



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Multi organ semantic segmentation in CT scans

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE ENGINEERING -
INGEGNERIA INFORMATICA

Author: **Raffaele Spinoni**

Student ID: 946381

Advisor: Prof. Daniele Loiacono

Co-advisors: Leonardo Crespi

Academic Year: 2021-22

Abstract

In the context of medical image analysis, accurate algorithms for automatic segmentation of organs at risk have the potential of improving disease diagnosis and radiotherapy treatment planning. In this work we focus on CT Scan images obtained from patients under treatment and labeled by medics, we perform a series of experiments with CNNs analyzing also the transferability of features coming from models pretrained on other data sources. We compared also two approaches in the context of multi-organ segmentation: the use of ensemble methods made of multiple binary nets, and the creation of a single network for multi-organ segmentation.

Keywords: Semantic segmentation, Medical image segmentation, CT-Scan, Transfer learning, Organ at risk

Abstract in lingua italiana

Nel contesto dell'analisi di immagini mediche, algoritmi automatici ad accurati che performano la segmentazione degli organi a rischio hanno il potenziale di migliorare la diagnosi di malattie e il planning dei trattamenti di radioterapia. In questo elaborato ci siamo concentrati sulle immagini CT Scan ottenute da pazienti in corso di trattamento e annotate dai medici. Abbiamo eseguito una serie di esperimenti usando Reti convoluzionali, analizzando anche la possibilità di trasferire *features* da modelli pre-allenati su altre basi di dati. Abbiamo paragonato anche due approcci nel contesto della segmentazione multi-organo: l'uso di metodi *ensemble* costituiti da varie reti binarie, e la creazione di una singola rete in grado di segmentare multipli organi.

Parole chiave: Segmentazione semantica, segmentazione di immagini mediche, CT-Scan, Transfer learning, Organi a rischio

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
1.1 Scope	2
1.2 Thesis structure	3
2 State of the Art and Theoretical background	5
2.1 Theoretical Background	5
2.1.1 Total Marrow and lymphoid irradiation	5
2.1.2 Artificial Neural Networks	6
2.1.3 Stochastic Gradient Descent	8
2.1.4 Convolutional Neural Network	9
2.1.5 Transfer learning	10
2.2 Related works	13
2.2.1 Artificial Intelligence and Deep learning	13
2.2.2 AI applied to medical image analysis	13
2.3 Summary	16
3 Dataset	17
3.1 Computerized Tomography	17
3.2 Dataset Technicalities	18
3.2.1 StructSeg	18
3.2.2 SegTHOR	18
3.2.3 Target Dataset	19
3.2.4 Intensity Analysis	21

3.3	Preprocessing	22
3.3.1	Normalization	22
3.3.2	Cropping	22
3.3.3	Rotate	23
3.4	Data Augmentation	23
3.4.1	Elastic Transformation	24
3.4.2	Grid Distortion	24
3.4.3	Rotation	25
3.5	Summary	25
4	System design and overview	27
4.1	Input Pipeline	27
4.1.1	Training from scratch	28
4.1.2	Fine-tuning	28
4.2	Networks Used	28
4.2.1	Unet	29
4.2.2	SE-ResUnet	30
4.2.3	DeepLabV3	31
4.3	Refinements of Small Organs	32
4.3.1	Solution Architecture	32
4.3.2	Small Window retrieval	34
4.4	Ensemble methods	35
4.5	Training	36
4.5.1	Loss	36
4.5.2	Learning Rate Scheduler	38
4.5.3	Implementation details	39
4.6	Evaluation	39
4.6.1	Confusion Matrix	39
4.6.2	Dice Similarity Coefficient	40
4.6.3	Jaccard Index	40
4.7	Summary	41
5	Design of Experiments	43
5.1	Binary Nets	43
5.2	Small organ study	44
5.2.1	Coarse network	45
5.2.2	Refinement network	45
5.2.3	Reduce data augmentation	45

5.2.4	Add Context	46
5.2.5	Use the Smart Window	46
5.3	Transfer Learning	46
5.3.1	Reduced number of patients	47
5.3.2	Layer freezing	48
5.3.3	Transfer Learning Summary	48
5.4	Multiclass	49
5.4.1	Single multiclass network	49
5.4.2	Last Layer Feature Fusion Ensemble Method	49
5.5	Summary	50
6	Results	51
6.1	Training from scratch results	51
6.1.1	Small Organs	57
6.1.2	Esophagus study	61
6.1.3	Discussion	62
6.2	Fine-tuning results	62
6.2.1	Left Lung	62
6.2.2	Right Lung	64
6.2.3	Heart	66
6.2.4	Marrow	68
6.2.5	Discussion	70
6.3	Multi Class results	71
6.3.1	Multiclass segmentation - 5 targets	71
6.3.2	Multiclass segmentation - 6 targets	71
6.3.3	Multiclass segmentation - 8 targets	72
6.3.4	Discussion	73
6.4	Last Layer Feature Fusion results	73
6.4.1	Last Layer Feature Fusion - 5 targets	74
6.4.2	Last Layer Feature Fusion - 6 targets	74
6.4.3	Last Layer Feature Fusion - 8 targets	75
6.4.4	Discussion	75
6.5	Summary	76
7	Conclusions and future works	77
7.1	Open Problems	78
7.2	Future Development	78

Bibliography	81
A Humanitas Dataset Information	85
B DICOM File Information	89
List of Figures	91
List of Tables	93
List of Symbols	95
Acknowledgements	97

1 | Introduction

Radiation therapy is a type of cancer treatment that uses beams of intense energy to kill cancer cells. The high-energy beams come from a machine outside the patient's body that aims at a precise target. Radiation therapy damages unhealthy cells by destroying the genetic material that controls how cells grow and divide. While both healthy and cancerous cells are damaged by radiation therapy, the goal of the latter is to destroy as few normal and healthy cells as possible.

To develop an effective treatment it's therefore important to correctly identify the regions to be targeted. In the Medical realm, these regions are called organs-at-risk (OaR), they are healthy tissues/organs placed near the clinical target volume (CTV) whose irradiation could cause damage that would make changes to the radiotherapy treatment plan. The heart, for example, in radiotherapy of left breast cancer, is an organ at risk [27]. Is therefore crucial to identify OaR regions in the patient's body before any radiation treatment. At the moment, in most cases, the contouring of the targets is done manually by doctors. Manual segmentation poses significant challenges for human experts, both because of the variability of tumor appearance but also because of the need to consult multiple images from different CT-Scan sequences in order to classify tissue type correctly. This laborious effort is not only time-consuming but prone to human errors and results in significant intra- and inter-rater variability [17]. To tackle these limitations, automatic segmentation systems are developed, these systems aim to provide a cheap and scalable solution for treatment planning. Automatic multi-organ segmentation techniques represent a significant innovation in daily practices of radiation therapy, expediting the segmentation process and enhancing contour consistency. The recent development of AI and in particular the field of Deep Learning allows for more and more powerful algorithms that perform automatic segmentation. In particular, Artificial neural networks have demonstrated high-level performances and promising results in the Computer Vision realm. However, the performances of these systems are strongly dependent on the amount and the quality of available data. The data required by these algorithms need to be labeled and prepared beforehand by doctors. Therefore, it's crucial that the data provided to the Artificial Neural Networks is meaningful, not biased, correct, and various.

This is the main limitation of NNs and, in particular, in the medical field, there are more challenges like the privacy of patients, the scarcity of certain diseases, and the precision required for manual segmentation.

Our work falls in the context of the Total Marrow and Lymph node Irradiation (TMLI) which represents a more targeted form of radiotherapy compared with the standard of the past: Total Body Irradiation (TBI) having as targets the whole patient's body. The TBI causes late toxicities like growth impairment, neurocognitive decline and secondary malignancies but also represents a clinically significant concern for older patients (e.g., risk of lung damage) [29]. For these reasons, the use of total body irradiation as part of conditioning regimens for cancer patients are progressively declining. As opposed to TBI, the Total Marrow and Lymph node Irradiation in new conditioning regimens allows making the whole procedure less time-consuming, more streamlined, and easier to integrate into the clinical workflow. As a drawback, this new technique is dependent on its technological implementation and requires a complex planning phase in order to classify the region that will be targeted by radiotherapy.

1.1. Scope

In our work, we evaluate and compare the performances of different segmentation methods extracting organ-at-risks from CT Scan images. Firstly, we trained binary segmentation models, segmenting organs and clinical target volumes, and training organ-specific networks. Then we evaluate the performance in a transfer learning context from other datasets. We analyze the effect of different scenarios in the transfer learning setting, such as the possible data scarcity and the variable amount of frozen layers. Finally, we evaluate the performance of models used to segment multiple organs at once from the input CT Scan. In particular, we work on multiclass models able to segment multiple targets and ensemble method: architecture made out of single binary nets and a fusion layer that combine the various results in a single multiclass segmentation output. The main contributions of this work are:

- Evaluation of the segmentation performance of different transfer learning scenarios over the training from scratch.
- Evaluation and training of different networks over different organs in the body. Comparison of the single organ Approach over multiple organs.
- Evaluation of the efficacy of ensemble methods in the multiclass segmentation task.

1.2. Thesis structure

The rest of the thesis is structured as follows:

- Chapter 2: Here we provide the context of this work in order to give some knowledge to best understand the following chapters.
- Chapter 3: Here we introduce the CT-Scan images, the dataset used, and the pre-processing steps carried over the samples.
- Chapter 4: Here we put a detailed presentation of the Neural network model, his specifications (e.g. Loss function, optimizer, ...) and the evaluation metrics used to properly validate the results.
- Chapter 5: Here we present the various experiments carried out.
- Chapter 6: Here we list the results of the experiments.
- Chapter 7: Here we include a conclusion about the work that has been done, and suggest some possible future improvements.

2 | State of the Art and Theoretical background

In this chapter, we present the actual state of research in the field of medical image segmentation, here we put a knowledge base useful to understand the context and the next chapters. In particular, in Section 2.1 we introduce some theoretical background on which our work is based, namely, Neural Network, Convolution and Transfer Learning. After, in Section 2.2 we introduce the context of AI and deep learning, focusing on the state-of-the-art models and the main problems of the image segmentation task.

2.1. Theoretical Background

Here we present some technicalities required to fully understand the context in which this work fit.

2.1.1. Total Marrow and lymphoid irradiation

Total body irradiation has been developed more than 60 years ago, and it has been used in medical treatment to eradicate malignant cells from the bone marrow, lymph nodes, and blood. However, its usage is declining mainly because of concerns about toxicities and as a result of the introduction of more targeted intensity-modulated radiotherapy which enables more control over the radiation dose delivery. The new approach that emerged as one of the most promising topics for future research is Total Marrow and Lymphoid Irradiation (TMLI). In fact, the TMLI allows for precise delivery of radiation doses to complex-shaped organ targets while sparing normal tissue.

In this paper [29], the authors presented a study that aims at showing the effects of TMLI compared with Total Body Irradiation.

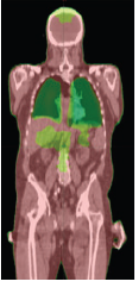
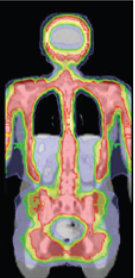
	Leukaemia eradication	Immuno-suppression	Engraftment	Treatment planning	Delivery	Late toxicity
TBI conditioning 	+++	+++	+++	+++	++	+
TMI or TMLI conditioning 	+++	+++	+++	+	++	+++

Figure 2.1: Comparison of TMLI vs TBI in terms of therapeutic effect, radiation planning, delivery, and toxicity profile.

Plus sign represents a 3-point rating scale, with three-plus signs indicating the best and one plus sign the worst. TBI=total body irradiation. TMI=total marrow irradiation. TMLI=total marrow and lymphoid irradiation.

A technological gap currently limits the widespread introduction of TMLI as an alternative to total body irradiation. Precise radiotherapy still represents a challenge, because of difficulties in target contouring and sophisticated planning. In this context, an automated mechanism to generate automatic target contouring is highly beneficial for clinical practice.

2.1.2. Artificial Neural Networks

Artificial Neural networks, ANN for short, are implementations of the idea that the only form of intelligence that we know is in the brain, so to simulate intelligence we need a machine that simulates the human brain.

How does the brain work? The modern knowledge of the brain is far from exhaustive and, also if it was the case, we don't build airplanes by reverse engineering feathers; we need underlying aerodynamics principles. There are anyway some physical processes

that we can monitor which give us some clues about the brain's mechanism. Donald Hebb, a Canadian psychologist, wrote in his 1949 book *The Organization of Behavior*: When an axon of cell A is near enough to cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased [14]. Or, to rephrase it simpler: neurons that fire together wire together[9]. The brain is an evolving machine that increases or decreases the connection between neurons, a single mental concept in this model is represented by a distributed group of neurons and their connections. This communication between neurons happens in parallel in different parts of the brain, so we would need highly parallel computation in order to make a powerful neural network. The number of transistors in the computer is catching up with the number of neurons in the human brain, but the number of connections between the brain's neurons is orders of magnitude bigger than the connection between transistors in a microprocessor; this is mainly due to the planar shape of the semiconductor technology used.

The core of the ANN is the concept of Perceptron developed by Rosenblatt: a perceptron is an emulator of a neuron, it takes different inputs, and it weights them differently. Every weighted input signal is summed and, after a certain threshold is passed, the perceptron fires a signal in the output.

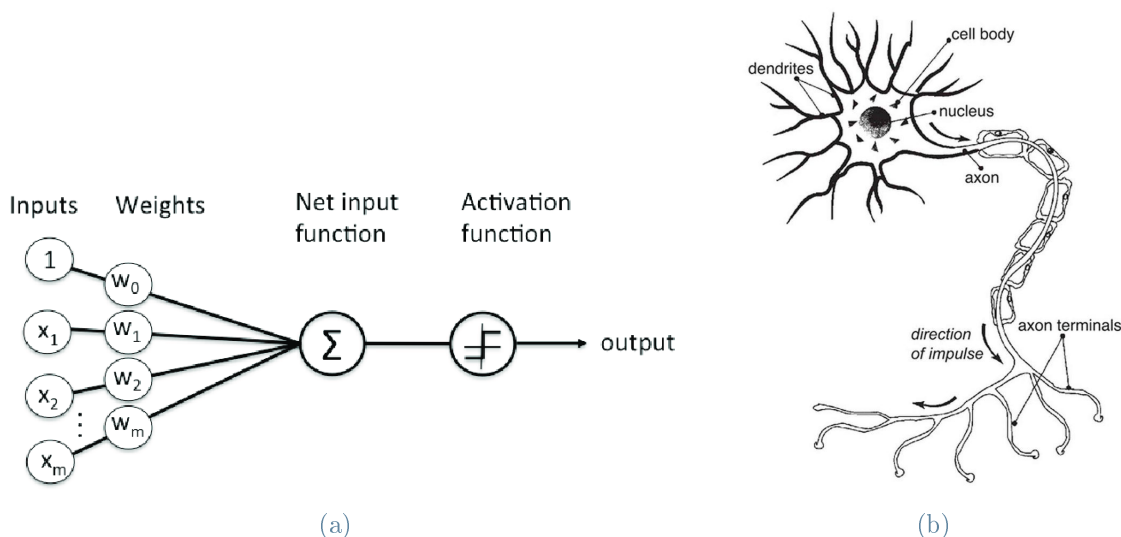


Figure 2.2: Artificial perceptron and biological neuron.

As far as we know, neurons in the brain work in the same way: they are connected with different strengths, and they can fire a signal through the axon giving an output of either 0 or 1. To build a neural network we create a network of neurons connected in different

ways. This led to networks with different shapes and different capabilities.

2.1.3. Stochastic Gradient Descent

The neural network can simulate the learning process by adjusting the weights that connect the neurons, it needs a dataset of couples $\langle \text{input}, \text{output} \rangle$. For every input, we measure the difference between the expected output (from the dataset) and the actual output computed by the net in an initialized random state. This difference is called Loss and the objective of the learning algorithm is to minimize it in order to have a network that produces an outcome as close as possible to the desired one. The learning algorithm is called Stochastic Gradient Descent, a Stochastic approximation of the Gradient Descent because it performs the algorithm on batches and takes into consideration the mean result over the batches for every iteration. The Gradient Descent is an optimization algorithm that aims at finding the minimum of a differentiable function (the loss function, in our case). It consists of an iterative process, in every iteration, it adjusts the weights by moving a step (in the weights multidimensional space) in the opposite direction with respect to the gradient of the function, in this way it is going in the direction of smaller loss values. The length of the step is called Learning rate. A drawback of this learning algorithm is that the loss function landscape has not only one minimum, but a lot of local minima, we should image the loss landscape like a mountain territory with a lot of tops and valleys (actually, the function landscape has not 3 dimensions like in the following image, but it's n-dimensional, with n equal to the number of adjustable parameters). While trying to find the optimal solution, the gradient descent will prefer directions that ensure a lower value of the function when it reaches a valley, the gradient descent will get stuck in that suboptimal state. To avoid this problem, learning processes incorporate an Optimizer which adds momentum and past information to the gradient descent's step and guarantee more exploration of the function's landscape. The problem of local minima endangers any certainty that gradient descent will find the best solution, because of this, we can reach different solutions based on where the start is placed. For this reason, is important to initialize the network weights in a way that allows the algorithm to find a good solution.

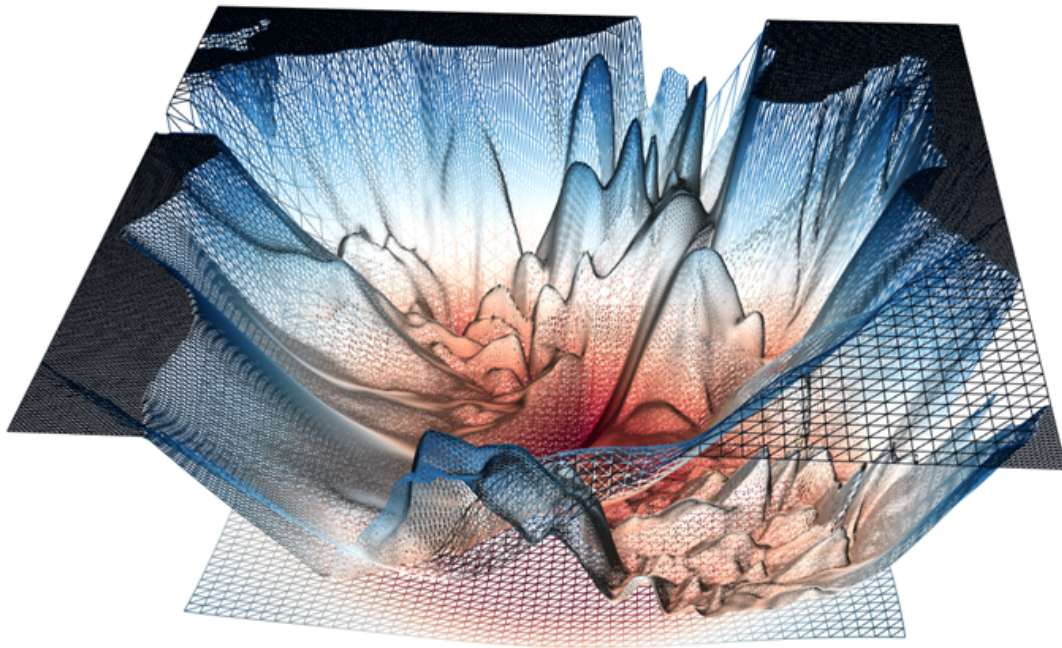


Figure 2.3: 3D Visualization of a Loss function landscape.

