pixel-wise evaluation metrics definitions in table 3.3: Even though these

Intersection over Union (IoU)	$\frac{TP}{TP+FN+FP}$
Precision (Pr)	$\frac{TP}{TP+FP}$
Recall (Re)	$\frac{TP}{TP+FN}$

Table 3.3: Pixel-wise metrics

metrics makes possible to get an overall idea of segmentation quality, and therefore compare performances of different methods, as already suggested in [4], they are not perfectly suitable for images inside our dataset. Segmentation target objects are cysts with variable sizes and the ones with larger size have obviously more pixels than smaller ones. This means that detection of larger cysts lead to unbalanced metrics because they will have a larger weight than small cysts. Since we are also greatly interested in the number of correctly or wrong predicted cysts, we will use also in this thesis cyst-wise metrics introduced in [4]. These evaluation metrics aim to weight all cysts equally regardless their number of pixels, moreover they are a cyst-based extension of pixel-wise ones described above. Cysts in ground-truth mask and predicted mask overlaps when have some pixels in common. A ground-truth cyst can be considered as Detected (DT) when it overlaps with a cyst in predicted mask or Missed (MS) when it does not overlap with a cyst in predicted mask. On the other hand, a predicted cyst in segmented mask can be labeled as Wrong (WR) when it does not overlap with any ground-truth cyst. Detected, Missed and Wrong are a cyst-based version of respectively TP, FN, FP. Finally we can report cyst-wise metrics definitions in table 3.4. It's important to define what happen when a ground-truth cyst overlap with multiple predicted cysts. In this situation one of the predicted is labeled as detected (DT) cyst while the rest of them as wrong (WR). Similarly, when a predicted cyst overlaps several ground truth ones only one is considered as detected (DT) cyst and the other as missed (MS). Image 3.17 from [4] depict all possible overlapping situations. As anticipated in previous sections,

IoU_{cyst}	$\frac{DT}{DT+MS+WR}$
Pr_{cyst}	$\frac{DT}{DT+WR}$
Re_{cyst}	$\frac{DT}{DT+MS}$

Table 3.4: Cyst-wise metrics

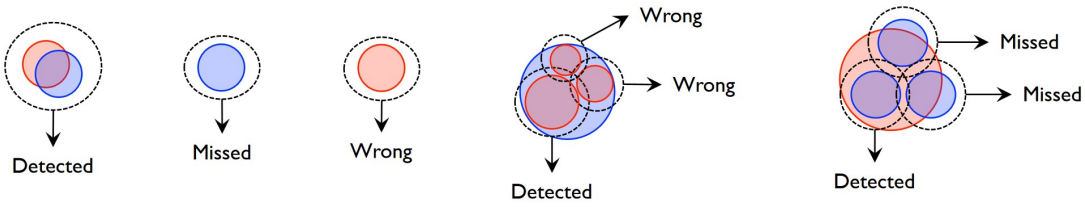


Figure 3.17: Possible cysts overlapping cases. In blue real cyst while predicted cyst in red.

our dataset consists of images belonging to 32 tubules and 4 experiments, each tubule is exclusively associated with only one experiment. Therefore images from a given tubule can be found only in one experiment. We divided dataset in train, validation and test set during our tests according to experiments, in order to train each model with images from a large number of tubules and test them later with images having slightly different features. Specifically, we decided to use experiment number 3, which contains 103 images, as test set and consequently the remaining 973 pictures have been split in training and validation set respectively with an 80-20% proportion. We use the third experiment as test set because the first two have few images and consequently using them to validate models performances would not return an accurate result, on the other hand experiment number 4 is the one with more images among the four, this means that removing it from training would have compromised the learning procedure because remaining images would not be enough to properly train models. Experiment number 3, indeed is the best candidate to be our test set and compute reliable performance evaluations. Moreover, in [4] Monaco et al. tested their models using a leave-one-out cross-validation approach based on tubules called *Leave-One-Tubule-Out* (LOTO) which, as the name suggests, consists of using 31 tubules as training set and the remaining

one as test set. In this way, averaging results computed over each tubule used as test set, the best possible estimate of model quality on unseen data is computed. In following chapters we will compare model performances obtained using the firstly described training setting, while for our Tandem method performances will be evaluated also with LOTO strategy in order to compare it even with models analyzed in [4].

Chapter 4

Results

In this chapter results computed with CaraNet, CE-Net and Unet++ are reported in order to compare them. Moreover, also some models ensembles are analyzed and finally Tandem proposed solution results, including LOTO strategy ones, are presented and discussed.

