POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Ensemble methods for multi-organ semantic segmentation

Laurea Magistrale in Computer Science Engineering

Author: Paolo Roncaglioni

Advisor: Prof. Daniele Loiacono

Co-advisor: Leonardo Crespi

Academic year: 2021-2022

---

1. Introduction

Radiation therapy is a common choice for cancer treatment, as it can efficiently destroy cancer cells with external radiation rays. Therefore, developing an effective treatment plan is vital for radiation therapy. A good plan allows you to determine the appropriate distribution of radiation doses in target tumors and nearby organs called organs-at-risk (OaRs). To be effective, the radiation dose must be decided so that the cancer cells receive enough dose to be completely killed. On the other hand, the OaRs, which are very sensitive to radiation, must be protected to avoid harm to the patient in healthy tissues. The accurate delineation and segmentation of these OaRs is therefore of great importance for proper treatment planning, but manual techniques are still too slow and prone to errors. The developments of recent years in the field of Machine Learning, more particularly in Deep Learning, have made it possible to make great strides in this particular task, called Semantic Segmentation.

One of the most important factors to obtain a good segmentation is to use a large dataset. However, it is often not possible to easily obtain complete datasets with multi-organ labels, as the images available in most datasets are not completely annotated: sometimes it is not necessary to segment all the organs for each patient or it is not useful for the therapy that is being used for treatment. It is therefore not unusual to have several different datasets available (even from the same source) with annotations of a single organ, but in a much smaller number images with complete multi-organ annotations.

Taking inspiration from multi-modal segmentation architectures [7], in which different medical image acquisition modes (e.g. CT or MRI) contribute to the final segmentation, we propose three ensemble models for multi-organ segmentation. These models combine the results of several networks for different architectures trained on single-organ or multi-organ dataset. In particular, we propose a detailed study regarding the use of these architectures in settings that we think are relevant: availability of several trained networks (single class or multi-class), training dataset for some targets different from the test dataset, and very small multi-class dataset available. We then evaluate all the results by comparing them with different metrics against meth-ods we identified as baselines: three multiclass networks with Unet, SE-ResUnet and DeepLabV3 architectures.

2. Related works

Semantic segmentation in the medical domain is a research field that has evolved rapidly in recent years. Atlas-based solutions are widely used in commercial environments, but among the different approaches currently in the state of the art we distinguish 6 different categories of network: Auto-encoders (AE), convolutional (CNN), fully convolutional (FCN), generative adversarial networks (GAN) and Regional-CNN.
This work is based on multiple FCN. In this kind of models the last convolutional layer (introduced by Shelhamer et al. [4]) is the main difference between standard CNN. Thanks to the improvement of deconvolutional kernels used to up-sample the feature map, FCN allows you to output a dense voxel-wise prediction even from an entire volume. One of the most famous state of the art structures for segmentation is the U-net, proposed by Ronnenberg et al. 2015, in which the concept of deconvolution is used [5].
Ensemble learning techniques combine different individual models in order to obtain better results than individual models. Deep ensemble learning models combine the advantages of both deep learning models and ensemble learning in order to have a final model with a better generalization performance. In this kind of approach we can find techniques such as Bagging, Boosting e Stacking. Specifically, stacking can be done either by combining outputs of multiple base models in some fashion or using methods to choose the "best" base model. Stacking is one of the integration techniques where the meta-learning model is used to integrate the output of base models [6].
In the literature different way to combine models have been proposed, based on different fusion approaches: input-level, layer-level and decision-level fusion strategy [7]. These types of approaches are commonly used in models involving training on multi-modal data (e.g. models trained on CT scan and MRI scan simultaneously) even on different datasets. This work relies heavily on the same blending methods applied to different classes as well.