2.1.4. Convolutional Neural Network

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which are inspired by the organization of animal visual cortex and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNN is a type of network that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into a final output. Convolution is a specialized type of linear operation used for feature extraction, where a small matrix of weights, called a kernel, is applied across the input. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map. The convolution operation is therefore searching for patterns (stored as weights in the kernel) across the input image. When the image's portion in analysis is matching the structure stored in the weights, the element-wise product will produce a higher value, in this sense, the output of a convolution can be considered as a high-definition heatmap for the pattern used in the kernel. This procedure is repeated by applying multiple kernels to form an arbitrary number of feature maps, which represent

different characteristics of the input; different kernels can, thus, be considered as different feature extractors. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers. This grid is fed to the CNN which extracts features thanks to the convolutional layers. Then, the down-sampling operation is performed: here the goal is to reduce the spatial dimensions in order to learn position invariant features at different scales. In general, the down-sampling is performed by a pooling operator which takes a matrix of pixels from the input image and outputs a single value (in general the maximum). This operation is applied in all the pixels of the image and leads to an output with a lower resolution allowing the next layers to learn features at different scales. The convolutional block made out of convolution and down-sampling is then repeated multiple times during the encoding phase. Here every convolution increases the number of features learned while the pooling reduces the image size leading to lower-level features in the deepest layers. The decoder is in charge of reconstructing the original image resolution with up-sampling operations, which are, in general, transposed convolutions. [30].

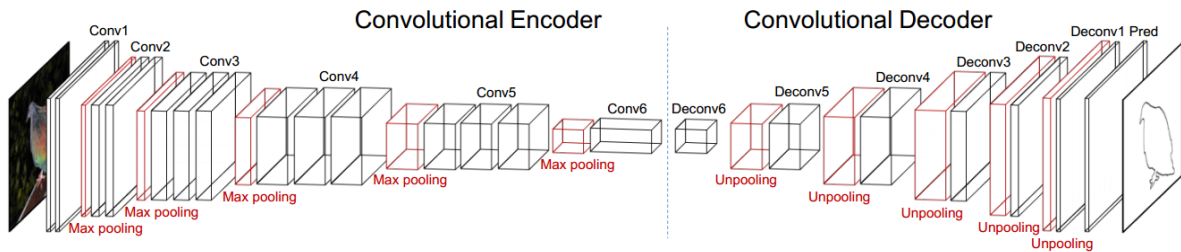


Figure 2.4: CNN composed of Encoder, bottleneck and Decoder.

CNNs are particularly useful with images because they can learn features independently of their position in the grid of pixels. Moreover, the number of weights used in the Convolutional block is by far less than the ones needed to train in a standard fully connected network allowing for faster training and a lighter model. As we can see in Figure 2.4 these convolutional blocks are repeated in a down-sampling path which extract features at different levels, then all these features go through a bottleneck and, finally, an up-sampling path maps the features learned to the final output. This is known as Encoder-Decoder architecture, and it's the state of the art in the task of image segmentation with CNNs.

2.1.5. Transfer learning

Many machine learning methods work well only under a common assumption: the training and test data are drawn from the same feature space and the same distribution. When the

distribution changes, most statistical models need to be rebuilt from scratch using newly collected training data. In many real-world applications, it is expensive or impossible to recollect the needed training data and rebuild the models. It would be nice to reduce the need and effort to recollect the training data. In such cases, knowledge transfer or transfer learning between task domains would be desirable [24]. As a definition, transfer learning aims to extract knowledge from one or more source tasks and applies the knowledge to a target task. We need to keep in mind three different aspects while deciding whether using Transfer Learning or not: what to transfer, how to transfer, and when to transfer. "What to transfer" asks which part of knowledge is generally between the source and destination tasks, some knowledge may be more transferrable than others. "How to transfer" asks about the practical way in which we perform the knowledge sharing. "When to transfer" asks in which situations transferring should be done. We can categorize three different sub-settings under the Transfer Learning category:

- Inductive transfer learning: the source task and target tasks are different (i.e. Segmentation and Classification).
- Transductive transfer learning: the source and target tasks are the same, while the source and target domains are different.
- Unsupervised transfer learning: here the source and targets task are different but related and the target is an unsupervised learning task.

As we will see, this work fall in the Transductive transfer learning sub-setting since we are trying to transfer knowledge from a different domain for the same task: image segmentation. In the specific case of Artificial Neural Networks, the knowledge is transferred directly as a trained model or part of it. Instead of training a model from scratch, we can take advantage of the knowledge from another domain embedded in a pretrained model. The learned features are transferred in a Network-based fashion, in order to do this, the source and target models need to be the same since the learned feature are technically represented by the weights of the network. After the features are transferred, we might need to adapt them a bit to the target domain, in order to do so we can train the network on some target data while using the transferred configuration as the initialization state of the model. As we have seen, the CNN models are extracting features at different levels, the deeper the network layer, the more task-specific are the feature extracted. As a consequence of this specialization of the deeper layers, transfer learning between CNN is typically done only for the more general layers while the deeper ones are trained from scratch. The layers that are only copied from the source network are called "frozen" because they will not change during subsequent training. However, there is no

clear distinction between a transferrable feature and a non-transferrable one.

In this study [31], the authors analyzed the performance of transfer learning based on the number of layers that are frozen. As suspected, the transfer from different domains became less effective as we increase the number of frozen layers, but surprisingly when trying to transfer knowledge over a target domain that is the same as the original domain, the performance drop if we froze only a subset of the layers, as you can see in the following figure.

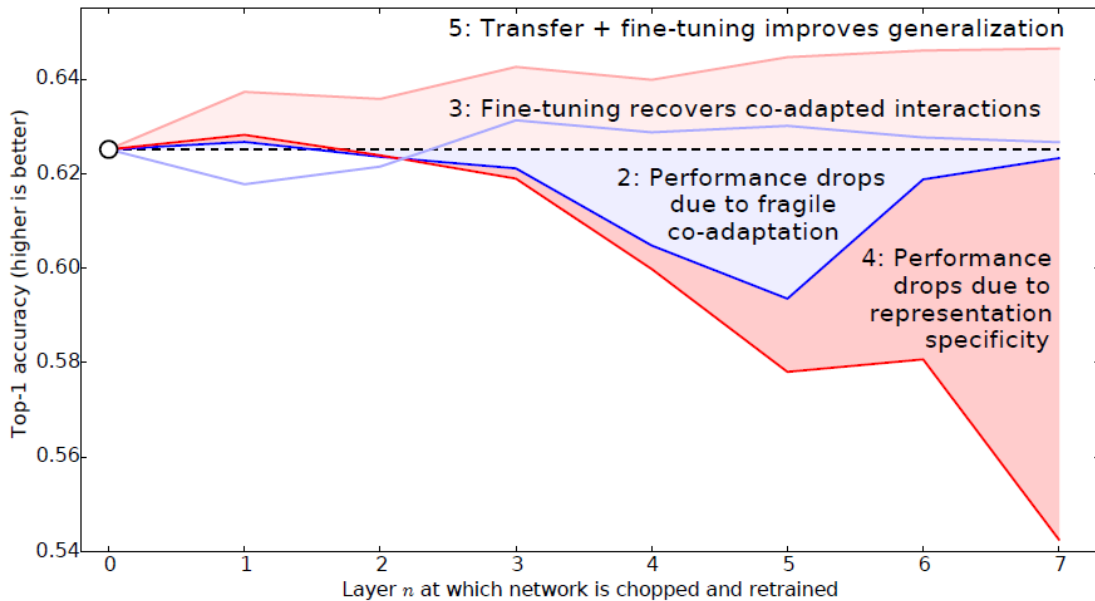


Figure 2.5: The blue lines represent transfer performance using the same source and target domain, red lines represent transfer over different domains. The light-colored lines are the result of transfer learning without any frozen layer, the dark-colored lines show the results of Transfer Learning with some frozen layers.

The explanation for this phenomenon is that some features interact with each other in a complex or fragile way, they are *co-adapted*. These linked features could lead to optimization difficulties if only some of them are frozen. And to avoid breaking them during the transfer, not a single layer should be frozen. In general, results show that initializing the network with transferred features is better than setting random weights, even if the distance between the source and destination tasks increases. Furthermore, when dealing with small datasets we can leverage transfer learning to improve generalization over a few samples and reach better inference results.

2.2. Related works

2.2.1. Artificial Intelligence and Deep learning

"Artificial intelligence" means the science and engineering of making intelligent machines. [22] An algorithm that is able to reproduce an intelligent behavior, is an AI. Recognizing objects' presence and location in an image is a task that only intelligent organisms can perform, so an algorithm doing the same task will be classified as an AI. This is a broad field of research, a subsection of this field is Machine Learning (ML): ML teaches a machine how to make inferences and decisions based on experience. ML involves the ability of an algorithm to learn and identify patterns in the data. This automation to reach conclusions by evaluating data saves human time for businesses and helps them make better decisions. We can think of ML as a way of speeding up the scientific method: it's a cyclic process of generating, testing, and discarding hypotheses; it automates discovery. In our work we focus on this type of ML algorithms that are able to learn from experience; in particular, we consider a subfield of ML: Deep Learning. DL groups a set of machine learning algorithms that are able to extract features from data at a different level of abstraction, they are able to extract "deep" features and gain a better representation of the input. DL is becoming more and more popular nowadays, in particular in Computer Vision tasks (tasks that emulate human vision). There are mainly three reasons contributing to their success: Firstly, the main reason behind the amazing success of deep learning over traditional machine learning models is the advancements in neural networks, they learn high-level features from data in an incremental manner, which eliminates the need of domain expertise and hard feature extraction, and they solve the problem in an end to end manner. Secondly, the appearance of GPU and GPU-computing libraries make the training of the model 10 to 30 times faster than on CPUs. And the open-source software packages provide efficient GPU implementations. Thirdly, publicly available datasets such as ImageNet, can be used for training, which allows researchers to train and test new variants of deep learning models. [33]

2.2.2. AI applied to medical image analysis

In the last two decades, AI has become integrated into some medical workflows, this integration has generated an area of research called Computer-aided detection (CAD). With the development of science and technology and the promotion of medical imaging applications, manual data interpretation, and analysis has gradually become a challenging task. Radiologists may misinterpret diseases because of inexperience or fatigue, leading

to missed diagnosis, that is, false-negative results, non-lesions may be interpreted as lesions, or benign lesions may be misinterpreted as malignant, that is, false-positive results. According to statistics, the misdiagnosis rate caused by humans in medical image analysis can reach 10-30%. In this background, the CAD system can be a great helpful tool for radiologists in medical image analysis. The workflow of a typical CAD system (shown in Figure 2.1) in medical image analysis can be divided into four steps: Image pre-processing, segmentation, feature extraction and selection, lesion classification. [12]

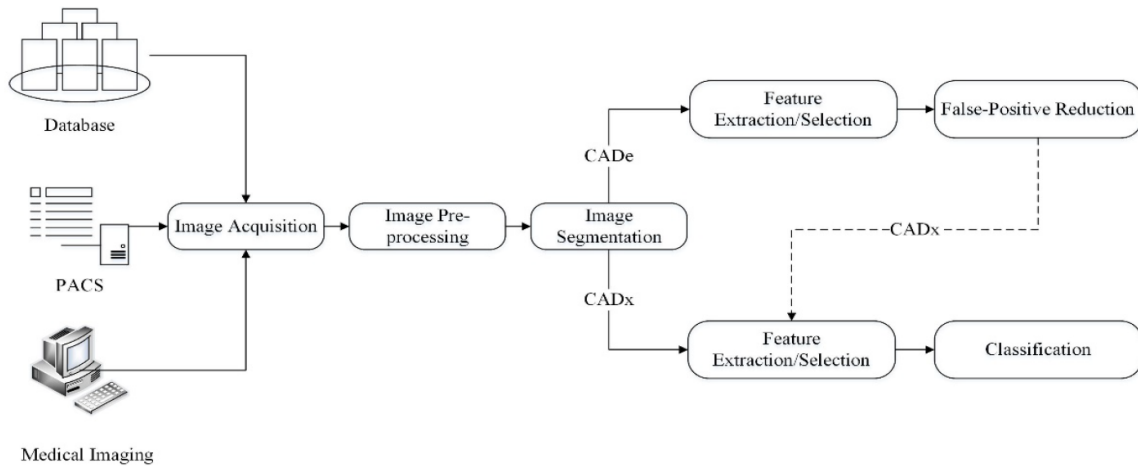


Figure 2.6: Workflow of a typical CAD system.

Our work falls into the Image Segmentation phase needed to develop a radiotherapy plan, we need to know where organs-at-risk are.

Cancer treatment by Radiation therapy consists of high-powered energy beams such as X-rays or protons are applied to the patient's body to kill cancer cells. It's important, during cancer treatment, to target only the cancer cells and avoid damage to other organs. Radiation therapy could lead to damage to the body, in particular, late side effects may first occur six or more months after radiation therapy is over. Late side effects may include infertility, joint problems, lymphedema, mouth problems, and secondary cancer. To target this issue, in the last decades, high precision radiation therapy such as intensity-modulated radiation therapy (IMRT), volumetric modulated radiation therapy (VMAT), and proton therapy have been widely used for cancer treatment due to their ability for highly conformal dose delivery. To minimize post-treatment complications, organs-at-risks (OARs), such as the spinal cord, eyes, optic nerves heart, and lungs, which must be accurately delineated. The complexity of OARs morphology and imperfection of imaging devices such as Magnetic Resonance Imaging or Computed Tomography, make manual delineation prone to errors and time-consuming. There is therefore a great demand for

more accurate OARs delineation and for a considerable reduction in the amount of manual labor in treatment planning. In this spirit, we propose an Artificial Neural Network able to automatically segment the OAR speeding up the treatment pipeline and lighting the medics' workload. [16]

Typically, deep learning methods can be divided into four categories: CNN-based methods, restricted Boltzmann machines (RBMs), or adversarial approach. The best results are obtained by the CNN-based methods which are also the current direction of research. CNN could be used in different domains, they have shown promising results in computer vision, different architectures have been proposed during the recent years to tackle different problems, but the convolutional building block is the same.

In the recent literature, CNNs have been widely used in the context of medical image segmentation. A lot of research focused on the brain and the segmentation of its various structures (White Matter, Gray matter, Cerebrospinal Fluid) [18]. The medical images used are in general Magnetic Resonance Images of various modalities (T1-weighted, T2-weighted) or CT Scan. From an input data perspective we can identify 3 main approaches:

- 2D: the input is divided into 2-dimensional slices and the network is fed with one slice at a time.
- 2.5D: the input is divided into 2-dimensional slices, but the network is fed with a few adjacent slices [26].
- 3D: the input is the whole 3-dimensional structure, and the network is fed with a small batch of 3D volumes [19].

A huge part of the research is concerned with the identification of lesions, for example, in this article [32], the authors developed a dilated convolution network to segment COVID-19 lesions from CT Scan. Other targets are the tumors: in [15], a segmentation process to identify brain tumors from multimodal MRI images is presented. Regarding the multiclass case, the research has been conducted with different approaches: in [10], the multiclass segmentation is performed by a single Unet working on 2.5D input and then the net is used as a generator in an adversarial setup. Instead, in [11], the segmentation process is split into a Region Of Interest extraction and, after, a binary network segmentation of the single structure.

2.3. Summary

In this section we provided an introduction to the segmentation problem addressed in the subsequent experiments, we put the focus on the need for this technology from a medical perspective; then we dive more into the details of how the AI and the Neural Networks actually work. We also addressed the topic of Transfer learning which will be used too in the subsequent experiments. Finally, we presented some related works in the field of Image segmentation in the medical realm.

3 | Dataset

In this chapter, we give some information regarding the datasets used for the various network training and fine-tuning; in particular, we focus firstly on what a CT Scan is, then we move on to describing the structure and some technicalities about the datasets, and finally, we describe the preprocessing steps applied before feeding the data to the various networks.

3.1. Computerized Tomography

Computerized Tomography is an imaging technique used to visualize nearly every part of the body, it is used by medics to diagnose diseases, analyze injuries, or plan surgical, or radiation treatment. More technically, CT refers to a computerized x-ray imaging procedure in which a narrow beam of x-rays is aimed at a patient and quickly rotated around the body, producing signals that are processed by the machine's computer to generate cross-sectional images—or “slices”—of the body. These slices are called tomographic images and contain more detailed information than conventional x-rays. Once the machine's computer collects a number of successive slices, they can be digitally “stacked” together to form a three-dimensional image of the patient that allows for easier identification and location of basic structures as well as possible tumors or abnormalities. Each time the x-ray source completes one full rotation, the CT computer uses mathematical techniques to construct a 2D image slice of the patient which can represent a thickness from 1 to 10 millimeters. When a full slice is completed, the image is stored and the motorized bed is moved forward incrementally into the x-ray source. The x-ray scanning process is then repeated to produce another image slice. This process continues until the desired number of slices is collected [2]. In our target dataset, the slice thickness is 5 millimeters and the slices cover all the upper body, including the head, chest, abdomen, and the beginning of the legs.

3.2. Dataset Technicalities

During the carried experiments, we used pretrained models, the publicly available source datasets used to pretrain those models were StructSeg and SegTHOR, instead, the private target dataset used to carry experiment is referred as AUTOMI:

- Structseg 2019: from the "Automatic Structure Segmentation for Radiotherapy Planning Challenge", part of the MICCAI 2019.
- SegTHOR 2019: "IEEE international Symposium on Biomedical Imaging 2019".
- Target Dataset - AUTOMI: provided by the Humanitas Research Hospital.

3.2.1. StructSeg

The StructSeg dataset contains the source CT Scans for 50 patients and the corresponding segmentation masks manually delineated by doctors for 6 OARs (Heart, Right lung, Left lung, trachea, esophagus, marrow). Both the images and masks have 512x512 voxels in every slice.

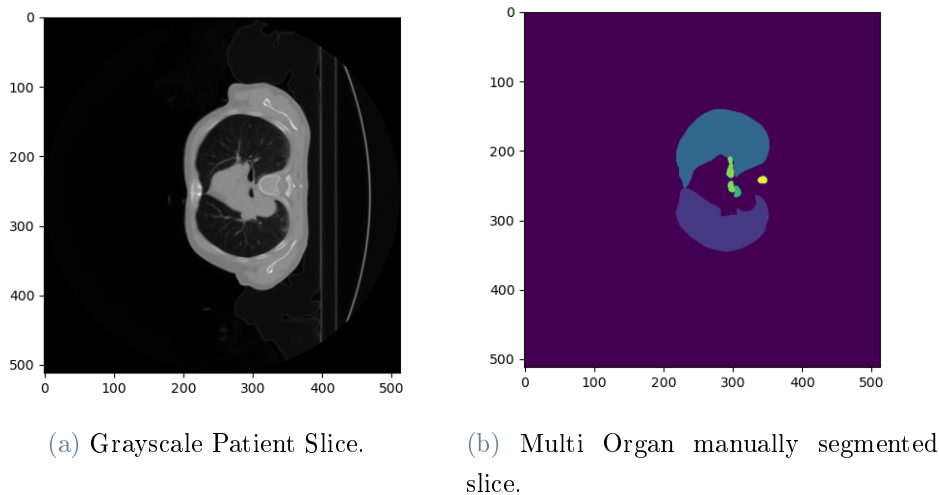
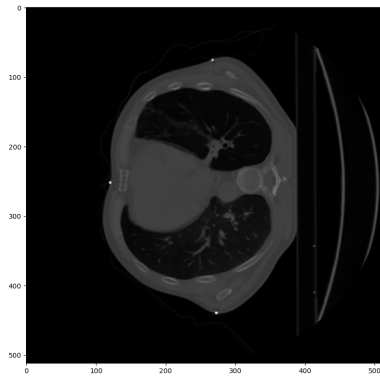


Figure 3.1: An example of slice and mask extracted from a random patient from the Structseg dataset

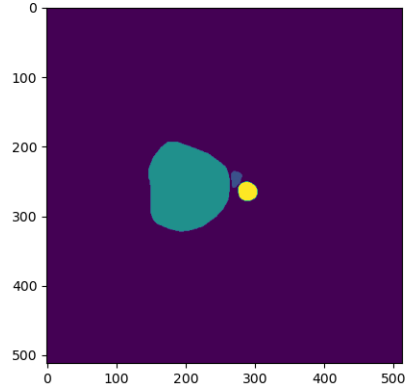
3.2.2. SegTHOR

The SegTHOR dataset contains the source CT Scans for 40 patients and the corresponding segmentation masks manually delineated by doctors for 3 OARs (heart, aorta, and

trachea) and 20 non-annotated patients. Both the images and masks have 512x512 voxels in every slice.



(a) Grayscale Patient Slice.