4.1 Training settings

Unet++, CE-Net and CaraNet has been trained using a batch size of 4 for training, validation and test. Images are augmented through augmentation techniques previously described in section 3.2. Training configurations are reported in table below: Adam optimizer, Cosine Annealing Warm Restart and learning rate set

	Loss	Scheduler	Optimizer	Learning rate
Unet++	BCE	CAWR	Adam	$1e-4$
CE-Net	Dice	CAWR	Adam	$1e-4$
CaraNet	Structure	CAWR	Adam	$1e-4$

CAWR: Cosine Annealing with Warm Restart

Table 4.1: Training settings

to 0,0001 were chosen as standard training settings because after several tests performed with different optimizer and scheduler combinations we noticed that this

specific was the most suitable. We trained models with early stopping termination criterion. The number of epochs needed to reach model convergence was architecture specific, it went from a minimum of 10 to a maximum of 25 whit early stopping patience parameter set to 10 epochs. Training and testing has been performed on Google Colab notebook with a 16GB NVIDIA Tesla T4 GPU.

4.2 Models comparison

Results computed with experiment 3 as test set are reported in terms of detected, missed and wrong predicted cysts in figure 4.1. Is clear that both CE-Net and CaraNet are able to outperform the baseline UNet++ model in terms of detected cysts, and obviously also performances over missed cysts are positively affected in the same way. Unfortunately, for what concerns wrong segmented cysts Unet++ is still better than the other two: indeed CaraNet produced three times more its number of wrong segmented cysts, while CE-Net performed even worse. Is clear that our two literature selected models are able to detect more cysts than UNet++ but, at the same time, they also introduce many undesirable wrong predictions. Other interesting information useful for comparing models and understanding their

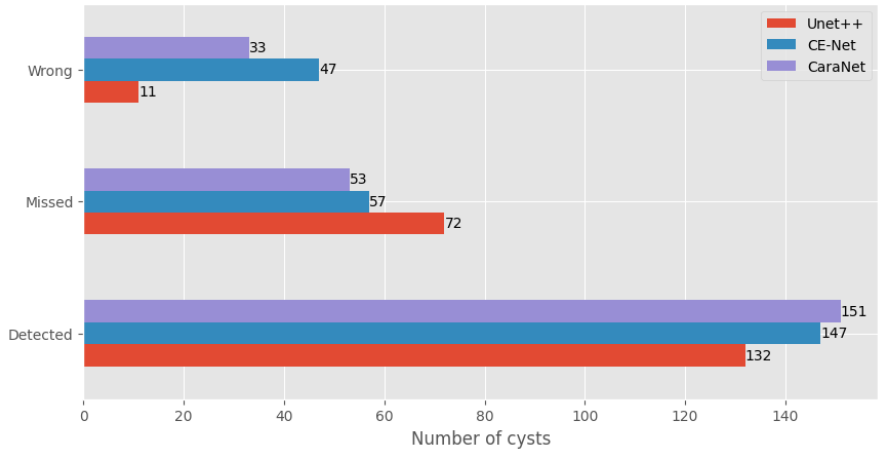


Figure 4.1: Raw models results computed with experiment 3 as test set

behavior with cyst segmentation can be extracted from image 4.2, where the number of missed cysts for each size zone (which have been described previously in table 3.1) and architecture is reported. Better performances in terms of detected small cysts,

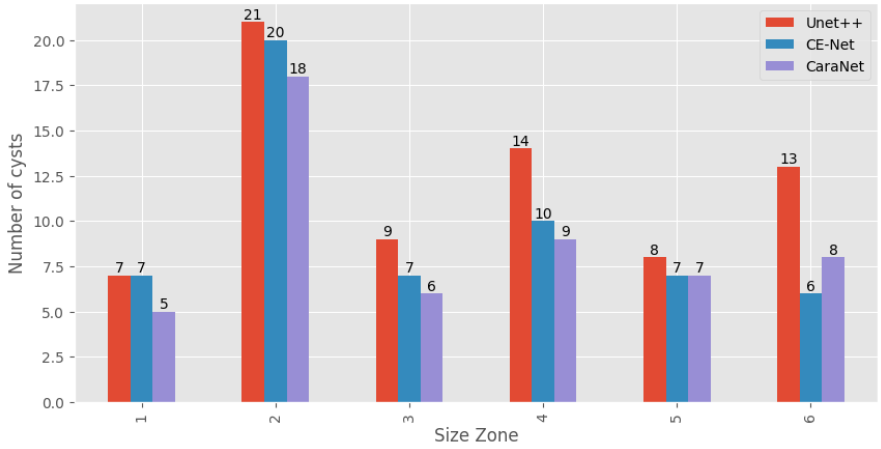


Figure 4.2: Number of missed cysts for each cyst size zone

and consequently reduced number of missed ones, from CaraNet can be noticed in almost every size zone. Moreover, we can report the same information also according to cysts size intervals in figure 4.3 where is even more clear how CaraNet outperforms the other two models when dealing with small cysts, intervals associated with smallest cysts show how good this architecture is at segmenting tiny objects. 4.3 Based on these evidence we focused our attention on CaraNet, which again was