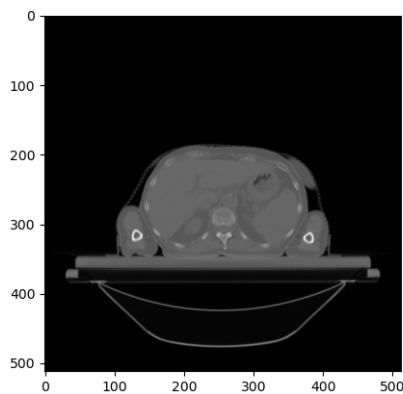


(b) Multi Organ manually segmented slice.

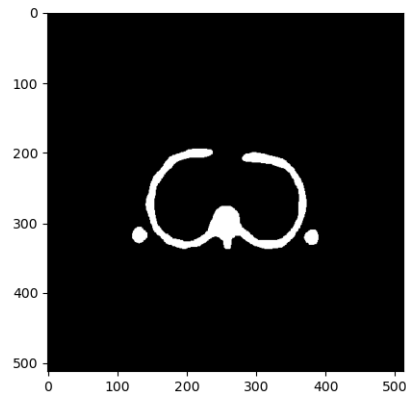
Figure 3.2: An example of slice and mask extracted from a random patient from the SegTHOR dataset

3.2.3. Target Dataset

The target dataset contains the source CT Scans of 100 patients and the corresponding segmentation masks manually delineated by doctors of different OARs and targets.



(a) Grayscale Patient Slice.



(b) Single Organ manually segmented slice.

Figure 3.3: The corresponding image and label from the Target dataset, in this example the label 'bones' is shown.

In particular, this dataset doesn't have a complete set of segmented organs for each patient, instead, every patient has a subset of OARs delineated, so every organ considered has a variable corresponding amount of data present in the dataset. A complete table of organs and the number of associated patients is present in the appendix A. The images have 512x512 voxels in every slice, the total number of slice for a patient is variable, the images are rotated 90 degrees clockwise with respect to the previous two Datasets. Differently from the previous datasets, here the labels are separated by target, so we will find a single binary label for each of them, instead of the multichannel labels seen previously.

Apart from labels containing OAR, in this dataset we are provided with other labels marked as *PTV*, these represent the target area decided by the medics, and the zones addressed with these labels will be targeted during the actual radiotherapy. In Particular, the label *PTV Total* is the union of all the other *PTV* labels.

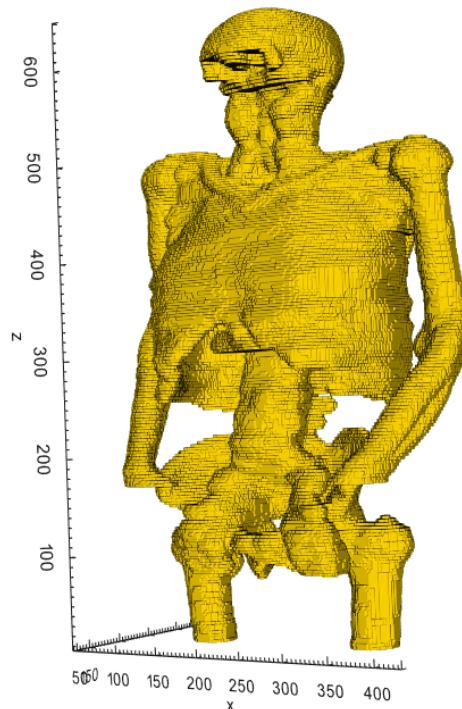


Figure 3.4: 3D model of a random patient showing the *PTV Total* Label representing the patient areas that will be exposed to radiotherapy.

3.2.4. Intensity Analysis

In order to make the transfer learning as effective as possible between datasets, we needed to check the differences between them. In particular, we studied the distribution of intensity values between the dataset's images. The intensity values of the CT Scan images are mapped to colors between black and white while displaying them, but the absolute intensity values are not visible from the image. These values are being used as input for the neural network, is, therefore, important to find eventual differences in the values' distribution between datasets since the convolutional kernels will be based on these values for near pixels.

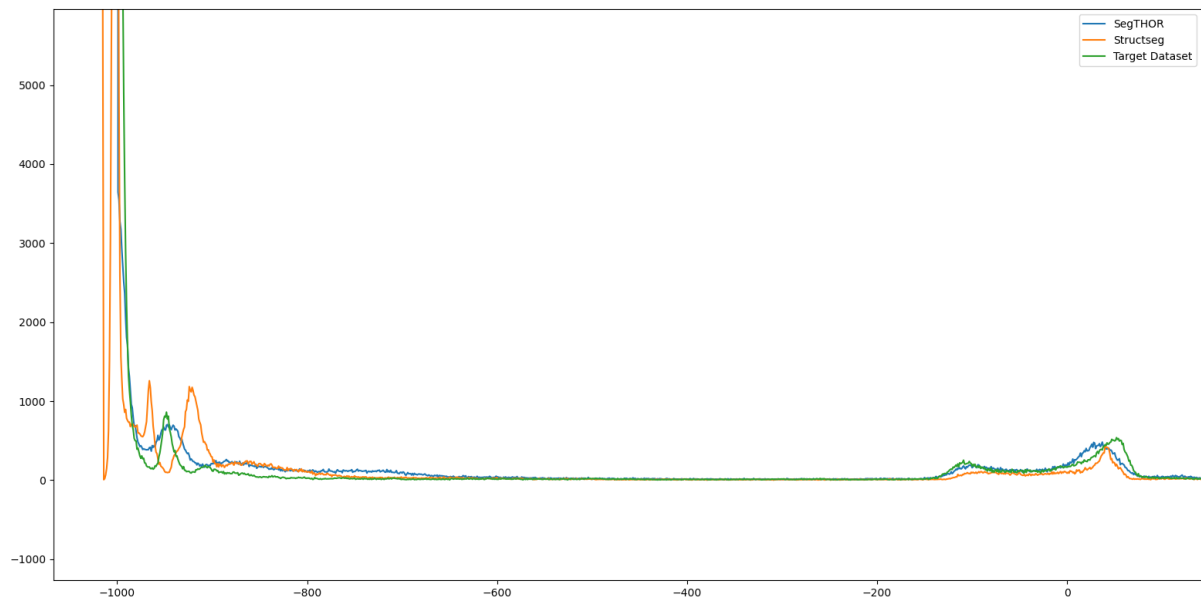


Figure 3.5: Graph showing intensity values mapped to their frequency in the dataset's images. Here the intensity values of the Target dataset are shifted to the left of 1000 integers for visualization purpose. For the same reason, the graph has been cut removing the higher frequency of the darkest values.

As we can see in the graph, the distributions of intensities are mostly the same, apart from an additive constant of 1000 integers present in the Target Dataset (which is removed in the graph). The effect of this constant shift will be easily removed thanks to the normalization step presented in the next section. Thanks to this similarity we can be confident about the transferability of features learned in a source dataset and used in the

target dataset.

3.3. Preprocessing

In order to optimize the effectiveness of a neural network's training, a proper clean-up of the data is needed. In this section, we present the different operations carried out over the dataset's images before being fed to the ANN. This work is particularly important in the case of transfer learning between datasets, in order for it to be as effective as possible we need to minimize the difference between the ANN's input data in the various scenarios.

3.3.1. Normalization

Among the best practices for training a Neural Network is to normalize your data to obtain a mean close to 0. Normalizing the data generally speeds up learning and leads to faster convergence.

Normalizing the intensities of the input image is also an effective way of removing the differences from different inputs. In particular, the normalization operation used in our experiments shifts all the intensity values in order to make them fit into the $[0, 1]$ set.

3.3.2. Cropping

As per the normalization, we introduce the cropping operation in order to feed the networks with data samples that are as similar as possible to the source task. In particular: some networks pretrained in the source datasets (SegThor and StructSeg), use a cropped image, since the interested region occupies only the central part of the input. In order to be consistent, we apply the same transformation for the same OARs. In particular, starting from the original dimension of a slice (512x512p in every dataset) we keep only the central square of 320x320p when dealing with smaller OARs.

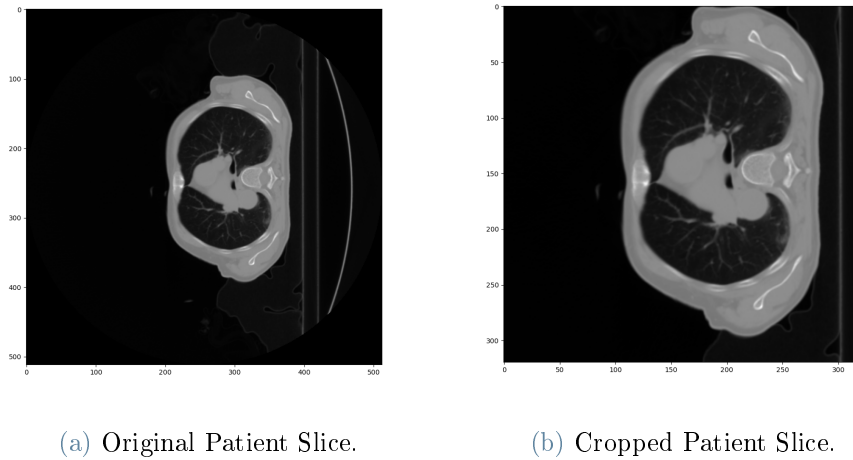


Figure 3.6: Example of original and cropped slice.

3.3.3. Rotate

The convolution operation is based on a filter, which operates over a matrix of pixels, if the image is rotated the values learned by filters are not useful anymore; in fact, CNNs are not sensitive to the position of the object to be segmented, but they are sensible with respect to the rotation. To address this problem, we rotate our dataset samples when dealing with pretrained model, in order to have an input data-oriented in the same way as the source datasets.

3.4. Data Augmentation

Data Augmentation is a popular technique used to increase the available data and increase the generalizability of the model. In practice, this is achieved by creating modified copies of data and adding them to the dataset. It's important to modify the data properly: if we just copy the data, this will not improve the model performances, but while modifying it we should be careful and generate data samples that are meaningful and realistic. In order to be as close as possible to the work done on the pretrained models, we choose the same augmentation techniques applied there. The transformations are applied online with a 50% probability each. More details about the specific transformations are provided in the next sections.

3.4.1. Elastic Transformation

Elastic transformation is a type of non-rigid transformation that deforms the image's shapes (Simply put, non-rigid transformations don't preserve the side lengths and angles in a shape). Technically this is achieved by creating a grid over the image, then a random displacement is applied over each grid intersection point, finally, the grid is interpolated to compute the displacement for every pixel in the image.

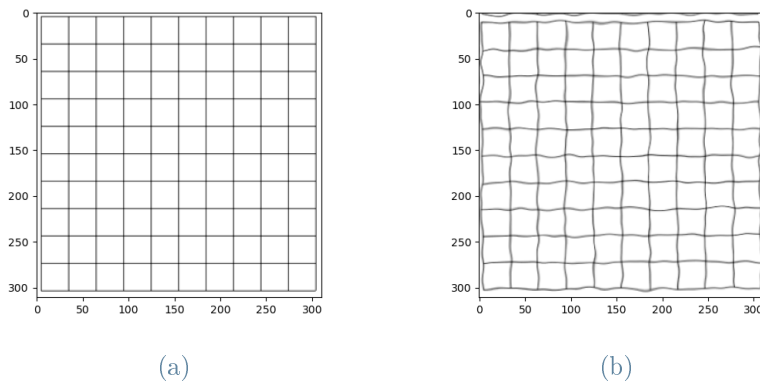


Figure 3.7: Example of original and elastic deformed image.

3.4.2. Grid Distortion

Grid distortion is a type of non-rigid transformation which deforms objects along a dimension in the image. This transformation applies a grid over the image and then randomly changes the dimension (horizontal or vertical) of the grid cells.

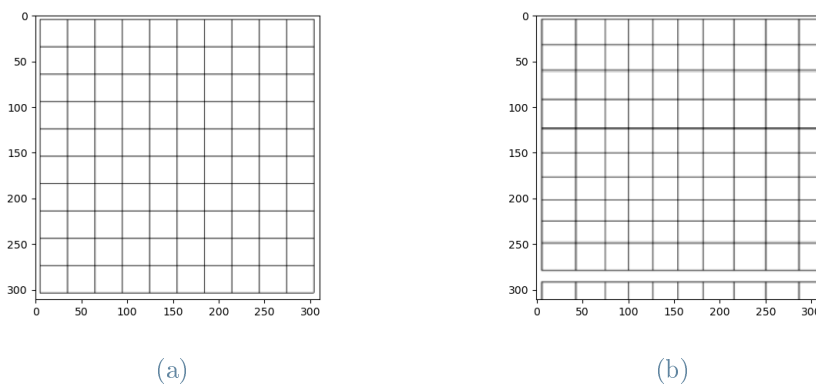


Figure 3.8: Example of original and grid distorted image.

3.4.3. Rotation

Rotation augmentation generates copies of the same image with a specified angle of rotation, this is a type of rigid transformation that doesn't change the shapes in the image. Differently from the preprocessing phase, here the angle is tiny, and it has the purpose of introducing possible misalignment of the patient position while taking the CT Scan.

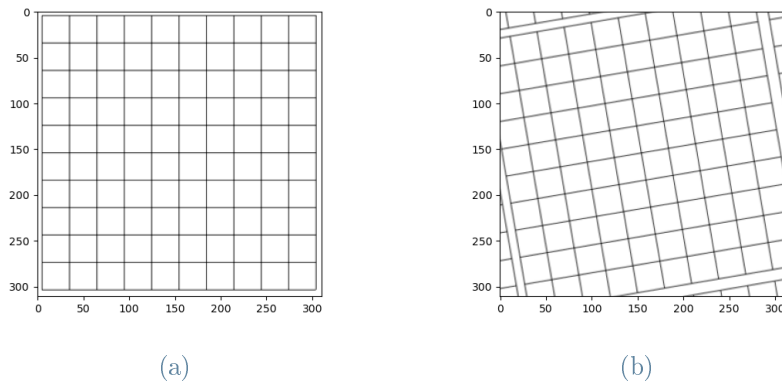


Figure 3.9: Example of original and rotated image.

3.5. Summary

In this chapter, we have introduced the imaging technique used by the doctors, then shared some details regarding the datasets used for the experimental analysis, finally, we have shown the preprocessing steps carried out over the input data before feeding it into the neural network. These steps are in common with all the experiments presented in the next chapters, apart from some exceptions that will be detailed later.

4 | System design and overview

In this chapter, we give a detailed description of the input pipeline which preprocess and feed the data to the various nets, we then show the neural network's architectures, after, we detail the process of refinements used to increase segmentation performances in smaller OARs. In later sections, we introduce an Ensemble method used in the source Datasets that will be replicated in order to perform fine-tuning in the target dataset. Finally, we present technical details regarding the execution of the experiment: the training of the network and the evaluation of the results.

4.1. Input Pipeline

The files loaded from the target dataset are in the DICOM (Digital Imaging and COmmunications in Medicine) format. This format contains some useful information regarding the patients and other technicalities related to the CT Scan and the acquisition method. A complete example of the data contained in a DICOM file could be found in Appendix B. Here we noticed that the patient's slice orientation was not always the same (also if the *Image Orientation* attribute wasn't changing) while the manually created labels had always the same orientation, in order to tackle this problem we created a JSON file associating every patient with a direction. Every slice image is then loaded considering the direction, then the images slices are ordered over the axial plane from the head to the legs.

The image data is contained in the *Pixel Data* as an array inside the DICOM file, and thus can be easily used programmatically. In the next sections we refer to two types of experiments:

- Training From Scratch: experiments done without transfer learning, here the Neural Networks are initialized randomly and trained over the target dataset.
- Fine Tuning: experiments done with transfer learning, here the Neural Networks are pretrained over the source datasets (StructSeg and SegTHOR); the pretrained state is loaded in the network which is then fine-tuned in the target dataset.

4.1.1. Training from scratch

When training a network from scratch, it is important to initialize its weights in the proper way, in doing so we allow the model to learn and explore the loss landscape. If weights are too big, the gradient will continue to grow while computing it in all the layers, in the opposite case, with too small weights, the gradient will vanish, and the network won't be able to adjust his weights and, thus, it won't learn anything. Every part of the network is different, but when it comes to Convolutional Layer, the technique used often is the so-called *Xavier initialization* which sets the weights to values extracted from a uniform distribution in the range:

$$\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}} \quad (4.1)$$

where n_i refers to the number of input of the layer, while n_{i+1} refers to the number of output of the layer. The data is preprocessed using only Normalization (3.3.1) and, in case the organ is tiny with respect to the whole slice, also Cropping (3.3.2) is applied. Then the data augmentation step (3.4) is executed, and the image is fed to the initialized network.

4.1.2. Fine-tuning

When fine-tuning the network, the initialized state is transferred directly in the model, and it is used as the initial state. The data is preprocessed as described in section 3.3 including all the operations: normalization, cropping (if applied in the pretrained net), and rotation of 90 degrees to remove the discrepancy between datasets. After this step, the usual data augmentation generation is carried over the image which is then fed to the network.

4.2. Networks Used

After having detailed the data loading and preprocessing steps, we move on to the actual Neural Networks used in the experiments. These base networks are the same as the ones pretrained on the source datasets:

- Unet, from [23],
- SE-ResUnet, from [7],
- DeepLabV3, from [8].

4.2.1. Unet

The Unet is a popular NN architecture used all over the literature for segmentation tasks. The popularity of the net derives from its versatility and simplicity; the network has a 'U' shape, it's composed of a convolutional encoder that works on 4 levels of down-sampling, then a bottleneck layer transfers the encoded data to the decoder which has 4 levels of up-sampling needed to reconstruct the original image dimensions. In particular, in the encoder, each convolutional block is made of 2 convolution operations in 3x3 kernels each followed by a Batch Normalization block and a ReLU activation function, then a block of down-sampling operators (MaxPooling) of factor 2 is applied. In the decoder, the same structure is repeated in the reverse, instead of the down-sampling operations, the up-sampling is carried out using Transposed convolutions. Additionally, the Unet has some direct connections from the decoder to the encoder, these connections are just copies of features taken from the encoder and concatenated to the decoder. Thanks to these connections, the information that is inevitably lost in the encoding operation is easily recovered during the decoding phase. At the end of the net, a 1x1 convolution is applied to map the final features to the desired number of classes predicted.

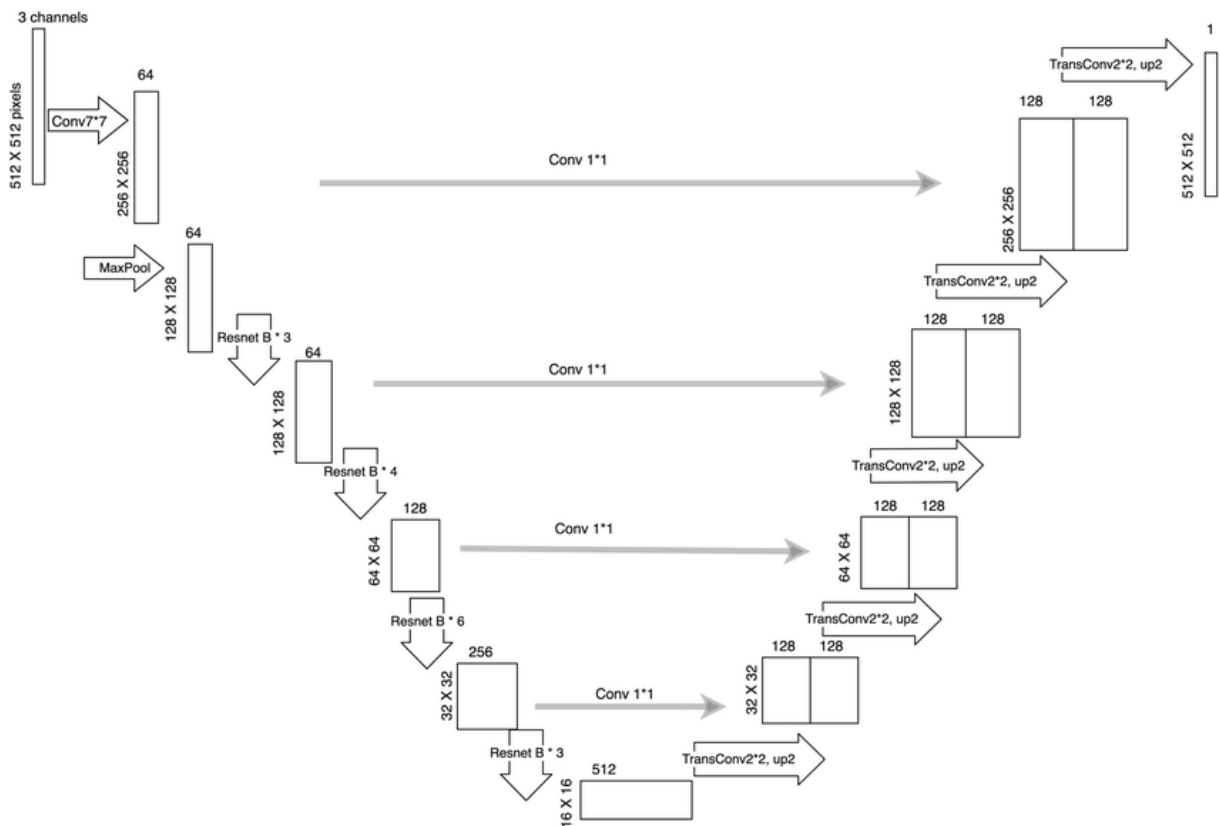


Figure 4.1: High level architecture of the Unet Neural Network

4.2.2. SE-ResUnet

SE-ResUnet is another encoder-decoder type architecture, it is similar to the Unet, it has the same number of down-sampling and up-sampling, and it employs the connecting paths from the encoder to the decoder. The main difference from the Unet is in the basic building block which in this case is called SE-ResBlock. This building block is made of 2 Convolutional Blocks at the beginning, then a Squeeze and Excitation block is applied. The Squeeze and Excitation block has the purpose of weighting differently every feature channel when creating the output feature maps. In order to compute the weight of the channel, the SE block performs a Global Average Pooling operation that averages the channel information in a single value, then Fully Connected layers and ReLU are executed to add non-linearity to the block. Finally, we weight every input channel with the computed weights. From a higher perspective, into the SE-ResBlock we can notice a flow that connects the input, and, skipping the other operations, it's appended to the output. The purpose of this skip connection is to avoid the problem of vanishing gradient, at least in the shortest paths, leading to easier optimization processes.

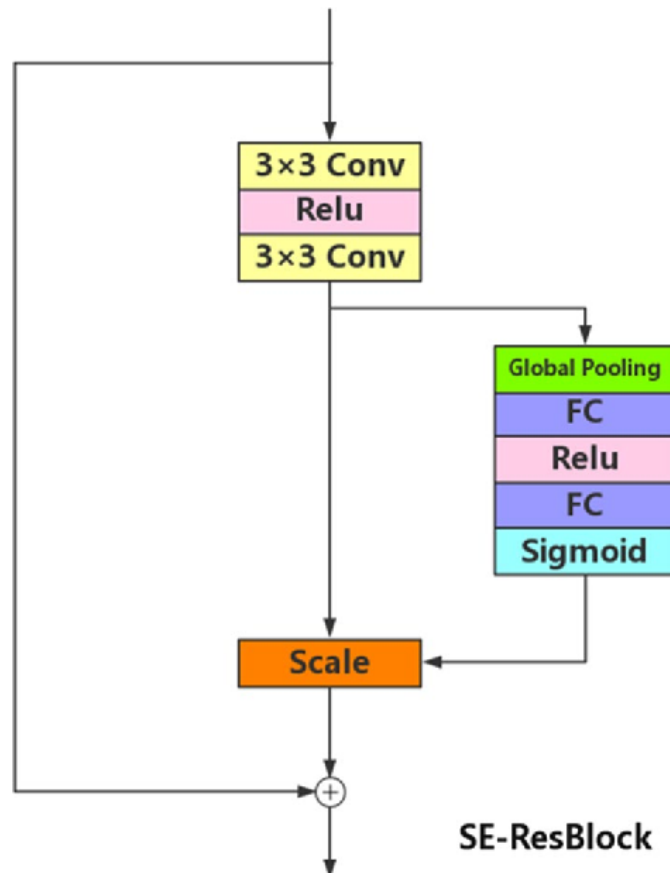


Figure 4.2: SE-ResBlock structure

4.2.3. DeepLabV3

The last base network used in our work is the DeepLabV3, created by Google in 2016. The generic structure of the net is the same as before: it has an encoder, a bottleneck, and a decoder. The main problems addressed by the development of this network are:

- reduced feature resolution: the features learned by CNNs are in general at low resolution because of pooling operations or other sub-sampling processes, on the other hand, this sub-sampling is needed to learn increasingly abstract representations;
- object at multiple scales: it often happens that the objects have different dimensions into the input image.

In order to address these problems, the Atrous Spatial Pyramid Pooling (ASPP) block has been developed. The ASPP block is based on Atrous convolution which, differently from the standard convolution, creates a bigger output using an *atrous rate* r , it inserts $r-1$ zeroes between two filter values along each spatial dimension (the French word Atrous means holes in English). This type of convolution allows us to adaptively modify the filter's field-of-view by changing the rate value. During the down-sampling phase, in DeepLabV3, the authors start using a cascade of atrous convolutions after a few normal down-sample, in this way the net extract features at different levels without decreasing too much the image resolution. As a basic block, the net uses the ResNet block, composed of three convolutions (kernel size of 3 x 3) where the last convolution also performs down-sampling with a stride of 2. The ASPP block incorporates multiple atrous convolutions at different levels, executed in parallel. This block is placed at the end of the decoder and allows for fewer down-sampling operations (2 in the net). The ASPP layer is also fed with context information computed with a global average pooling on the previous feature map of the model. Then the decoder is composed of standard up-sampling blocks, but, differently from the previous nets, only 2 up-samples are needed to reconstruct the original size.

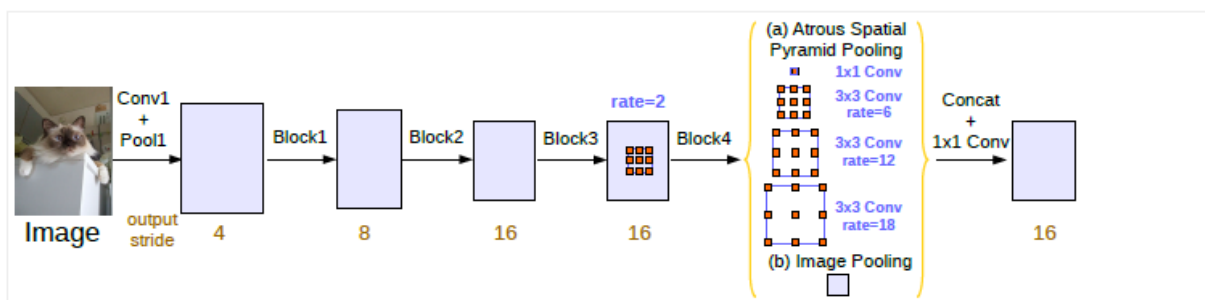


Figure 4.3: DeepLabV3 encoder structure

4.3. Refinements of Small Organs

In the context of the experiments *From Scratch*, we developed a specific training process for the binary segmentation of organs that occupy a tiny region of the CT Scan slice (from now on we refer to them as *Small Organs*). These targets occupy few pixels in the image, and they are a cause of imbalance between the classes (one class representing the organ and the other representing the background). For this reason, the segmentation performances are lower on these organs. To tackle this problem we designed a specific process that is able to refine the predictions and obtain better results.

4.3.1. Solution Architecture

Inspired by the work done in the literature designing the FocusNetv2 architecture [13], while dealing with the training and inference processes of the segmentation of Small Organs we perform two main steps: a coarse feature extraction and a refinement segmentation. In order to obtain better performances and avoid displacements of the Small Organs we turned off the rotation and grid distortion operations in the data augmentation step. The input slices and labels are cropped as described in 3.3.2 reaching the resolution 320x320. Then, the slices are fed to the first NN called *Coarse Net*: this network is a Unet that outputs a coarse segmentation of the small organ. After this first step, a smaller area of the input image (containing the small organ in question) is cropped by a *Window retrieval* component. The small window is stacked to the raw output of the first network, in this way we are incorporating the context from the bigger input slice into the small-sized image. The small window with context is then used as input for a second Neural Network called *Refined Net* which is again a Unet architecture trained on smaller input sizes. The output of this second network is the actual segmentation result of the small organ.

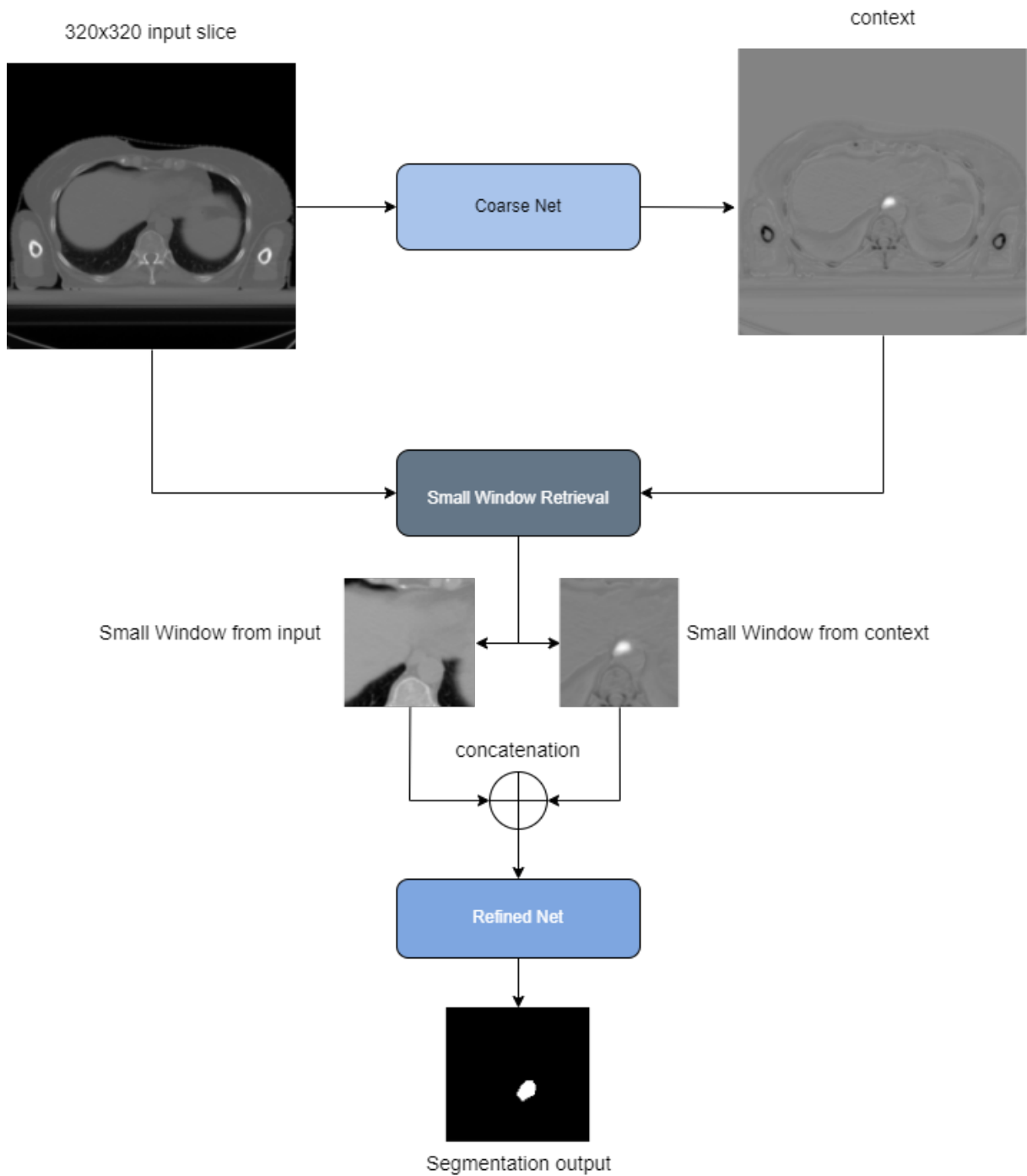


Figure 4.4: Small organ inference architecture

During inference, the whole network is working in cascade, first, the coarse net is applied to retrieve the context information, then the small window is retrieved from the input and fed to the Refined Net together with the context. While training, the two nets are separated: first the Coarse Net is trained over the 320x320 slice, then the Coarse Net is

trained using the first net in inference mode and the extracted slice of smaller dimension as input.

4.3.2. Small Window retrieval

In order to always feed the Refined Net with the right input, a Small Window retrieval component has been developed. The component crop both the input slice and the context tensor in two possible modalities:

- Hard-coded
- Smart Window

In Hard-coded mode, the central position of the Small Window to be cropped is retrieved from a configuration file as well as the Small Window dimension. The configuration file contains the central positions of the Small Organs and the window side dimension, both of which have been visually validated and tuned in order to always include the whole organ and enough context.

In Smart Window mode, the small window retrieval component uses the coarse prediction present in the context to find the window position, while the window dimension is decided as an input parameter. In order to find the best window position, we apply a sigmoid to the context image and then a 2D convolution. Using a kernel size equal to the size of the window to be retrieved and the constant weights having all the values 1, we are creating the image of the window to be retrieved.

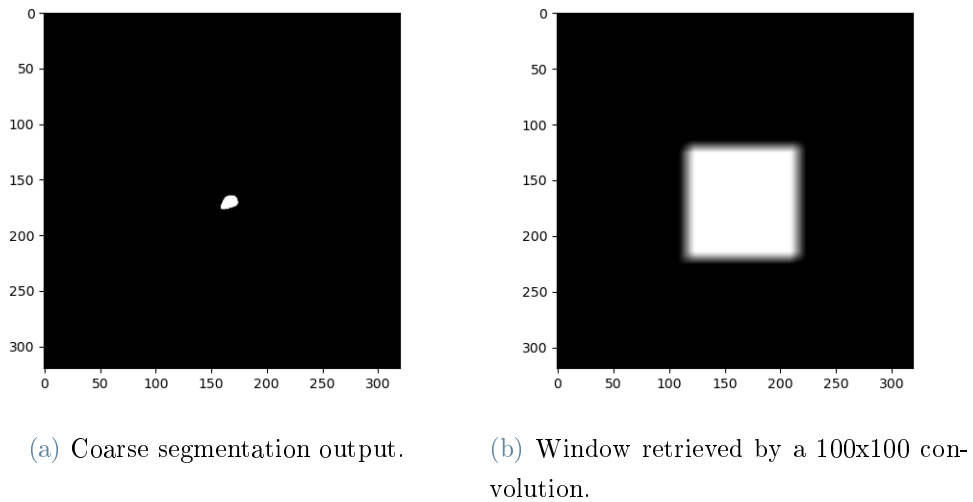


Figure 4.5: Example of window retrieval starting from the coarse segmentation result of an esophagus.

A final sigmoid layer is applied to push the intensity values to the extremes 0 and 1. Then the coordinates are extracted using the intensity values of the result; finally, the coordinates are used to crop the original image as well as the context, and both are fed to the refined net.

4.4. Ensemble methods

In order to perform multi-organ segmentation, we replicate the ensemble method used in [25], in particular, we implement the simplest one: argmax; and the best performing one: Last Layer Feature Fusion.

We replicate the method Argmax as a baseline of results for the multi-organ segmentation using ensemble methods. In order to merge the results of the single binary networks, this method simply takes the maximum value between the networks' output, pixel by pixel. In this way, every region is segmented by the network having the higher output. Notice also that, in this context, no train nor transfer is needed.

The ensemble methods allow using multiple binary networks trained over specific organs and merging their results to create the final multi-class segmentation. In order to better exploit the results of the single network, we don't only concatenate the results, but we tune a NN that is going to weight and extract the most useful features from the binary networks. In particular, the last layer feature fusion method combines the last layers of

features of the different binary models and fed them into a 1×1 convolution which is trained to learn which feature weight more and outputs a single dimension array for every pixel: the i -th element in the array represents the likelihood of the pixel being in the i -th class. In our case, the last layer is different based on the binary network used: while using SE-ResUnet or Unet, it is the last convolutional block before the final 1×1 convolution; If we use a DeepLabV3 architecture instead, the last layer is composed of the last convolutional block before the final 1×1 convolution together with the last up-sampling operation.

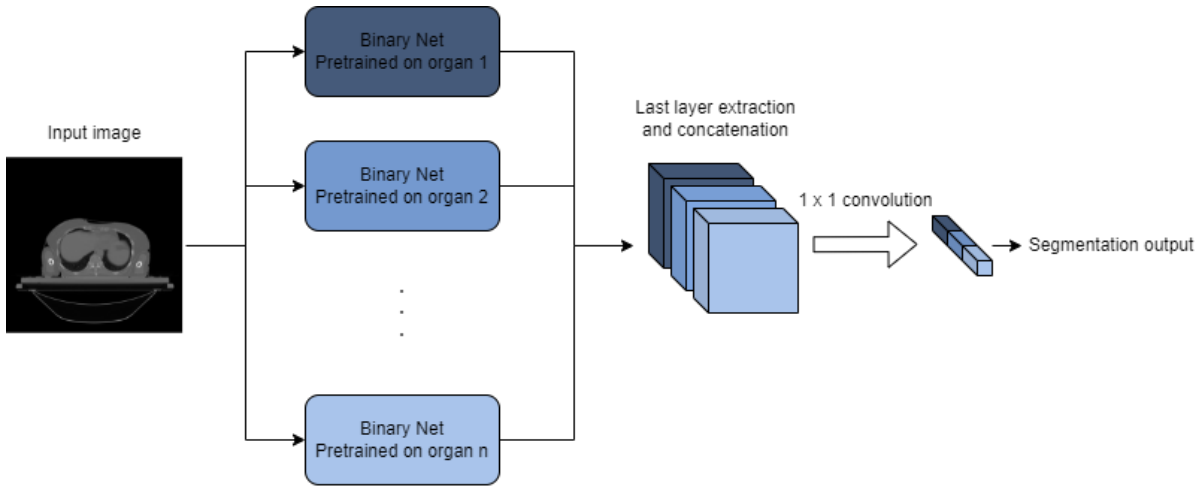


Figure 4.6: Last Layer Feature Fusion

In detail: the number of features taken from each Unet or SE-ResUnet is 64, and 256 for every DeepLabV3.

4.5. Training

In this section we present the training of the various neural networks, we first lay out details regarding the Loss function used to drive the training, then we move on to the evolution of the learning rate parameter during the various epochs, finally, we add some finer technical details of the process.

4.5.1. Loss

The Loss function has the purpose of measuring the difference between the expected results and the actual ones. While working on images it's important to choose the most useful loss function. If we just compute the number of right-guessed pixels we will end up with excellent results for the wrong prediction in case tiny organs are being segmented. In order to take into account also the dimension of the prediction, we need more complex

functions. We should first make a distinction between different types of loss:

- Distribution-based losses: functions that aims at minimize dissimilarity between two distribution (e.g. Cross-entropy);
- Region-based losses: functions that aims at minimize mismatch between ground truth and predictions (e.g. Dice loss);
- Boundary-based losses: functions that aims at minimize the distance between ground truth and predicted segmentation (e.g, Hausdorff Distance).

For our segmentation tasks, we decided to use Region-Based losses, since the qualitative indicator of quality is the similarity between ground truth and prediction shape and position. Moreover, when the level of class imbalance increases, loss functions based on overlap measures are more robust. In particular, while working on binary segmentation, we used the Dice Loss, which is directly optimizing the Dice similarity coefficient used in validation and testing. The Dice Loss has the following definition [20]:

$$1 - \frac{2 \sum_{c=1}^C \sum_{i=1}^N g_i^c s_i^c}{\sum_{c=1}^C \sum_{i=1}^N g_i^c + \sum_{c=1}^C \sum_{i=1}^N s_i^c} \quad (4.2)$$

Where g_i^c is the ground truth binary indicator of class c of pixel i , s_i^c is the corresponding predicted segmentation class; C represents the set containing all the classes and N represent the total number of pixel. The loss' second term is the dice coefficient: it computes the intersection between the ground truth and the prediction, multiply it by two and divide by the total number of pixels (ground truth + prediction).