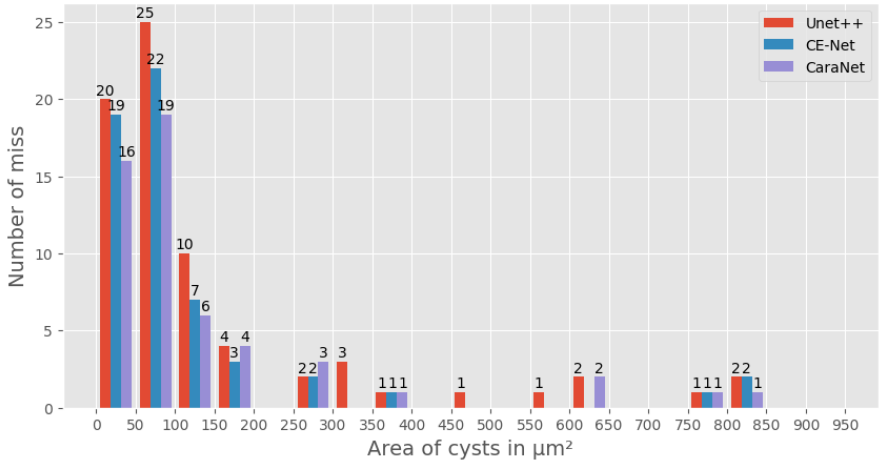


Figure 4.3: Number of missed cysts for each dimension interval

the most promising architecture. Specifically, we decided to take a step further implementing the so called Multi-Scale (MS) training which consists of not using

any augmentation technique but instead increasing and decreasing training images dimension. This means that each image, during training, is shown to the model 3 times at 1x,0.75x and 1.25x. This technique was originally presented in CaraNet paper, authors report that it is beneficial if used instead of augmentations. Indeed also in our tests results have been improved in terms of detected cysts training CaraNet with multi-scale but also wrong prediction number grew up considerably, results are reported in figure 4.4. Computed results show that CaraNet architecture

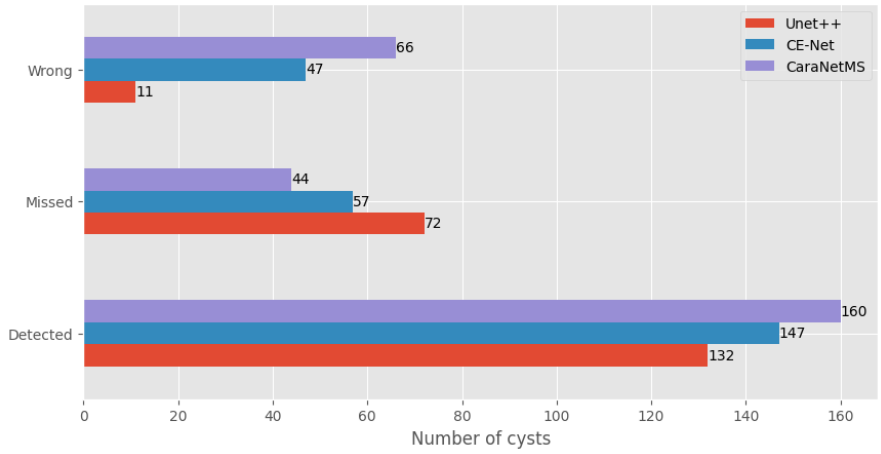


Figure 4.4: CaraNet with Multi-Scale training compared to other models

actually benefits from multi-scale training in terms of predicted detected cysts number, passing from 151 of CaraNet with augmentations to 160, but the price for that improvement is a doubled up wrong predictions number, which made us think that it is not the most suitable technique for our dataset. We also applied multi-scale training to UNet++, aiming to see how this architecture would respond, but results were really bad: the number of correctly detected cysts boosted up from 131 to 171 but wrongs grew up exponentially, passing from 11 to 307. During our experiments, we observed that multi-scale procedure slowed down training: more epochs were required to reach model convergence and also a lot more GPU memory resources was needed. Indeed, since each training sample have to be re-scaled two times (0,75x and 1,25x), each training batch sample number will increase passing from 4 to 12, thus multi-scale training is computational expensive. At this point, having trained several models, we decided to try ensembles, which basically means using different models together at inference time in order to average

their predictions and compute, possibly, segmentation results better than the ones obtained with single models. Below are reported results computed with some ensembles, we tested several combinations of models, also weighted, but results were not exciting.