In the case of multi-class segmentation, we adopt the extension of the dice loss for multiple classes, the Generalised Dice Loss [28]:

$$1 - \frac{2 \sum_{c=1}^C w^c \sum_{i=1}^N g_i^c s_i^c}{\sum_{c=1}^C w^c \sum_{i=1}^N g_i^c + s_i^c} \quad (4.3)$$

$$w^c = \frac{1}{(\sum_{i=1}^N g_i^c)^2} \quad (4.4)$$

The formulation is similar to the previous one, the main difference being the term w^c which is used to provide invariance to the different classes' properties. This weighting factor has the purpose of correcting the contribution of each label by dividing by its volume squared.

4.5.2. Learning Rate Scheduler

During training, the most important hyperparameter to be tuned is the learning rate; adaptive learning rate methods demonstrate better performance than learning rate schedules, and they require much less effort in hyperparameter settings while designing the training process[3]. After some manual tweaking of the param, we ended up with a starting value of $1 * 10^{-3}$, big enough to allow the nets to explore more. During the epochs we wanted the NN to reduce the exploration and improve the stability of the parameters learned. In order to achieve this, we embedded a scheduler in the learning process; its contribution consists in changing the value of the learning rate during the various epochs. Early in the training, the learning rate is set to be large in order to reach a set of weights that are good enough. Over time, these weights are fine-tuned to reach higher accuracy by leveraging a small learning rate. In particular, we choose an exponential decay with a rate of 0.6 for every epoch.

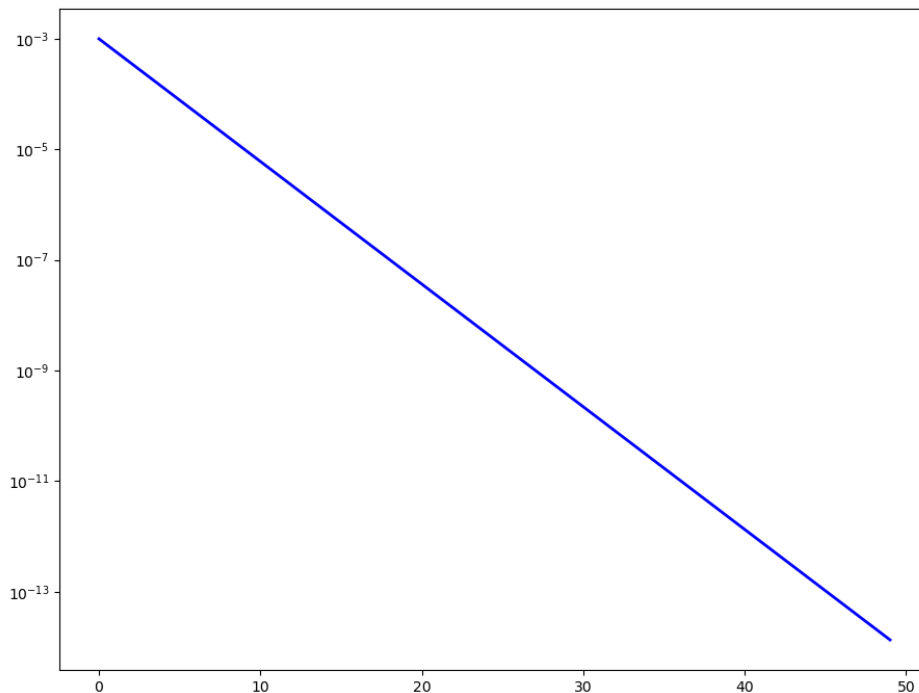


Figure 4.7: Learning rate decay during the epochs (Shown in logarithmic scale for visualization purposes).

4.5.3. Implementation details

In all the experiments we choose a batch size of 2 both for train and validation. While the use of large mini-batches increases the available computational parallelism, small-batch training has been shown to provide improved generalization performance and training stability[21]. We trained each network for 50 epochs, using a training-validation data split of 80% - 20%. We use the Adam optimizer during training and a Scaler which is used to increase the gradient scale and avoid vanishing the gradient.

We used [6] as the Unet implementation, [5] for SE-ResUnet and [4] for DeepLabV3.

The experiments were implemented in Python (version 3), using the PyTorch library to handle parallel computation over tensors. We used Matplotlib to generate the plots visible in this work and Vedo to generate the 3D results. The experiment's result summary, the configuration files, and the dataset information files were all written/generated in JSON format since its high usability and accessibility in almost every programming language. The training of the models took place mainly in the Jupyter environment, served by a machine running over a GeForce GTX 1060 with 6GB of memory.

4.6. Evaluation

In this section, we present the metrics used in the experiments to assess the prediction performances. In general, we avoid using the precision metric since the class imbalance problem makes this metric too optimistic. We start with an introduction to the Confusion Matrix, then, we present the Dice Similarity Coefficient (DSC) and the Jaccard Index, which are used in this work.

4.6.1. Confusion Matrix

If we evaluate the results of a binary segmentation process pixel by pixel, we are regressing to a binary classification process. In this setup, each pixel could be either in the right class or in the wrong one. The confusion matrix is generally useful in this context to evaluate the performances: it is a table with ground truth values as rows and predicted values as columns. These values are grouped together in order to separate the correct prediction from the wrong ones. In particular, we define

- *True Positive* (TP) the values correctly predicted as members of the first class;
- *False Positive* (FP) the values wrongly predicted as members of the first class;

- *True Negative* (TN) the values correctly predicted as members of the second class;
- *False Negative* (FN) the values wrongly predicted as members of the second class.

		Predictions	
		1	0
Ground Truth	1	TP	FN
	0	FP	TN

Figure 4.8: Confusion Matrix.

4.6.2. Dice Similarity Coefficient

The Dice Similarity Coefficient is a metric that measures the overlap of two sets of data. This index has become arguably the most broadly used tool in the validation of image segmentation algorithms created with AI, but it is a much more general concept that can be applied to sets of data for a variety of applications. It has a value range from 0 to 1 (1 being the perfect overlap), it's also the negative term in the Dice Loss presented in 4.5.1.

If we define it in terms of the classification terminology introduced before:

$$DSC = \frac{2TP}{2TP + FP + FN} \quad (4.5)$$

4.6.3. Jaccard Index

The Jaccard Index, or Jaccard Similarity Coefficient, or Tanimoto Coefficient is another metric measuring the overlap between sets of data. The Jaccard coefficient is widely used in computer science, ecology, genomics, and other sciences, where binary or binarized data are used. Recalling the classification notation it can be defined as:

$$J = \frac{TP}{TP + FP + FN} \quad (4.6)$$

Another way of seeing the Jaccard Index is the ratio between the area of intersection (between ground truth and the predicted shape) and the union of them. For this reason,

it has a behavior similar to the one of the DSC, having the total area at the denominator, it removes the dependency on the target's size.

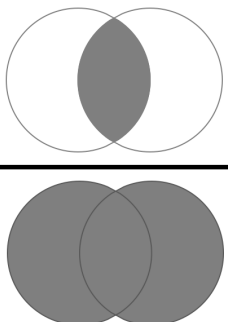
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 4.9: Jaccard Index seen as Intersection Over Union (IoU).

4.7. Summary

In this chapter we have seen the training process in detail, we have described how the data is fed to the networks, and shown the architectures used. We dived into the learning rate scheduling and technicalities regarding the actual implementation of the experiments. Finally, we presented the evaluation metrics used to assess performances.

5 | Design of Experiments

In this section, we present how the experimental works were conducted. Firstly, we analyze the binary segmentation experiments, putting the focus on the small organ process. After, we detail the Transfer Learning setup and the fine-tuning experiments, before moving on to the Multi-Organ experiments and their variations. Finally, we lay out some experiments conducted using the ensemble method Last Layer Feature Fusion.

5.1. Binary Nets

The first category of experiment design involves the binary segmentation of single OaRs and PTVs. For all these experiments we used the Unet model detailed in 4.2.1, we used different input sizes depending on the target's dimensions. We choose to segment OaR at high priority (being defined by medics) and high complexity for the manual labeling task, when the OaR occupied a tiny portion of the input size we adopted instead the Small Organ strategy described in 4.3. We also segment the different PTVs, in particular, every PTV label is a section of the total target volume, which is labeled as PTV Total.

	OaR and PTV	Center Crop Dimension
1	Oral Cavity	320
2	Ribs	512
3	Heart	512
4	Liver	512
5	Intestine	320
6	Right Lung	512
7	Left Lung	512
8	PTV Abdomen	512
9	PTV Arms	512
10	PTV Legs	512
11	PTV Head	512
12	PTV Chest	512
13	PTV Total	512
14	Rectum	320
15	Stomach	320
16	Testicles	320

Table 5.1: OARs segmented with a standard Unet

5.2. Small organ study

During the experiments' execution, we noticed that smaller organs weren't segmented with good performances. We present in the following table the organs recognized as small, together with the small window dimension used by the refined net:

	OaR	Small Window dimension
1	Esophagus	100
2	Marrow	100
3	Right Eye	100
4	Left Eye	100
5	Right Parotid	120
6	Left Parotid	120
7	Thyroid	140

Table 5.2: OARs identified as Small Organ

For these organs, we developed the specific process explained in 4.3.

In order to assess the performance gain, we conducted a study comparing different experimental setups. The study was conducted over the esophagus, being a tricky organ to segment both for neural networks and medics. The various setups are presented here in increasing performance order. The changes presented in every step are applied cumulatively in the successive ones.

5.2.1. Coarse network

Firstly, we perform a standard binary segmentation with an Unet using a center crop size of 320x320. We obtain validation scores significantly smaller concerning other bigger organs. This outcome motivates us to refine the segmentation process for these tiny OARs.

5.2.2. Refinement network

In order to improve performances, we introduced a refined net to be used in a cascade fashion. We choose to use another Unet for the refinement process; at this point the segmentation process consists of two Unet in cascade. As a Small Window Retrieval process, we used a hard-coded window position, with dimensions of 100x100.

5.2.3. Reduce data augmentation

During the previous experiments we noticed that the position of the smaller organ was not always the same and, supposing this effect was reducing the inference performances, we

decided to reduce some data augmentation effects for these smaller organs, keeping only the Elastic transformation. Also in this case we gained a bit more in validation metrics.

5.2.4. Add Context

In order to provide more information about the full-sized image in the small window, we added a context channel to the refined net's input image. The context is provided by the coarse net which is working over a 320x320 input size. The context information turned out to be useful and increased performance.

5.2.5. Use the Smart Window

Finally, we developed the Smart Window mode for the Window Retrieval Component. This enables the ability to automatically retrieve the best window position for each image assuring to place the OaR always in the center of the window.

In definitive, we see that using two Unet in a cascade fashion with reduced data augmentation, together with context information creates a high-performance segmentation process in the context of smaller organs.

5.3. Transfer Learning

The experiments carried out in the transfer learning setting are binary segmentation processes executed over the OaRs that are in common between the source and target datasets. Moreover, they needed to be executed on different networks, since we need to transfer the weights from the pretrained models; also the center crop sizing has been adapted to the ones used in the source dataset. We report here the setting used for each OaR present both in the source and target datasets:

	OaR	Model used	Center crop size
1	Left Lung	SE-ResUnet	512
2	Right Lung	DeepLabV3	512
3	Heart	DeepLabV3	320
4	Esophagus	SE-ResUnet	320
5	Marrow	SE-ResUnet	320

Table 5.3: OaRs used for the transfer learning experiments and the relative experimental setups.

For each of these OaRs, we evaluate firstly the performance of the pretrained models in the target datasets directly, without any fine-tuning of the nets, in order to ensure that the transfer learning was actually working. Then, we trained the specified network from scratch using only the target dataset, in this way we are creating a set of baseline results of the same nets (These experiments are run in the same set as the Binary net trained from scratch (5.1), the only differences being the network used and the center crop). Notice also that none of these organs has been trained using the *Small Organ* approach.

We then move on to the actual fine-tuning experiments in which we initialize the pretrained models and tune them over the target dataset. The only differences with the previous setup are the fact that the networks are initialized with the pretrained models and that the images and labels loaded from the target DS are rotated by 90 degrees in order to make them similar to the ones in the source DS.

Finally, we executed more experiments in other tweaked scenarios, as explained below.

5.3.1. Reduced number of patients

After having assessed the performance gain of the fine-tuning process, we repeated the experiments in a different scenario, the difference being the number of patients used in the training over the target DS. The motivation for this scenario is to assess the utility of the transferred knowledge in a context where data scarcity is the main problem and transfer-learning is seen as a solution. In particular, we adopted three variations:

- Only 50% of the available patients used;
- Only 30% of the available patients used;

- Only 20% of the available patients used.

We replicated the third scenario also in the baseline models trained from scratch over the 5 OaRs present in all the datasets.

5.3.2. Layer freezing

In order to evaluate the effectiveness of transfer learning in different scenarios, we also executed different experiments freezing some network layers while fine-tuning the others. In particular, we defined a *layer* as the set of operations performed in the network’s encoder between two consecutive down-sampling operations. After having initialized the networks with the pretrained states, we performed fine-tuning experiments in the following setups:

- 1 frozen layer;
- 2 frozen layers;
- 3 frozen layers.

5.3.3. Transfer Learning Summary

To summarize, we present here the set of experiments executed in the transfer learning setting for each of the five OaRs:

	General Setup	Percentage of Patients used	Number of frozen layers
1	Train from scratch	100%	0
2	Train from scratch	20%	0
3	Fine-tuning	100%	0
4	Fine-tuning	100%	1
5	Fine-tuning	100%	2
6	Fine-tuning	100%	3
7	Fine-tuning	50%	0
8	Fine-tuning	30%	0
9	Fine-tuning	20%	0

Table 5.4: Summary of the various experiments carried out in the transfer learning setting.

5.4. Multiclass

The Multiclass experiments have as an objective the ability to make inferences on multiple organs with a single end-to-end process. Since we are using some pretrained networks here too, we will run some of these experiments on the same OaRs as in section 5.3. The difference being the fact that we will segment all the five OaRs at once.

5.4.1. Single multiclass network

In this first set, we train and test the performance of a network having as output multiple channels, each of which is the inferred segmentation of a specific target. We performed different types of training from scratch in this setting, and we summarize them in the table below.

Network Used	Targets
DeepLabV3	Left Lung, Right Lung, Heart, Marrow, Esophagus
Unet	Left Lung, Right Lung, Heart, Marrow, Esophagus
DeepLabV3	Left Lung, Right Lung, Heart, Marrow, Esophagus, PTV Total
DeepLabV3	Ribs, Intestine, Left Parotid, Right Parotid, Liver, Marrow, Testicles

Table 5.5: Summary of the various experiments carried in the multiclass modality.

Apart from the experiment including the PTV Total target which is using an input size of 512x512, the other networks have been trained with an input image cropped to a size of 320x320.

5.4.2. Last Layer Feature Fusion Ensemble Method

In the Last layer feature fusion setting, we train and test a multiclass network which is composed of multiple binary pretrained networks and the feature fusion module (presented in 4.4). We performed some experiments training the fusion module over different groups of targets. In particular, we replicated some setups presented in the multiclass segmentation, and compare the results.

Center Crop	OaRs
320	Left Lung, Right Lung, Heart, Marrow, Esophagus
512	Left Lung, Right Lung, Heart, Marrow, Esophagus, PTV Total
320	Ribs, Intestine, Left Parotid, Right Parotid, Liver, Marrow, Testicles

Table 5.6: Summary of the various experiments carried using the Last Layer Feature Fusion ensemble method.

5.5. Summary

In this section we went through the design of the various experiments in detail, we lined out firstly the base case of the binary segmentation networks, and we analyzed how the Small Organ segmentation process was developed. After, we move on to the transfer learning scenarios and their variants; finally, we presented the Multiclass settings involving the segmentation of multiple OaRs at the same time.

6 | Results

In this chapter, we present the result of our experiments. We start from the models trained from scratch, presenting the quantitative results, including some graphs and visualization over the data, after, we present some qualitative results showing predictions over test input slices. Then, we proceed presenting the result obtained in the various transfer-learning setting where each case is presented starting from a quantitative perspective followed by a qualitative one. The same structure is then repeated for the experiments executed with the Multiclass networks and the Last Layer Feature Fusion ensemble models. For each category, we also provide some discussion over the results achieved.

The values presented in the quantitative results are the Dice Score Coefficient and Jaccard Index computed over the window of 10 last training epochs. In particular, for every epoch we compute DSC and Jaccard Index over the validation patients, then, we group the results of the last 10 epochs. The value shown here are the average, maximum and minimum of the scores computed over the epochs window. For both DSC and Jaccard index, we show values in percentage, where 100% represent a value of 1, being the maximum value achievable in both cases.

6.1. Training from scratch results

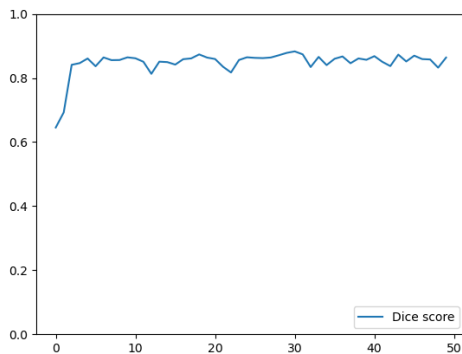
We present here the results of binary nets trained from scratch over the target dataset, as lined out in 5.1. These networks are standard Unet feed with the CT Scan slice and the binary mask of the single target in consideration. For both the DSC and the Jaccard Index, we present the average value, the maximum and the minimum values.

For every target, we also show an example of prediction over a meaningful slice, together with the respective mask annotated by specialists. We start showing some ordinary targets, segmented using a single Unet, then we move on presenting some targets segmented in the Small organ setting.

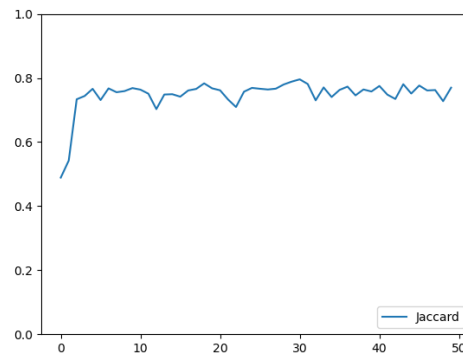
Quantitative Results

	DSC	Jaccard Index
Oral Cavity	85.62 (83.22 - 87.29)	75.86 (72.76 - 78.05)
Heart	91.83 (91.20 - 92.60)	85.78 (84.84 - 86.96)
Left Lung	96.39 (96.12 - 96.88)	93.53 (93.15 - 94.23)
Right Lung	96.21 (95.60 - 96.83)	93.39 (92.43 - 94.23)
PTV Total	79.11 (74.17 - 80.70)	67.99 (63.39 - 69.82)

Table 6.1: Result of Binary segmentation from scratch over some targets.

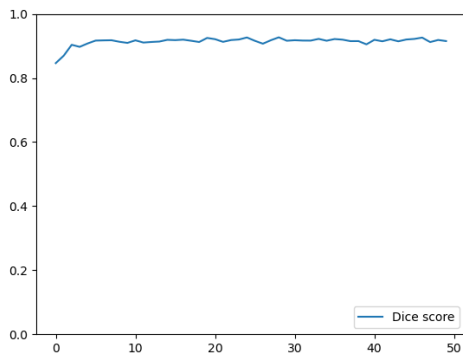


(a) DSC values' evolution.

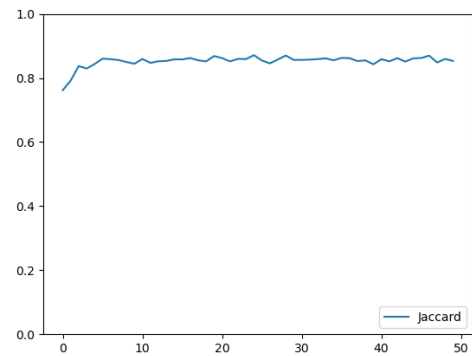


(b) Jaccard Index values' evolution.

Figure 6.1: Validation indexes evolution during training over the target: Oral Cavity.

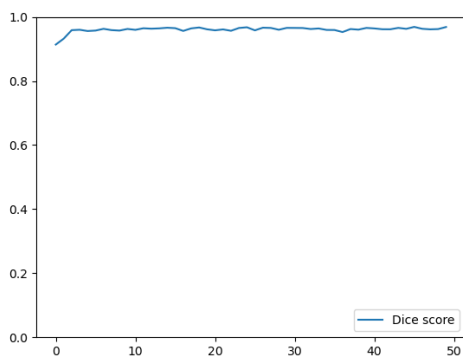


(a) DSC values' evolution.

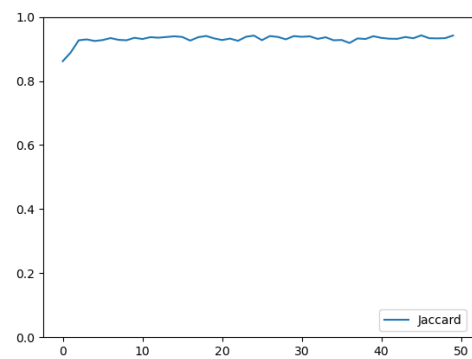


(b) Jaccard Index values' evolution.

Figure 6.2: Validation indexes evolution during training over the target: Heart.

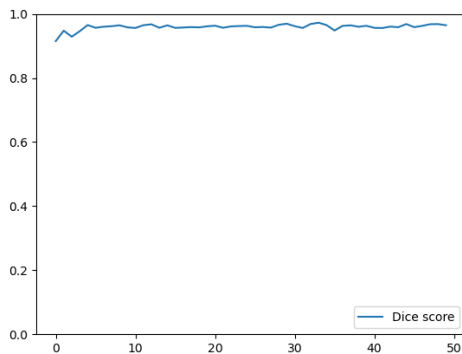


(a) DSC values' evolution.

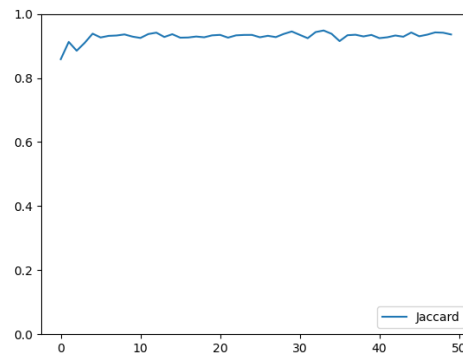


(b) Jaccard Index values' evolution.

Figure 6.3: Validation indexes evolution during training over the target: Left Lung.

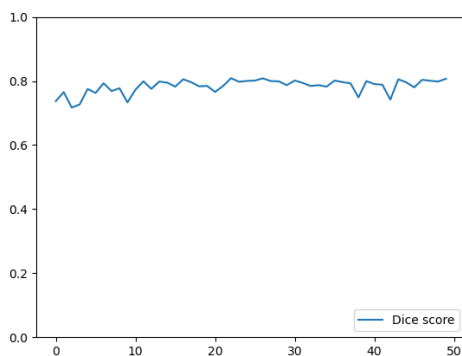


(a) DSC values' evolution.

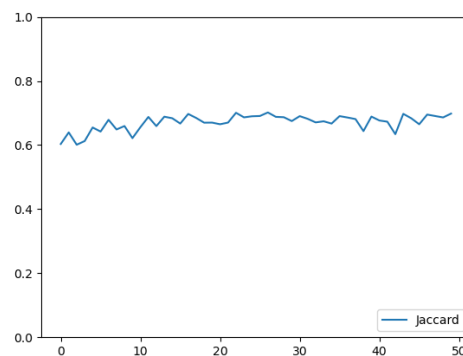


(b) Jaccard Index values' evolution.

Figure 6.4: Validation indexes evolution during training over the target: Right Lung.



(a) DSC values' evolution.



(b) Jaccard Index values' evolution.

Figure 6.5: Validation indexes evolution during training over the target: PTV Total.

Qualitative Results

Notice that each medical image is flipped horizontally, this is happening because they are read from the specialists as though facing the patient. In the following images, the left lung will therefore appear at the right and *viceversa*.

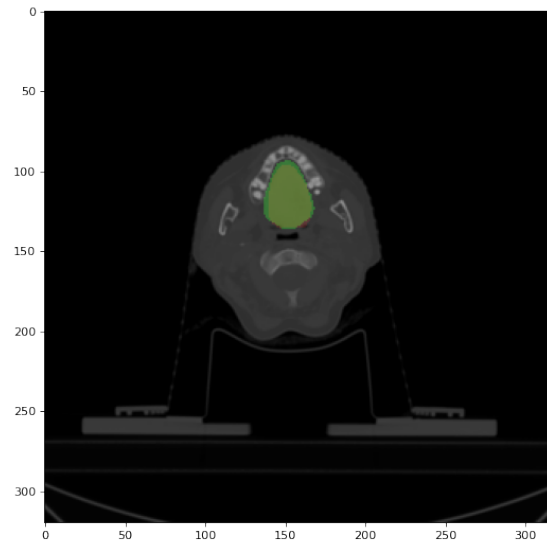


Figure 6.6: The prediction (in green) and the ground truth (in red) of the target: Oral Cavity. Notice that the overlap between prediction and ground truth is shown in yellow.

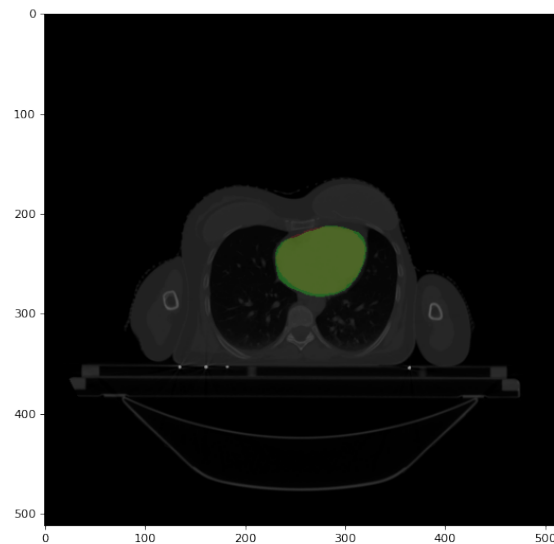


Figure 6.7: The prediction (in green) and the ground truth (in red) of the target: Heart.

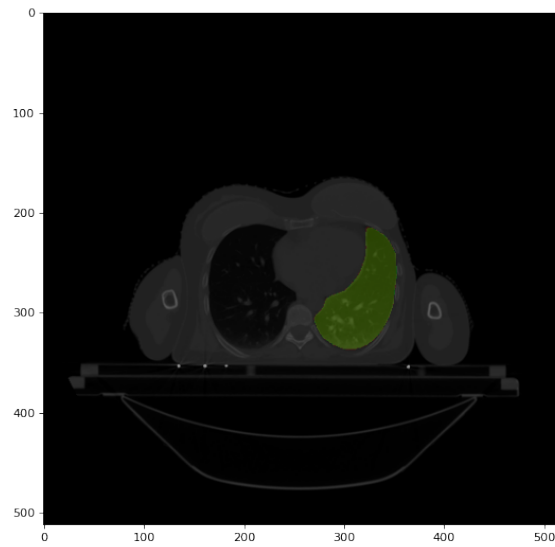


Figure 6.8: The prediction (in green) and the ground truth (in red) of the target: Left Lung.

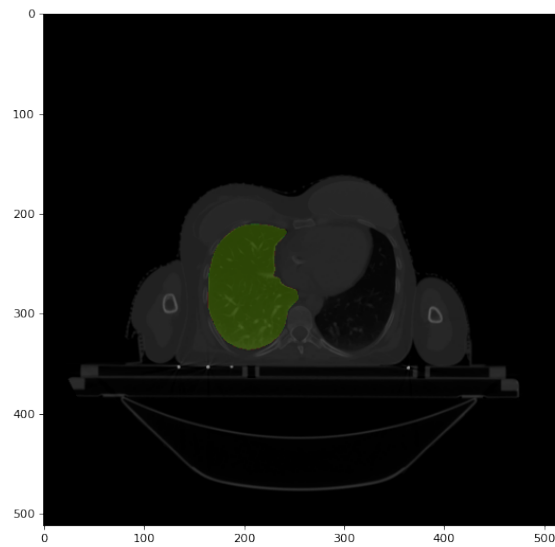


Figure 6.9: The prediction (in green) and the ground truth (in red) of the target: Right Lung.

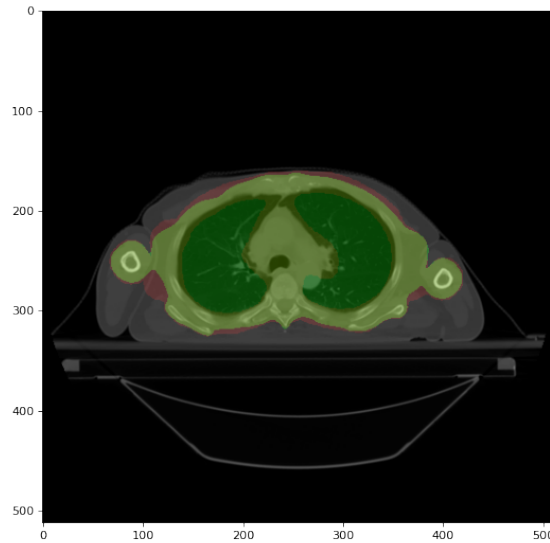


Figure 6.10: The prediction (in green) and the ground truth (in red) of the target: PTV Total.

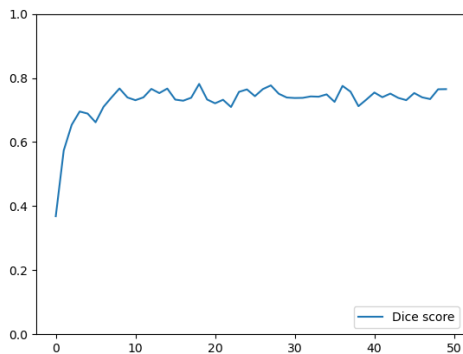
6.1.1. Small Organs

Quantitative Results

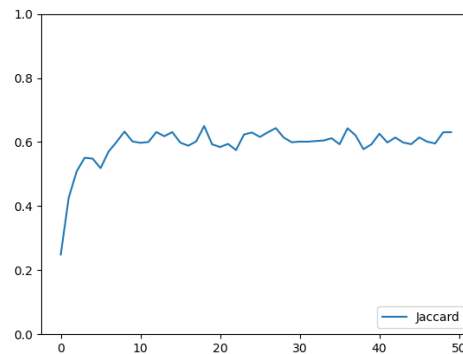
We present here some DSC and Jaccard index values showing the segmentation performance of the Unet over the targets Esophagus and Marrow, notice that we used a central crop of 320x320.

	DSC	Jaccard Index
Esophagus	74.71 (73.07 - 76.51)	61.02 (59.33 - 63.07)
Marrow	82.01 (74.72 - 85.71)	71.06 (65.03 - 75.60)

Table 6.2: Result of Binary segmentation from scratch over the targets: Esophagus, Marrow.

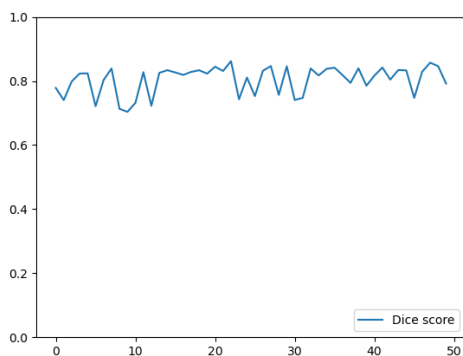


(a) DSC values' evolution.

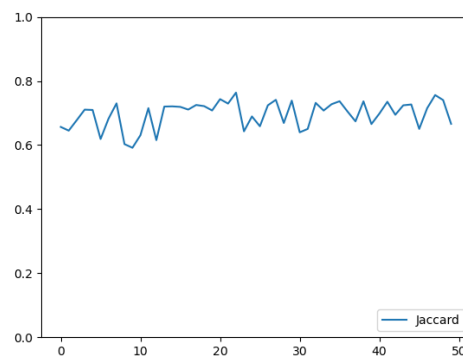


(b) Jaccard Index values' evolution.

Figure 6.11: Validation indexes evolution during training over the target Esophagus.



(a) DSC values' evolution.



(b) Jaccard Index values' evolution.

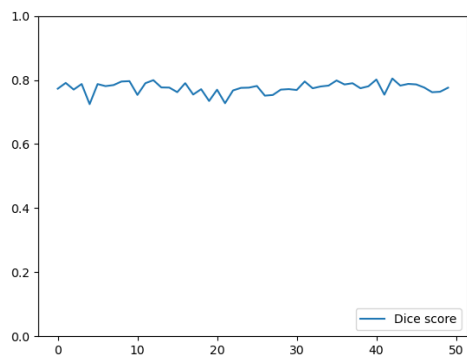
Figure 6.12: Validation indexes evolution during training over the target Marrow.

The esophagus and marrow targets, are classified as *Small Organ*, therefore, we segmented them using the refinement net, and we present in the table 6.3 the results achieved. Notice that the small window size is 100x100 for the refinement net in both the targets. For better comparison, we present the values in both the small window retrieval mode: Hard-coded and Smart.

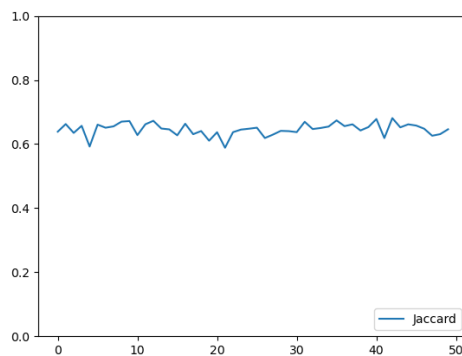
	DSC	Jaccard Index
Hard-coded Window		
Esophagus	77.06 (74.88 - 78.90)	63.98 (61.61 - 66.17)
Marrow	82.05 (73.03 - 87.41)	72.36 (63.36 - 78.97)
Smart Window		
Esophagus	77.94 (75.41 - 80.46)	64.99 (61.86 - 68.09)
Marrow	83.54 (75.75 - 85.80)	73.16 (66.65 - 75.81)

Table 6.3: Result of Binary segmentation from scratch over the targets: Esophagus and Marrow.

To avoid redundancy, in the following images, we present the evolution of the validation scores only in the smart window modality.



(a) DSC values' evolution.



(b) Jaccard Index values' evolution.

Figure 6.13: Validation indexes evolution during training of the target Esophagus.

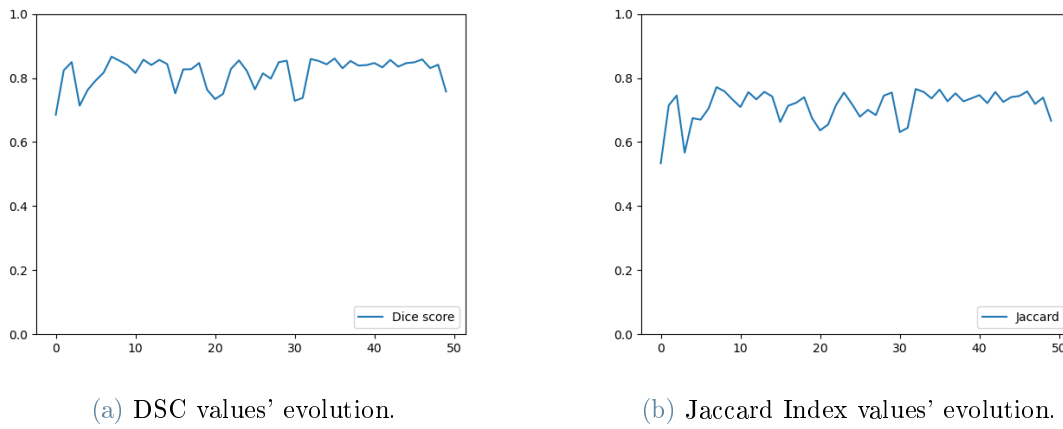


Figure 6.14: Validation indexes evolution during training of the target Marrow.

Qualitative Results

We present the segmentation output together with the manually annotated mask of a meaningful slice segmented using the small organ approach in smart window mode.

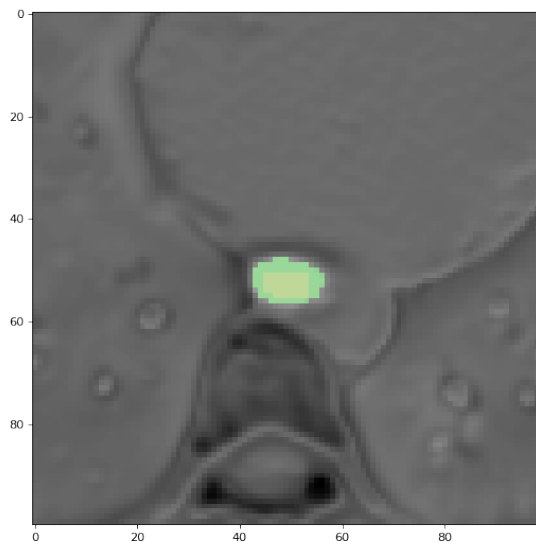


Figure 6.15: The prediction (in green) and the ground truth (in red) of the target: Esophagus. Notice that the overlap between prediction and ground truth is shown in yellow.

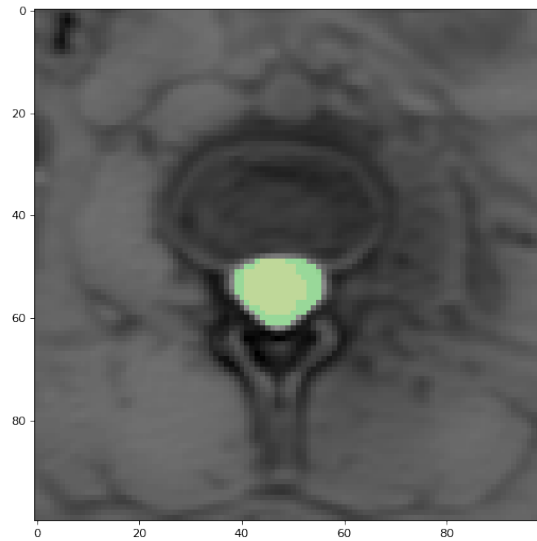


Figure 6.16: The prediction (in green), the ground truth (in red) of the target: Marrow. Notice that the overlap between prediction and ground truth is shown in yellow.

6.1.2. Esophagus study

We present here the different validation performances for each type of experiment conducted over the target esophagus (detailed in 5.2) in order to design the small organ segmentation process.

	Segmentation Process	Dice Score Coefficient	Jaccard Index
1	Coarse network	74.71	61.02
2	Refinement net	73.79	60.58
4	Reduce data augmentation	77.99	65.44
5	Add Context	78.26	65.66
6	Use the Smart Window	77.94	64.99

Table 6.4: Average Validation Performance of different experimental setups in the small organ context for the target: Esophagus

6.1.3. Discussion

We have noticed the different results that the same CNN can achieve over different organs. We can therefore use these result as indicators of the difficulty in segmenting the different OaRs. The most complex-to-segment OaRs corresponds to the ones annotated as difficult by the medics, this mean that those organs are in fact harder to recognize starting from the input slice. In particular, our refinement process developed to tackle the smaller organ shows improvements over the base results as we saw in 6.1.1. Interestingly, we can notice that the performance of the segmentation has not relevant changes between the *Hard-coded window* mode and the *Smart Window* one. In fact, the only difference between those approaches is the position of the OaR and, in general, the features learned from a CNN are position invariant. CNN’s filter slides from left-to-right and top-to-bottom across an input, and will activate when it comes across a particular edge-like region, corner, or color blob. Therefore, CNNs can be seen as *not caring* exactly where an activation fires, simply that it does fire, and, in this way, we naturally handle translation inside a CNN.

6.2. Fine-tuning results

In this section, we present the results of the experiments conducted fine-tuning the pre-trained networks, as lined out in 5.3. For each target in analysis, we show a quantitative comparison of the results from the different settings, then we share some data visualization of the experiment results. We include also the quantitative result obtained training the networks from scratch, as baseline results obtained with the different architecture used to match the pretrained one.