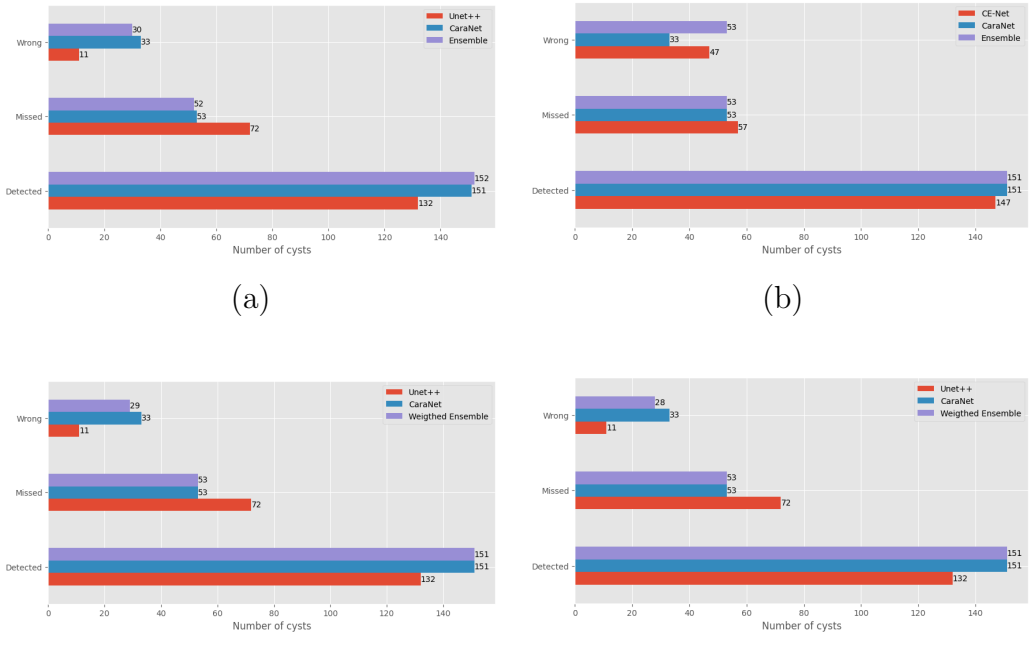


Figure 4.5: (a), (b): Ensambles via sum of outputs; (c), (d): Weighted ensembles

In order to build an ensemble each model has been trained separately, than at inference time their weights were loaded and ensemble output is computed merging models predictions with a simple sum, or as in (c) and (d), through weighted sum. Results show that the best possible outcome using ensembles is a small reduction in terms of wrong predicted cysts, but no improvement in detected/missed cysts has been registered at all. From reported plots we can understand that CaraNet was leading in terms of detection and the best ensembles could do was to reduce the wrong cysts number by 3/5 units, specifically when Unet++ results were merged with CaraNet ones. On the other hand, CE-Net related ensembles did not bring any improvements but rather made worse performances in terms of wrong predictions. Since CaraNet resulted the best model from our tests over experiment 3, we decided to use classification aiming to boost its performances, specifically we wanted to use classifier contribution to obtain an efficiency in terms of wrong predicted cysts

similar to UNet++, or at least better than CaraNet one. Therefore, we thought to train classifier and segmentation model separately, in order to use them together only at inference time. In this way we could exploit also CaraNet multi-scale trained version, which we will refer to as CaraNetMS, because the multi-scale is memory-resource hungry only at training time. This classifier and segmentation

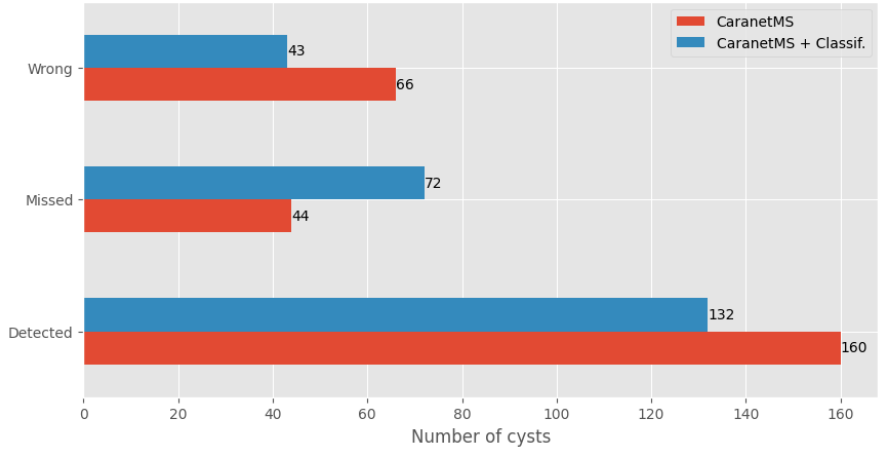


Figure 4.6: Comparison between CaraNetMS and CaraNetMS + Classifier

approach can be considered as an embryonic version of Tandem solution, specifically the idea was to segment an input image and than, using the classifier, assign a label to each patch of the input RGB image associated with a segmented cyst in the segmentation model output, obtaining finally a mask in which cysts related to classified-as-wrong patches were erased, basically the classifier was a second verification step for segmented cysts. In order to train the classifier, a dataset of patches extracted from training samples (i.e. our original dataset without images belonging to experiment 3) was built. It had two classes: positive class, in which patches contained centered cysts, and negative class, where patches did not contain any cyst. The first kind of samples were obtained using ground-truth masks, each cyst depicted in them was extracted from associated RGB image in order to center it, meanwhile the negative patches were computed dividing each RGB image in a fixed number of patches, only the ones containing tissue and not cysts were selected. In this way we did not use as negative examples totally black patches which could negatively affect classifier training. We used ResNet50 trained with the previously described patches dataset as classifier in sequence with CaraNetMS,

performances were better in terms of wrong cysts, which were reduced, but detected cysts were too much negatively affected, see figure 4.6. Finally, we decided to define another approach: train classifier and segmentation model together, i.e. the Tandem method, our goal was to achieve a higher Unet++ detected cysts number while reducing considerably the quantity of wrong predicted cysts found when dealing with literature selected models. Therefore, we decided to focus our attention on CaraNet trained with augmentations and not with multi-scale procedure, indeed our idea of training a segmentation model together with a classifier would not have been possible using multi-scale because, as mentioned before, is a computational expensive training strategy and we had a limited amount of GPU memory available. Tandem solution results over experiment 3 images are reported in figure 4.7

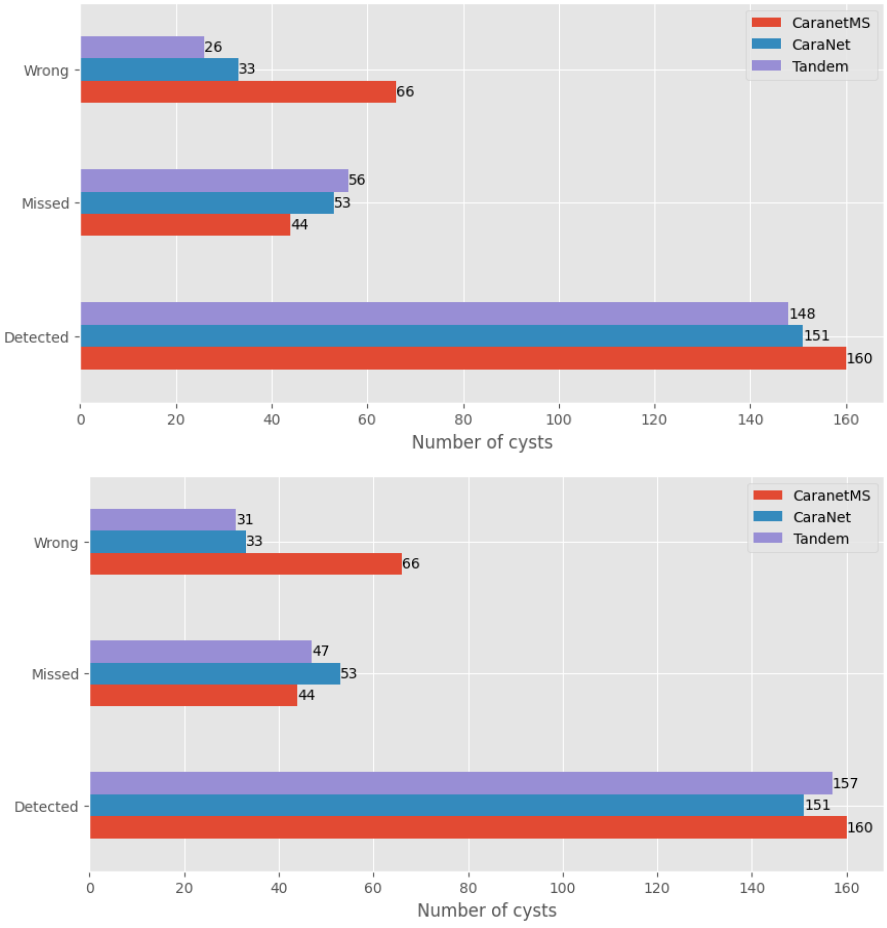


Figure 4.7: Tandem results computed with two different seeds

Tandem method performed well on experiment 3 images, it was able to reduce the number of wrong predictions and, at the same time, preserve a considerable amount of correctly detected cysts. For this reason we finally performed a LOTO cross validation to precisely assess Tandem performances, we performed this evaluation strategy also for standalone CaraNet in order to better understand the classifier contribution to this specific architecture. In following table LOTO computed results are reported for both CaraNet and Tandem, moreover also performances of other models with cysts dataset from [4] are included in order to further expand Tandem comparison. As clearly visible from table 4.2 Tandem method is able to