In this section, for the DSC and Jaccard index, we show average, minimum and maximum values; all in the same table cell for visualization purposes.

6.2.1. Left Lung

Quantitative Results

We recall that the left lung segmentation is done with a 512x512 input size and a SE-ResUnet architecture to match the pretrained model setup.

General Setup	Patients' percentage	FL	DSC	Jaccard Index
Train from scratch	100%	0	95.81 (95.11 - 96.38)	92.69 (91.74 - 93.40)
Train from scratch	20%	0	95.66 (95.16 - 96.07)	92.54 (91.66 - 95.16)
Fine-tuning	100%	0	96.03 (95.18 - 96.74)	93.09 (91.82 - 94.04)
Fine-tuning	100%	1	95.83 (95.20 - 96.27)	92.73 (91.84 - 93.44)
Fine-tuning	100%	2	95.81 (95.38 - 96.31)	92.79 (92.42 - 93.39)
Fine-tuning	100%	3	95.24 (94.15 - 96.12)	91.91 (90.57 - 92.98)
Fine-tuning	50%	0	95.78 (94.90 - 96.68)	92.72 (91.78 - 93.83)
Fine-tuning	30%	0	95.75 (95.00 - 96.39)	92.59 (91.72 - 93.55)
Fine-tuning	20%	0	95.77 (94.93 - 96.67)	92.81 (91.61 - 93.90)

Table 6.5: DSC and Jaccard Index values of the different single class transfer learning settings. FL: frozen layers

Qualitative results

The prediction show is obtained using the model trained over the complete fine-tuning setting: 100% of available patients and 0 frozen layers.

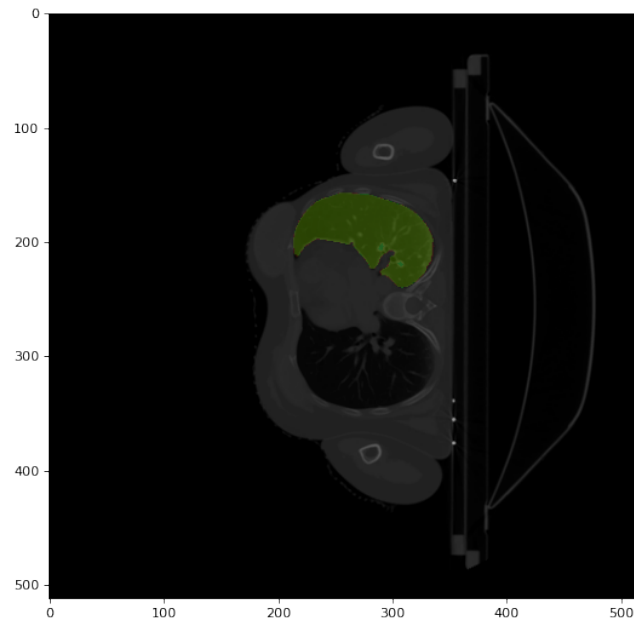


Figure 6.17: The prediction (in green) and the ground truth (in red) of the target: Left Lung. Notice that the overlap between prediction and ground truth is shown in yellow.

6.2.2. Right Lung

Quantitative Results

We recall that the right lung segmentation is done with a 512x512 input size and a DeepLabV3 architecture to match the pretrained model setup.

General Setup	Patients' percentage	FL	DSC	Jaccard Index
Train from scratch	100%	0	94.17 (93.45 - 95.01)	90.06 (89.24 - 91.18)
Train from scratch	20%	0	93.08 (92.78 - 93.44)	88.34 (87.88 - 88.94)
Fine-tuning	100%	0	95.03 (94.39 - 95.88)	91.45 (90.73 - 92.57)
Fine-tuning	100%	1	95.05 (94.59 - 95.78)	91.44 (90.71 - 92.29)
Fine-tuning	100%	2	94.99 (93.93 - 95.86)	91.35 (89.87 - 92.50)
Fine-tuning	100%	3	94.97 (94.12 - 95.58)	91.29 (90.38 - 91.96)
Fine-tuning	50%	0	94.77 (93.97 - 95.60)	91.10 (90.03 - 92.15)
Fine-tuning	30%	0	94.90 (94.24 - 95.47)	91.31 (90.44 - 92.01)
Fine-tuning	20%	0	94.30 (93.42 - 95.33)	90.52 (89.65 - 91.79)

Table 6.6: DSC and Jaccard Index values of the different single class transfer learning settings. FL: frozen layers

Qualitative results

The prediction show is obtained using the model trained over the complete fine-tuning setting: 100% of available patients and 0 frozen layers.

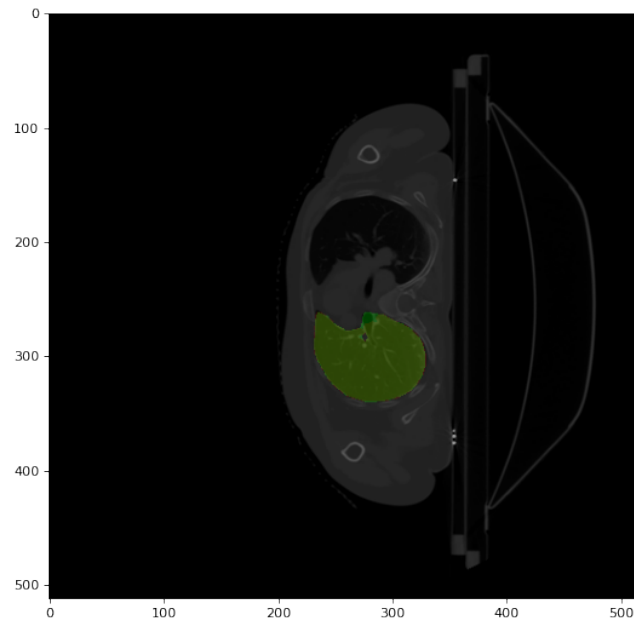


Figure 6.18: The prediction (in green) and the ground truth (in red) of the target: Right Lung. Notice that the overlap between prediction and ground truth is shown in yellow.

6.2.3. Heart

Quantitative Results

We recall that the heart segmentation is done with a 320x320 input size and a DeepLabV3 architecture to match the pretrained model setup.

General Setup	Patients' percentage	FL	DSC	Jaccard Index
Train from scratch	100%	0	87.91 (86.34 - 89.48)	80.47 (78.68 - 82.42)
Train from scratch	20%	0	84.95 (83.44 - 86.25)	76.74 (75.00 - 78.16)
Fine-tuning	100%	0	90.07 (88.59 - 91.44)	83.46 (81.53 - 85.23)
Fine-tuning	100%	1	89.82 (88.83 - 91.03)	83.13 (82.00 - 84.69)
Fine-tuning	100%	2	89.63 (88.39 - 90.61)	82.82 (81.11 - 84.36)
Fine-tuning	100%	3	88.73 (87.16 - 89.64)	81.70 (79.66 - 83.06)
Fine-tuning	50%	0	88.55 (86.84 - 90.03)	81.47 (79.73 - 83.08)
Fine-tuning	30%	0	87.54 (86.09 - 88.91)	80.32 (78.79 - 82.22)
Fine-tuning	20%	0	86.13 (84.74 - 87.21)	78.58 (76.64 - 80.04)

Table 6.7: DSC and Jaccard Index values of the different single class transfer learning settings. FL: frozen layers

Qualitative results

The prediction show is obtained using the model trained over the complete fine-tuning setting: 100% of available patients and 0 frozen layers.

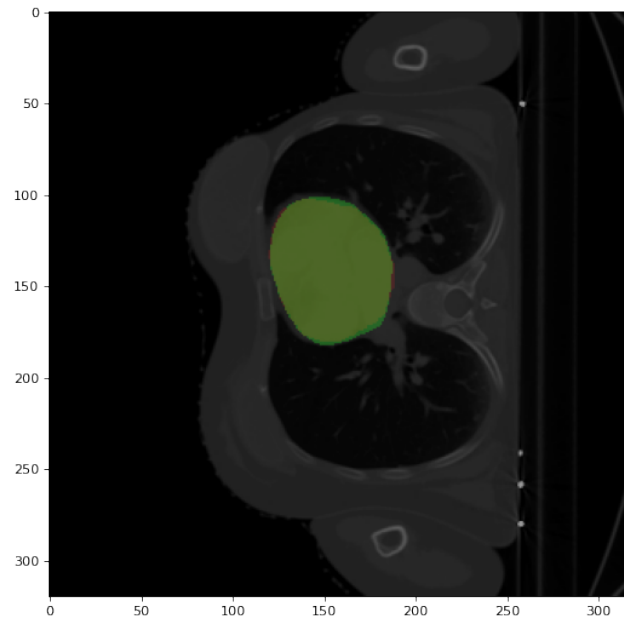


Figure 6.19: The prediction (in green) and the ground truth (in red) of the target: Heart. Notice that the overlap between prediction and ground truth is shown in yellow.

6.2.4. Marrow

Quantitative Results

We recall that the Marrow segmentation is done with a 320x320 input size and a SE-ResUnet architecture to match the pretrained model setup.

General Setup	Patients' percentage	FL	DSC	Jaccard Index
Train from scratch	100%	0	81.79 (76.25 - 84.72)	70.58 (63.43 - 74.67)
Train from scratch	20%	0	00.67 (00.31 - 01.36)	00.02 (00.01 - 00.02)
Fine-tuning	100%	0	80.26 (70.97 - 85.28)	68.89 (60.51 - 75.10)
Fine-tuning	100%	1	79.71 (69.07 - 84.20)	68.26 (57.60 - 73.57)
Fine-tuning	100%	2	79.97 (69.80 - 84.75)	68.86 (58.96 - 74.24)
Fine-tuning	100%	3	76.76 (69.39 - 80.20)	64.61 (59.03 - 68.43)
Fine-tuning	50%	0	78.64 (70.73 - 83.28)	67.54 (60.25 - 72.18)
Fine-tuning	30%	0	80.89 (73.61 - 83.95)	69.56 (63.38 - 73.10)
Fine-tuning	20%	0	78.64 (70.73 - 83.28)	67.54 (60.25 - 72.18)

Table 6.8: DSC and Jaccard Index values of the different single class transfer learning settings. FL: frozen layers

Qualitative results

The prediction show is obtained using the model trained over the complete fine-tuning setting: 100% of available patients and 0 frozen layers.

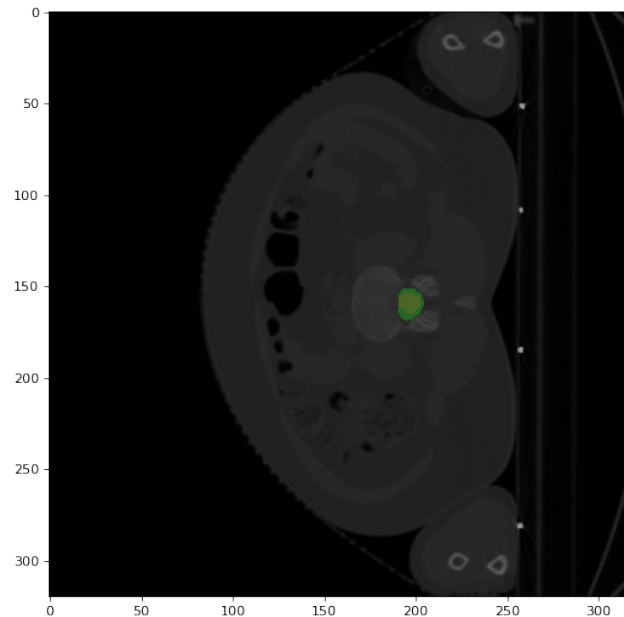


Figure 6.20: The prediction (in green) and the ground truth (in red) of the target: Marrow. Notice that the overlap between prediction and ground truth is shown in yellow.

6.2.5. Discussion

We have seen during our experiments in the transfer learning settings that the rotation of the input image was necessary to be able to have some meaningful results. In general, unless your training data includes targets that are rotated across the full 360-degree spectrum, your CNN is not truly rotation invariant [1].

In the transfer learning scenarios, we have confirmed the utility of the transferred knowledge and the networks' initialization with a pretrained state; in fact, the fine-tuned nets showed better performances over the pretrained ones. The experiments carried out with some frozen layer are not showing a unique tendency; we can see that, in general, the greater the number of frozen layers, the greater the performance loss will be. This effect, though, is not always present, and, following what we said in 2.1.5, we can suppose that the feature *co-adaptation* is responsible for this effect. Regarding the experiments carried out over different subsections of the target dataset, we can see a general trend in which the fewer the available patients, the poorer the performance will be. However, this effect has different quantitative outcomes based on the OaR analyzed: we have seen a critical drop in performance in "harder" OaRs like the Marrow and a smaller effect in "easier"

OaR like the Lungs. To explain this effect, we can suppose that some patients are more important than others based on how they were positioned during the scan or based on the intensity values of their CT Scans.

6.3. Multi Class results

In this section we show the results of the experiments performed in the multi-class setting as described in 5.4.1. In the following section we present the results obtained from the different targets considered. As before, we present the DSC and Jaccard Index values showing average, minimum and maximum in the same table cell.

6.3.1. Multiclass segmentation - 5 targets

We present here the result of a multiclass segmentation performed from scratch using the DeepLabV3 network over 5 targets: Left Lung, Right Lung, Heart, Marrow, Esophagus. Every input slice has been cropped to reach the dimension of 320x320.

	OaR	DSC	Jaccard Index
1	Left Lung	95.45 (94.39 - 96.63)	92.29 (90.66 - 93.74)
2	Right Lung	95.46 (93.48 - 97.16)	92.11 (89.51 - 94.61)
3	Heart	87.76 (85.43 - 90.34)	79.77 (76.21 - 83.20)
4	Marrow	83.06 (78.50 - 88.76)	72.02 (65.48 - 80.53)
5	Esophagus	63.62 (57.95 - 67.21)	48.74 (44.34 - 52.20)

Table 6.9: DSC and Jaccard Index values of the different targets segmented from scratch using the DeepLabV3 model in the multiclass setup.

6.3.2. Multiclass segmentation - 6 targets

We performed a multiclass segmentation experiment using 6 targets, including a target volume representing the total areas of the body targeted by the planned radiotherapy treatment. We trained from scratch the DeepLabV3 model over the 512x512 input slice.

	OaR	DSC	Jaccard Index
1	Left Lung	92.09 (90.51 - 93.51)	86.66 (84.59 - 88.52)
2	Right Lung	91.77 (88.49 - 93.50)	86.29 (82.13 - 88.35)
3	Heart	84.80 (81.93 - 88.96)	76.25 (73.26 - 81.29)
4	Marrow	81.03 (78.97 - 82.68)	68.90 (65.90 - 71.43)
5	Esophagus	61.72 (54.76 - 68.47)	47.17 (40.88 - 53.66)
6	PTV Total	80.21 (74.51 - 84.59)	68.23 (61.33 - 73.80)

Table 6.10: DSC and Jaccard Index values of the different targets segmented from scratch using the DeepLabV3 model in the multiclass setup.

6.3.3. Multiclass segmentation - 8 targets

We present here another multiclass segmentation setting, we used the DeepLabV3 model trained over 8 targets: Ribs, Intestine, Left Parotid, Right Parotid, Liver, Heart, Marrow, Testicles. We used 320x320 as central crop dimension.

	OaR	DSC	Jaccard Index
1	Ribs	74.53 (66.65 - 77.32)	60.09 (51.88 - 63.34)
2	Intestine	81.78 (78.39 - 84.17)	70.81 (66.93 - 73.49)
3	Right Parotid	66.32 (60.99 - 70.19)	51.97 (47.72 - 55.65)
4	Left Parotid	66.57 (59.37 - 72.77)	51.49 (65.90 - 58.58)
5	Liver	84.32 (81.69 - 87.57)	74.35 (72.01 - 78.56)
6	Heart	86.30 (82.08 - 89.88)	77.93 (72.42 - 82.24)
7	Marrow	82.39 (79.43 - 85.22)	71.21 (68.07 - 74.69)
8	Testicles	75.18 (72.19 - 78.87)	62.23 (57.44 - 66.84)

Table 6.11: DSC and Jaccard Index values of the different targets segmented from scratch using the DeepLabV3 model in the multiclass setup.

6.3.4. Discussion

In the multiclass scenarios, we see that the performances obtained by the networks are a bit worse than the ones achieved by the binary networks trained from scratch, the main motivation for this is the increase in complexity of the segmentation task over the constant amount of training data. Here we can see that while increasing the number of targets to be segmented, we lose in segmentation performances. In order to reach better results we experimented also with the ensemble method in the multiclass segmentation, the results are presented in the next section.

6.4. Last Layer Feature Fusion results

In this section, we present results of the experiments performed using the ensemble method Last Layer Feature fusion, as detailed in 5.4.2. As before, the results are presented in average, maximum, minimum DSC and Jaccard index computed over the last 10 training epoch.

6.4.1. Last Layer Feature Fusion - 5 targets

	OaR	DSC	Jaccard Index
1	Right Lung	95.30 (92.99 - 97.52)	92.29 (89.71 - 95.34)
2	Left Lung	96.92 (96.31 - 97.73)	94.50 (93.25 - 95.75)
3	Heart	88.26 (84.35 - 90.87)	80.93 (77.02 - 84.44)
4	Esophagus	71.56 (62.83 - 76.74)	57.84 (49.97 - 63.16)
5	Marrow	79.15 (56.78 - 90.23)	69.71 (49.93 - 82.87)

Table 6.12: DSC and Jaccard Index values of the different targets segmented using the Last Layer Feature Fusion ensemble method.

6.4.2. Last Layer Feature Fusion - 6 targets

	OaR	DSC	Jaccard Index
1	Right Lung	96.03 (93.22 - 97.94)	93.43 (89.58 - 96.00)
2	Left Lung	96.92 (95.83 - 97.79)	94.33 (92.43 - 95.75)
3	Heart	88.29 (83.19 - 91.60)	81.19 (74.13 - 85.46)
4	Esophagus	71.94 (68.80 - 76.93)	58.00 (54.34 - 63.46)
5	Marrow	84.05 (80.41 - 87.34)	73.62 (68.14 - 78.31)
6	PTV Total	85.26 (79.98 - 88.38)	75.37 (68.41 - 79.67)

Table 6.13: DSC and Jaccard Index values of the different targets segmented using the Last Layer Feature Fusion ensemble method.

6.4.3. Last Layer Feature Fusion - 8 targets

	OaR	DSC	Jaccard Index
1	Ribs	78.37 (73.53 - 82.55)	65.61 (60.58 - 70.54)
2	Intestine	85.34 (83.66 - 87.81)	76.02 (74.71 - 78.77)
3	Right Parotid	00.23 (00.17 - 00.31)	00.00 (00.00 - 00.00)
4	Left Parotid	00.27 (00.19 - 00.36)	00.00 (00.00 - 00.00)
5	Liver	91.53 (88.74 - 93.49)	85.07 (81.84 - 87.87)
6	Heart	87.99 (84.38 - 90.28)	80.38 (76.44 - 83.23)
7	Marrow	80.53 (79.08 - 83.57)	68.69 (66.50 - 73.24)
8	Testicles	00.22 (00.11 - 00.87)	00.00 (00.00 - 00.00)

Table 6.14: DSC and Jaccard Index values of the different targets segmented using the Last Layer Feature Fusion ensemble method.

6.4.4. Discussion

The ensemble methods are, in general, able to obtain higher performances with respect to the multiclass settings. The reason for this is the use of a pretrained binary net for each target to be segmented. An exception to this trend is the experiment with 8 targets where the smaller ones are showing poor performances after the training. The reason for this is the high unbalance between the targets' sizes, both in terms of area in the single slice and the number of slices where the target is present. In this unbalanced setting, the training will focus on the bigger targets that account for a higher portion of the loss function. We remind also that the training here involves only the fusion layer, while the binary networks are frozen.

6.5. Summary

We've seen, in this chapter, the result of the experiments carried out; we've presented four main setups:

- Binary nets;
- Transfer Learning;
- Multiclass;
- Last Layer Feature Fusion.

For each category, we've shown the quantitative results, some examples of inference over real data, and a discussion giving insights over the data.