Model	IoU	IoU_{cyst}	Pr_{cyst}	Re_{cyst}
UNet	0.5845 ± 0.0451	0.6029 ± 0.035	0.7743 ± 0.0448	0.7463 ± 0.0353
UNet++	0.5821 ± 0.0525	0.6059 ± 0.0493	**0.8385 ± 0.0321**	0.696 ± 0.0554
HardNet-MSEG	0.5311 ± 0.0527	0.5664 ± 0.0453	0.787 ± 0.0324	0.6812 ± 0.0516
PraNet	0.5868 ± 0.0493	0.6224 ± 0.0369	0.7171 ± 0.0372	0.8276 ± 0.0323
UACANet	**0.6239 ± 0.0359**	0.6464 ± 0.0395	0.7406 ± 0.0411	0.8346 ± 0.0262
CaraNet	0,4930±0,0096	0,5723±0,0139	0,6416±0,0166	**0,8441±0,0053**
Tandem	0,5887±0,0082	**0,6609±0,0098**	0,7898±0,0066	0,7994±0,0069

Table 4.2: LOTO performances comparison

outperform UNet and Unet++ in terms of cyst-wise IoU, actually it obtained the best result among all considered models. UACANet has the best Pixel-wise IoU result, followed by Tandem method at second place. Indeed, considering both pixel and cysts based IoU metrics, is evident how beneficial is the classifier contribution to CaraNet predictions, the latter alone in fact did not performed great results because of its tendency to produce a high number of wrong cysts. For what concerns cyst-wise Precision, even though Unet++ is still the best option so far, Tandem method obtains also here the second best score. Finally, in terms of cyst-wise Recall CaraNet performed best because of the high number of correctly segmented cysts, while Tandem reached a good positioning also there.

Chapter 5

Conclusions and future works

In this thesis we addressed the medical image segmentation topic, more specifically we focused our attention on segmentation of small medical objects, which in our case were ADPKD kidney cysts. We exposed the challenges that characterizes this task and the reasons why it is so important to improve available solutions. Moreover, a brief view of medical image segmentation state-of-art has been reported. We selected from literature two models explicitly developed to well perform in segmentation of medical images depicting small target objects, i.e. CaraNet and CE-Net, explaining their main architectural aspects. Our first contribution is testing those models for the first time with a dataset containing images of engineered kidney tubules affected by ADPKD and its characteristic small and scattered cysts. In order to improve segmentation performances computed by Monaco et al. in [4] over the previously mentioned dataset with several and popular segmentation methods, we tried different solutions involving our literature selected models. Finally, our main contribution is introduced: we developed a segmentation method called Tandem consisting of a segmentation model (CaraNet) trained together with a classifier which is able to improve segmentation performances deleting segmented cysts identified as wrong. Classification head scans a 4-channels image obtained through concatenation of input RGB image and CaraNet segmented mask to produce a refinement mask. In this way the classifier exploits both input image

and segmentation mask to improve the final output mask combining refinement mask with intermediate CaraNet segmentation prediction. Our experiments results proved that Tandem method is able to outperform popular segmentation models, as UNet and UNet++, in this task. More specifically it reaches the best cyst-wise IoU performance. We believe that our method is a valid solution in medical segmentation tasks characterized by small target objects having, more than anything else, a scattered and unpredictable disposition in the input image, therefore future development involve testing Tandem method with other challenging datasets with similar images and features. Another open challenge could be improving the proposed method exploiting multi-scale training technique to reach an even higher number of correctly segmented cysts using more powerful GPU graphic cards since this technique is memory resource hungry. Since Tandem method is based on a classifier acting as a post segmentation refinement component, another kind of usage would be extracting a heat map for the input RGB image based on classifier probability outputs. At the end of the day, physicians are interested in a fast detection of cysts in images with high precision. Therefore, considering Tandem method limits derived from the fixed grid patches extraction, i.e. possible deletions of correctly detected cysts which lay in the same patch of wrong ones, Tandem utilization can not be optimal in some cases. For these reasons an open challenge is using the classification head as a *soft refinement*, in which it is used only to suggests physicians the probability of cysts presence in each area of the image instead of directly editing the CaraNet predictions. In this way, having both CaraNet segmentation mask and the heat map computed from the classifier, physicians could refine the segmented mask themselves.

Bibliography

[1] Albert C.M. Ong and Peter C. Harris. «Molecular pathogenesis of ADPKD: The polycystin complex gets complex». In: *Kidney International* 67.4 (2005), pp. 1234–1247. ISSN: 0085-2538. DOI: https://doi.org/10.1111/j.1523-1755.2005.00201.x. URL: https://www.sciencedirect.com/science/article/pii/S0085253815505796 (cit. on p. 1).

[2] Arlene B. Chapman et al. «Autosomal-dominant polycystic kidney disease (ADPKD): executive summary from a Kidney Disease: Improving Global Outcomes (KDIGO) Controversies Conference». In: *Kidney International* 88.1 (2015), pp. 17–27. ISSN: 0085-2538. DOI: https://doi.org/10.1038/ki.2015.59. URL: https://www.sciencedirect.com/science/article/pii/S2157171615321468 (cit. on p. 1).

[3] Carsten Bergmann. «ARPKD and early manifestations of ADPKD: the original polycystic kidney disease and phenocopies». en. In: *Pediatric Nephrology* 30.1 (Jan. 2015), pp. 15–30. ISSN: 1432-198X. DOI: 10.1007/s00467-013-2706-2. URL: https://doi.org/10.1007/s00467-013-2706-2 (visited on 02/21/2024) (cit. on p. 1).

[4] Simone Monaco et al. «Cyst segmentation on kidney tubules by means of U-Net deep-learning models». In: *2021 IEEE International Conference on Big Data (Big Data)*. Dec. 2021, pp. 3923–3926. DOI: 10.1109/BigData52589.2021.9671669 (cit. on pp. 1–3, 16, 17, 21, 38–41, 49, 50).

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV] (cit. on pp. 2, 10, 11, 20).

[6] Zaiwang Gu, Jun Cheng, Huazhu Fu, Kang Zhou, Huaying Hao, Yitian Zhao, Tianyang Zhang, Shenghua Gao, and Jiang Liu. «CE-Net: Context Encoder Network for 2D Medical Image Segmentation». In: *IEEE Transactions on Medical Imaging* 38.10 (Oct. 2019), pp. 2281–2292. ISSN: 1558-254X. DOI: 10.1109/tmi.2019.2903562. URL: http://dx.doi.org/10.1109/TMI.2019.2903562 (cit. on pp. 3, 12, 22, 23, 25).

[7] Ange Lou, Shuyue Guan, Hanseok Ko, and Murray H. Loew. «CaraNet: context axial reverse attention network for segmentation of small medical objects». In: *Medical Imaging 2022: Image Processing*. Vol. 12032. International Society for Optics and Photonics. SPIE, 2022, pp. 81–92. DOI: 10.1117/12.2611802 (cit. on pp. 3, 26, 30).

[8] Antonelli et al. «A view of computational models for image segmentation». In: *Ann Univ Ferrara 68* (2022), pp. 277–294 (cit. on pp. 4, 5).

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV] (cit. on p. 5).

[10] Mohammad Hesam Hesamian, Wenjing Jia, Xiangjian He, and Paul Kennedy. «Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges». en. In: *Journal of Digital Imaging* 32.4 (Aug. 2019), pp. 582–596. ISSN: 1618-727X. DOI: 10.1007/s10278-019-00227-x. URL: https://doi.org/10.1007/s10278-019-00227-x (visited on 01/13/2024) (cit. on pp. 6, 8, 9).

[11] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. «Image Segmentation Using Deep Learning: A Survey». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (July 2022). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 3523–3542. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3059968. URL: https://ieeexplore.ieee.org/document/9356353 (visited on 12/30/2023) (cit. on pp. 6, 9).

[12] Olga Russakovsky, Jia Deng, Zhiheng Huang, Alexander C. Berg, and Li Fei-Fei. «Detecting avocados to zucchinis: what have we done, and where are we going?» In: *International Conference on Computer Vision (ICCV)*. 2013 (cit. on pp. 6, 20).

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386 (cit. on p. 6).

[14] Xiaozheng Xie, Jianwei Niu, Xuefeng Liu, Zhengsu Chen, Shaojie Tang, and Shui Yu. «A survey on incorporating domain knowledge into deep learning for medical image analysis». eng. In: *Medical Image Analysis* 69 (Apr. 2021), p. 101985. ISSN: 1361-8423. DOI: 10.1016/j.media.2021.101985 (cit. on p. 7).

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 8).

[16] Keiron O'Shea and Ryan Nash. «An Introduction to Convolutional Neural Networks». In: *CoRR* abs/1511.08458 (2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458 (cit. on p. 8).

[17] PhungVan Hiep and RheeEun Joo. «A Deep Learning Approach for Classification of Cloud Image Patches on Small Datasets». In: *Journal of Information and Communication Convergence Engineering* 16.3 (Sept. 2018), pp. 173–178 (cit. on p. 8).

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. «Fully Convolutional Networks for Semantic Segmentation». In: *CoRR* abs/1411.4038 (2014). arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038 (cit. on p. 9).

[19] Evgenii Sovetkin, Elbert Jan Achterberg, Thomas Weber, and Bart E. Pieters. *Encoder-decoder semantic segmentation models for electroluminescence images of thin-film photovoltaic modules*. 2020. arXiv: 2010.07556 [eess.IV] (cit. on p. 10).