7 | Conclusions and future works

In the previous chapter, we have shown the quantitative and qualitative results of the work carried over the target dataset. In particular, we have trained various networks to segment different targets with high performance, using the Unet model we have created a baseline of results showing how the label's complexity can influence the segmentation result, moreover, we noticed that the targets considered more complex by the specialists are the ones that are segmented with poorer results. We have also developed a specific pipeline to deal with smaller organs that are known in the literature for being hard to segment. The architecture composed of two segmentation networks, achieve better results with respect to the baseline, but it has some minor drawback: higher model complexity and longer training time. The results of the different transfer learning settings confirm the hypothesis that the usage of network pretrained over publicly available dataset is beneficial while working on a similar private dataset. The fine-tuned nets provide, in general, higher scores with respect to the baseline nets trained from scratch. The variations of these experiments show that the performances decrease in a way proportional to the number of frozen layers; this effect is even stronger when considering more complex targets (i.e. esophagus).

In the multiclass setting, we have firstly generated a set of baseline results, which achieve a set of per-target performances that are in general lower with respect to the ones obtained by the binary networks. We took into account the ensemble methods to see if this approach could improve the segmentation results, by using a binary net for each target and then merging the results. As a general trend, the ensemble method considered, the last layer feature fusion, showed a boost in segmentation performance over the baseline multiclass scenarios for some targets. The ensemble approach has proved to reach segmentation accuracy as high as the one obtained by the binary nets also in the multiclass scenario for bigger targets, while the smaller ones are purged during training. This is due to the unbalanced class used, while optimizing, the fusion module will give priority to the bigger targets that are responsible for bigger differences in the loss values. The drawback of this setting is also the need for binary networks trained over the single target, on the contrary, the inference time, with enough parallel computation available, remains similar to the one

needed by a single binary network allowing fast multi-label segmentation.

To generalize, the results obtained are proof of the ability of the CNNs in the task of organ segmentation from CT Scan images and the efficiency of these automated methods over the handmade segmentation done by specialists.

In this chapter we conclude by presenting some possible problems of our experimental environment and, finally, we propose some future evolution of this work.

7.1. Open Problems

Over this section, we present some possible problems not tackled by this work.

- **Model used:** in our work we mainly focused on Fully Convolutional Neural Networks (FCN); for our work on the binary networks we used a simple Unet in order to have a comparison of results from different OaRs, other networks and architectures could be used in order to achieve better segmentation results. For example, we could think about some different approaches like Generative Adversarial Networks (GANs).
- **Dataset size:** the Dataset used, AUTOMI, is composed of 100 patients, but not all of them present all the manually annotated labels. In some cases, the number of patients available is less than half with respect to the total (see Appendix A). We suppose that using a dataset with more annotation available could lead to better performances.

7.2. Future Development

In this section, we discuss what we think are possible future research directions.

- **Training hyperparameters:** in this work, we did not tune extensively hyperparameters, we used values often present in the literature, we believe that an extensive search for the best hyperparameters could lead to an increase in performance.
- **Integration into CAD system:** possible future development of this work could consist in the integration of the automatic segmentation networks in a clinical environment, where the specialists did not need to spend time manually labeling each organ, but they would only need to supervise and check the results of the automatic processes, providing corrections if needed.
- **Confidence Estimation:** linked to the previous point, a possible evolution of this work could be the development of an estimation metric for each prediction. It

consists of a value computed during the segmentation process which will represent how much the network is confident with the result provided. With this setup, it will be easier for specialists to detect problems and, on the other side, avoid wasting too much time on robust results. The system could, therefore, highlight difficult situations and uncertainty in the results.

Bibliography

- [1] Are cnns invariant to translation, rotation, and scaling? <https://pyimagesearch.com/2021/05/14/are-cnns-invariant-to-translation-rotation-and-scaling/>.
- [2] Computed tomography ct. <https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct>.
- [3] Learning rate schedules and adaptive learning rate methods for deep learning. <https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1>.
- [4] Deeplabv3 implementation. <https://pypi.org/project/segmentation-models-pytorch/>.
- [5] Se-resnet implementation. <https://github.com/zjuybh/StructSeg2019>.
- [6] Unet implementation. <https://github.com/milesial/Pytorch-UNet>.
- [7] Z. Cao, B. Yu, B. Lei, H. Ying, X. Zhang, D. Z. Chen, and J. Wu. Cascaded se-resnet for segmentation of thoracic organs at risk. *Neurocomputing*, 453:357–368, 2021. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2020.08.086>. URL <https://www.sciencedirect.com/science/article/pii/S092523122100093X>.
- [8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation, 2017. URL <https://arxiv.org/abs/1706.05587>.
- [9] P. Domingos. *The Master Algorithm*. 2015.
- [10] X. Dong, Y. Lei, T. Wang, and M. Thomas. Automatic multiorgan segmentation in thorax ct images using u-net-gan. 2019.
- [11] R.-R. Galea, L. Diosan, A. Andreica, L. Popa, S. Manole, and Z. Bálint. Region-of-interest-based cardiac image segmentation with deep learning. *applied sciences*, 2021.

- [12] J. Gao, Q. Jiang, B. Zhou, and D. Chen. Convolutional neural networks for computer-aided detection or diagnosis in medical image analysis: An overview. 2019.
- [13] Y. Gao, R. Huang, Y. Yang, J. Zhang, K. Shao, C. Tao, Y. Chen, D. N. Metaxas, H. Li, and M. Chen. Focusnetv2: Imbalanced large and small organ segmentation with adversarial shape constraint for head and neck ct images. *Elsevier*, 2021.
- [14] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. 1949.
- [15] Y. Hu¹, X. Liu, X. Wen, C. Niu, and Y. Xia. Brain tumor segmentation on multi-modal mr imaging using multi-level upsampling in decoder. 2019.
- [16] B. Ibragimov and L. Xing. Segmentation of organs-at-risks in head and neck ct images using convolutional neural networks. *Med Phys*, 2017.
- [17] K. Kamnitsas, E. Ferrante, S. Parisot, C. Ledig¹, A. Nori, A. Criminisi, D. Rueckert¹, and B. Glocker. Deepmedic for brain tumor segmentation. 2017.
- [18] B. Lee, N. Yamanakkanavar, and J. Y. Choi. Automatic segmentation of brain mri using a novel patch-wise u-net deep architecture. *PLOS ONE*, 15(8):1–20, 08 2020. doi: 10.1371/journal.pone.0236493. URL <https://doi.org/10.1371/journal.pone.0236493>.
- [19] P. M., H. D., A. C., and N. A. 3d convolutional neural networks for tumor segmentation using long-range 2d contex. *Computerized Medical Imaging and Graphics*, 2019.
- [20] J. Ma, J. Chen, M. Ng, R. Huang, Y. Li, C. Li, X. Yang, and A. L. Martel. Loss odyssey in medical image segmentation. 2021.
- [21] D. Masters and C. Lusch. Revisiting small batch training for deep neural networks. 2018.
- [22] J. McCarthy. What is artificial intelligence? 2007.
- [23] T. B. Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, 2015.
- [24] S. J. Pan and Q. Yang. A survey on transfer learning. 2010.
- [25] P. Roncaglioni. Ensemble methods for multi-organ semantic segmentation. 2021. URL <http://hdl.handle.net/10589/183382>.
- [26] H. R. Roth, L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, and E. Turk-

- bey. A new 2.5d representation for lymph node detection using random sets of deep convolutional neural network observations. *PubMed Central*, 2014.
- [27] F. Sciacca. Organs at risk. *Radiopaedia.org*, 2022. URL <https://doi.org/10.53347/rID-80650>.
- [28] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. 2017.
- [29] J. Y. C. Wong, A. R. Filippi, M. Scorsetti, S. Hui, and L. P. Muren. Total marrow and total lymphoid irradiation in bone marrow transplantation for acute leukaemia. *Lancet Oncol*, 2020.
- [30] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology. 2018.
- [31] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? 2014.
- [32] J. Zhang, X. Ding, D. Hu, and Y. Jiang. Semantic segmentation of covid-19 lesions with a multiscale dilated convolutional network. *Nature*, 2022.
- [33] T. Zhou¹, S. Canu, and S. Ruan. A review: Deep learning for medical image segmentation using multi-modality fusion. 2020.

A | Humanitas Dataset Information

Number of patient containing specific delineated OARs

	OAR	Number of patients
1	Heart	98
2	Esophagus	64
3	Liver	96
4	Marrow	54
5	Spleen	12
6	Right Lung	86
7	Left Lung	87
8	Right Kidney	95
9	Left Kidney	95
10	Thyroid	89
11	Larynx	85
12	Oral Cavity	95
13	Right Eye	95
14	Left Eye	95
15	Right Crystalline	90
16	Left Crystalline	90
17	Right Parotid	92
18	Left Parotid	92
19	Brain	97
20	Rectum	95
21	Bladder	96
22	Stomach	94

Table A.1: Number of patient containing specific delineated OARs

	OAR	Number of patients
23	Intestine	93
24	Testicles	57
25	PTV Abdomen	95
26	PTV Arms	93
27	PTV Legs	93
28	PTV Head	95
29	PTV Chest	94
30	PTV Total	91
31	Ribs	36

Table A.2: Number of patient containing specific delineated OARs

B | DICOM File Information

Dataset.file_meta	
(0002, 0010) Transfer Syntax UID	UI: Explicit VR Little Endian
(0008, 0005) Specific Character Set	CS: 'ISO_IR 192'
(0008, 0008) Image Type	CS: ['ORIGINAL', 'PRIMARY', 'AXIAL', 'HELIX']
(0008, 0012) Instance Creation Date	DA: ''
(0008, 0013) Instance Creation Time	TM: ''
(0008, 0016) SOP Class UID	UI: CT Image Storage
(0008, 0018) SOP Instance UID	UI: 2.25.1000777134112839911018461211032725607228
(0008, 0020) Study Date	DA: ''
(0008, 0021) Series Date	DA: ''
(0008, 0022) Acquisition Date	DA: ''
(0008, 0023) Content Date	DA: ''
(0008, 0030) Study Time	TM: ''
(0008, 0031) Series Time	TM: ''
(0008, 0032) Acquisition Time	TM: ''
(0008, 0033) Content Time	TM: ''
(0008, 0050) Accession Number	SH: ''
(0008, 0052) Query/Retrieve Level	CS: 'IMAGE'
(0008, 0054) Retrieve AE Title	AE: ''
(0008, 0060) Modality	CS: 'CT'
(0008, 0070) Manufacturer	LO: 'Philips'
(0008, 0080) Institution Name	LO: ''
(0008, 0090) Referring Physician's Name	PN: ''
(0008, 1010) Station Name	SH: ''
(0008, 1030) Study Description	LO: ''
(0008, 103e) Series Description	LO: ''
(0008, 1040) Institutional Department Name	LO: ''
(0008, 1048) Physician(s) of Record	PN: ''
(0008, 1070) Operators' Name	PN: ''
(0008, 1090) Manufacturer's Model Name	LO: 'Brilliance Big Bore'
(0010, 0010) Patient's Name	PN: '0c8aeb38b9^^'
(0010, 0020) Patient ID	LO: '0c8aeb38b9'
(0010, 0030) Patient's Birth Date	DA: ''
(0010, 0032) Patient's Birth Time	TM: ''
(0010, 0040) Patient's Sex	CS: ''
(0018, 0022) Scan Options	CS: 'HELIX'
(0018, 0050) Slice Thickness	DS: "5.0"
(0018, 0060) KVP	DS: "120.0"
(0018, 0090) Data Collection Diameter	DS: "600.0"
(0018, 1020) Software Versions	LO: '2.3.0'
(0018, 1100) Reconstruction Diameter	DS: "600.0"
(0018, 1120) Gantry/Detector Tilt	DS: "0.0"
(0018, 1130) Table Height	DS: "148.0"
(0018, 1140) Rotation Direction	CS: 'CW'

(0018, 1151) X-Ray Tube Current	IS: "413"
(0018, 1152) Exposure	IS: "300"
(0018, 1160) Filter Type	SH: 'C'
(0018, 1210) Convolution Kernel	SH: 'C'
(0018, 5100) Patient Position	CS: 'HFS'
(0020, 000d) Study Instance UID	UI: 2.25.165845434312740816543162347140247700983
(0020, 000e) Series Instance UID	UI: 2.25.2236245083127622133926042666532330381121
(0020, 0010) Study ID	SH: ''
(0020, 0011) Series Number	IS: "2"
(0020, 0012) Acquisition Number	IS: None
(0020, 0013) Instance Number	IS: "104"
(0020, 0032) Image Position (Patient)	DS: [-300, -193, -70.5]
(0020, 0037) Image Orientation (Patient)	DS: [1, 0, 0, 0, 1, 0]
(0020, 0052) Frame of Reference UID	UI: 2.25.293983793812338344177396391983502987841
(0020, 1040) Position Reference Indicator	LO: ''
(0020, 1041) Slice Location	DS: "-70.5"
(0020, 4000) Image Comments	LT: ''
(0028, 0002) Samples per Pixel	US: 1
(0028, 0004) Photometric Interpretation	CS: 'MONOCHROME2'
(0028, 0010) Rows	US: 512
(0028, 0011) Columns	US: 512
(0028, 0030) Pixel Spacing	DS: [1.171875, 1.171875]
(0028, 0100) Bits Allocated	US: 16
(0028, 0101) Bits Stored	US: 12
(0028, 0102) High Bit	US: 11
(0028, 0103) Pixel Representation	US: 0
(0028, 1050) Window Center	DS: "60.0"
(0028, 1051) Window Width	DS: "330.0"
(0028, 1052) Rescale Intercept	DS: "-1000.0"
(0028, 1053) Rescale Slope	DS: "1.0"
(7fe0, 0010) Pixel Data	OW: Array of 524288 elements

List of Figures

2.1	TBI vs TMLI	6
2.2	Artificial perceptron and biological neuron.	7
2.3	3D Visualization of a Loss function landscape.	9
2.4	CNN composed of Encoder, bottleneck and Decoder.	10
2.5	Transfer performance study	12
2.6	Workflow of a typical CAD system.	14
3.1	Random structSeg slice	18
3.2	Random segthor slice	19
3.3	Random targetDS slice	19
3.4	3d ptv tot	20
3.5	Values distribution in all datasets	21
3.6	Example of original and cropped slice.	23
3.7	Example of original and elastic deformed image.	24
3.8	Example of original and grid distorted image.	24
3.9	Example of original and rotated image.	25
4.1	High level architecture of the Unet Neural Network	29
4.2	SE-ResBlock structure	30
4.3	DeepLabV3 encoder structure	31
4.4	Small organ inference architecture	33
4.5	Example of window retrieval starting from the coarse segmentation result of an esophagus.	35
4.6	Last Layer Feature Fusion	36
4.7	Learning rate decay during the epochs (Shown in logarithmic scale for visualization purposes).	38
4.8	Confusion Matrix.	40
4.9	Jaccard Index seen as Intersection Over Union (IoU).	41
6.1	Validation indexes evolution during training over the target: Oral Cavity.	52
6.2	Validation indexes evolution during training over the target: Heart.	53

6.3	Validation indexes evolution during training over the target: Left Lung. . .	53
6.4	Validation indexes evolution during training over the target: Right Lung. .	54
6.5	Validation indexes evolution during training over the target: PTV Total. .	54
6.6	The prediction (in green) and the ground truth (in red) of the target: Oral Cavity. Notice that the overlap between prediction and ground truth is shown in yellow.	55
6.7	The prediction (in green) and the ground truth (in red) of the target: Heart.	55
6.8	The prediction (in green) and the ground truth (in red) of the target: Left Lung.	56
6.9	The prediction (in green) and the ground truth (in red) of the target: Right Lung.	56
6.10	The prediction (in green) and the ground truth (in red) of the target: PTV Total.	57
6.11	Validation indexes evolution during training over the target Esophagus. . .	58
6.12	Validation indexes evolution during training over the target Marrow. . . .	58
6.13	Validation indexes evolution during training of the target Esophagus. . . .	59
6.14	Validation indexes evolution during training of the target Marrow.	60
6.15	The prediction (in green) and the ground truth (in red) of the target: Esophagus. Notice that the overlap between prediction and ground truth is shown in yellow.	60
6.16	The prediction (in green), the ground truth (in red) of the target: Marrow. Notice that the overlap between prediction and ground truth is shown in yellow.	61
6.17	The prediction (in green) and the ground truth (in red) of the target: Left Lung. Notice that the overlap between prediction and ground truth is shown in yellow.	64
6.18	The prediction (in green) and the ground truth (in red) of the target: Right Lung. Notice that the overlap between prediction and ground truth is shown in yellow.	66
6.19	The prediction (in green) and the ground truth (in red) of the target: Heart. Notice that the overlap between prediction and ground truth is shown in yellow.	68
6.20	The prediction (in green) and the ground truth (in red) of the target: Marrow. Notice that the overlap between prediction and ground truth is shown in yellow.	70

List of Tables

5.1	OARs segmented with a standard Unet	44
5.2	OARs identified as Small Organ	45
5.3	OaRs used for the transfer learning experiments and the relative experimental setups.	47
5.4	Summary of the various experiments carried out in the transfer learning setting.	48
5.5	Summary of the various experiments carried in the multiclass modality. . .	49
5.6	Summary of the various experiments carried using the Last Layer Feature Fusion ensemble method.	50
6.1	Result of Binary segmentation from scratch over some targets.	52
6.2	Result of Binary segmentation from scratch over the targets: Esophagus, Marrow.	57
6.3	Result of Binary segmentation from scratch over the targets: Esophagus and Marrow.	59
6.4	Average Validation Performance of different experimental setups in the small organ context for the target: Esophagus	61
6.5	DSC and Jaccard Index values of the different single class transfer leaning settings. FL: frozen layers	63
6.6	DSC and Jaccard Index values of the different single class transfer leaning settings. FL: frozen layers	65
6.7	DSC and Jaccard Index values of the different single class transfer leaning settings. FL: frozen layers	67
6.8	DSC and Jaccard Index values of the different single class transfer leaning settings. FL: frozen layers	69
6.9	DSC and Jaccard Index values of the different targets segmented from scratch using the DeepLabV3 model in the multiclass setup.	71
6.10	DSC and Jaccard Index values of the different targets segmented from scratch using the DeepLabV3 model in the multiclass setup.	72

6.11 DSC and Jaccard Index values of the different targets segmented from scratch using the DeepLabV3 model in the multiclass setup.	73
6.12 DSC and Jaccard Index values of the different targets segmented using the Last Layer Feature Fusion ensemble method.	74
6.13 DSC and Jaccard Index values of the different targets segmented using the Last Layer Feature Fusion ensemble method.	74
6.14 DSC and Jaccard Index values of the different targets segmented using the Last Layer Feature Fusion ensemble method.	75
A.1 Number of patient containing specific delineated OARs	86
A.2 Number of patient containing specific delineated OARs	87

List of Symbols

Variable	Description
TBI	Total Body Irradiation
TMLI	Total Marrow and Lymph node Irradiation
CNN	Convolutional Neural Networks
ANN	Artificial Neural Network
ML	Machine Learning
DL	Deep Learning
AI	Artificial Intelligence
CT	Computed Tomography
OAR	Organ At Risk
CTV	Clinical Target Volume
ReLU	Rectified Linear Unit
ASPP	Atrous Spatial Pyramid Pooling
DSC	Dice Similarity Coefficient
TP	True positive
TN	True negative
FP	False positive
FN	False negative
IoU	Intersection over Union
DS	Dataset
N\A	Not Applicable
FCN	Fully Convolutional Neural Network
GAN	Generative Adversarial Network
FL	Frozen Layers

Acknowledgements

I would like to thank first of all the Professor Daniele Loiacono and the doc. Leonardo Crespi for their help, disponibility, and patience. Big thanks to my family, more importantly to my parents: mom and dad, you never stopped believing in me and you supported me in all these years. Special thanks to Cristina, always there to listen and to comfort me during both highs and downs, thanks for being part of my life and for your patience during my stressful moments. Thanks also to my colleagues here at the university at "Tavolo 5", you guys were always there if I needed help, and you helped me alleviate the stress of incoming exams. Thanks to all my friends all over the world, and to everybody that in some way enriched my life.