[20] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. «SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495. DOI: 10.1109/TPAMI.2016.2644615 (cit. on p. 10).

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on pp. 10, 23, 34).

[22] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. «UNet++: A Nested U-Net Architecture for Medical Image Segmentation». In: *CoRR* abs/1807.10165 (2018). arXiv: 1807.10165. URL: http://arxiv.org/abs/1807.10165 (cit. on pp. 11, 21, 22).

[23] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. 2016. arXiv: 1606.04797 [cs.CV] (cit. on pp. 12, 25).

[24] Chien-Hsiang Huang, Hung-Yu Wu, and Youn-Long Lin. *HarDNet-MSEG: A Simple Encoder-Decoder Polyp Segmentation Neural Network that Achieves over 0.9 Mean Dice and 86 FPS*. 2021. arXiv: 2101.07172 [cs.CV] (cit. on p. 12).

[25] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. *PraNet: Parallel Reverse Attention Network for Polyp Segmentation*. 2020. arXiv: 2006.11392 [eess.IV] (cit. on p. 12).

[26] Agostina J. Larrazabal, Cesar Martinez, and Enzo Ferrante. *Anatomical Priors for Image Segmentation via Post-Processing with Denoising Autoencoders*. 2019. arXiv: 1906.02343 [eess.IV] (cit. on pp. 13, 14).

[27] Artem Sevastopolsky, Stepan Drapak, Konstantin Kiselev, Blake M. Snyder, Jeremy D. Keenan, and Anastasia Georgievskaya. *Stack-U-Net: Refinement Network for Image Segmentation on the Example of Optic Disc and Cup*. 2018. arXiv: 1804.11294 [cs.CV] (cit. on p. 14).

[28] Vajira Thambawita, Steven A. Hicks, Pål Halvorsen, and Michael A. Riegler. *DivergentNets: Medical Image Segmentation by Network Ensemble*. 2021. arXiv: 2107.00283 [eess.IV] (cit. on pp. 14, 15).

[29] Jiwoong J Jeong, Amara Tariq, Tobiloba Adejumo, Hari Trivedi, Judy W Gichoya, and Imon Banerjee. «Systematic Review of Generative Adversarial Networks (GANs) for Medical Image Classification and Segmentation». In:

Journal of digital imaging 35.2 (Apr. 2022), pp. 137–152. ISSN: 0897-1889. DOI: 10.1007/s10278-021-00556-w. URL: https://europepmc.org/articles/PMC8921387 (cit. on p. 15).

[30] K. Wada. *labelme: Image Polygonal Annotation with Python.* 2016. URL: https:%20//github.com/wkentaro/labelme (cit. on p. 17).

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision.* 2015. arXiv: 1512.00567 [cs.CV] (cit. on pp. 23, 27).

[32] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. «Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations». In: *Lecture Notes in Computer Science.* Springer International Publishing, 2017, pp. 240–248. ISBN: 9783319675589. DOI: 10.1007/978-3-319-67558-9_28. URL: http://dx.doi.org/10.1007/978-3-319-67558-9_28 (cit. on p. 25).

[33] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. «Res2Net: A New Multi-Scale Backbone Architecture». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.2 (Feb. 2021), pp. 652–662. ISSN: 1939-3539. DOI: 10.1109/tpami.2019.2938758. URL: http://dx.doi.org/10.1109/TPAMI.2019.2938758 (cit. on pp. 26, 34).

[34] Ange Lou and Murray Loew. *CFPNet: Channel-wise Feature Pyramid for Real-Time Semantic Segmentation.* 2021. arXiv: 2103.12212 [cs.CV] (cit. on p. 27).

[35] Shuhan Chen, Xiuli Tan, Ben Wang, and Xuelong Hu. *Reverse Attention for Salient Object Detection.* 2019. arXiv: 1807.09940 [cs.CV] (cit. on p. 29).

[36] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. *Focal Loss for Dense Object Detection.* 2018. arXiv: 1708.02002 [cs.CV] (cit. on p. 37).

Acknowledgements

Ringraziamenti

Ringrazio i miei genitori per avermi donato la possibilitá di intraprendere il mio percorso accademico culminato con questa tesi, li ringrazio per avermi dato sostegno indiscusso, sempre, emotivo e non solo. Neanche un momento é stato dato per scontato da parte mia e ve ne saró sempre grato. Ringrazio Francesca, grazie per avermi sempre incoraggiato e per essere sempre stata presente, soprattutto nei momenti in cui ho creduto di non essere all'altezza e di non farcela. Grazie per avermi fatto sentire leggero quando ne avevo bisogno. Ringrazio infine la mia famiglia e i miei amici per avermi spronato a dare il massimo e per aver sempre creduto in me. Grazie per ogni parola o momento che mi avete dedicato.