3. Datasets

In this work we mainly focused on analyzing the medical images acquired with Computed Tomography (CT). Pixels in an image obtained by CT scanning are displayed in terms of relative radiodensity (measured in Housenfield Units) with range values from +3.071 to -1.024 depending on the considered tissue.
Our reference datasets are therefore 2 public datasets belonging to the homonymous organ segmentation challenges in CT scan: task 3 of Structseg2019 (with annotations of lung, heath, trachea, esophagus, spinal cord) and SegTHOR2019 (heart, aorta, trachea, esophagus). StructSeg is composed of 50 volumes belonging to different patients, 77 slices per volume for a total of 3861 scan/ground truth mask pairs while in SegTHOR there are 7390 total slices in 40 volumes and 185 average slices. Each slice has a size of 512x512 on a single channel.

3.1. Preprocessing

In any work on deep learning there is almost always some sort of preprocessing on the datasets required to clean or transform the data in a way that suits the experiments. Working on CT scan, the first thing to do is to adjust the range of the Housenfield Units (HU) characterized by the parameters of *width* (contrast) and *level* (brightness). This process is called windowing. Table 1 shows the used values.

Tissue	Width	Level
Lungs	1500	-600
Heart	350	50
Esophagus	300	80
Trachea	1200	-440
Spinal Cord	600	0

Table 1: Hounsfield window according to tissue type

Finally, we normalize every single slice and perform a cropping to reduce the images to 320x320 in order to cut away most of the background, which is not useful for our purpose.

4. System Design

Before starting the training we transform the format from DICOM to HDF5 for ease of processing.

4.1. Augmentations

The augmentations used in this work are applied in an online approach with a probability of 50%, and randomized parameters in a specific range. We use non-rigid transformations to help generalize the models. In our pipeline we have included elastic transformation, grid distortion and rotation in order to simulate the possible difference between different patients and different characteristics.

4.2. Architectures

We based our architectures on some specific segmentation Fully Convolutional Networks (FCNs). These are used in combination with each other in the ensemble to refine the final segmentation masks. The architectures chosen are: Unet, SE-ResUnet and DeepLabV3. Unet and SE-ResUnet both use the usual encoder-decoder structure [5], although the latter uses a different type of building block called the "Squeeze and Excitement" block (SE) [3]. DeepLabV3 [2] has a multi-level pyramid structure contraction/expansion approach with atrous convolution to effectively preserve spatial information.

4.3. Ensemble methods

The models we implemented use the results of the FCNs mentioned. We distinguish four different ensemble approaches:

Argmax of binary outputs: used as the main baseline of this work, it is the pixel-wise combination of the best results of the single binary networks used. In the base case, we used a binary network for each organ which will output a single-channel probability mask. For each pixel of each mask then we take the best result and associate it with the corresponding class. The operation is therefore an argmax of C classes for each *(x, y)* pixel on the set of masks M:

$$\arg\max_{C} M_c(x, y)$$

The result will be a complete multi-organ segmentation mask.

1x1 convolution ensemble stacking: using a 1x1 convolution (usually used in the final layer of convolutional networks) on top of the stacked outputs of binary networks allows to weight an input mask depending on the target. This method is also more meaningful in the results: intuitively, a higher value associated with a input mask means that the corresponding organ is evaluated more than the other inputs.

Unet ensemble stacking: stacking (also called meta-ensembling) is a model ensembling technique used to combineinformation from multiple predictive models to generate a new model. In this work we use an FCN network, specifically a Unet, to obtain a multiclass segmentation starting from the logit masks of the single-class FCNs in input.

Last-layer feature fusion: This method is part of layer-level fusion strategies, in which individual well-trained networks are combined through the concatenation of the learned individual feature representations of a specific layer. Depending on the network, the features are taken from what we have identified as the last layer before the decision and fed to a 1x1 convolution for dimensional reduction. This leads to having a number of learnable parameters equal to the number of concatenated features * the number of classes in output (+ biases). Due to the different structure of the architectures, the number of features that are extracted from Unet and SE-ResUnet networks amounts to 64 for both, and 256 for DeepLab.

In this work we will use abbreviations for each network: feature fusion (Last-layer feature fusion), Unet meta-model (Unet ensemble stacking) e 1x1 convolution (1x1 convolution ensemble stacking).

4.4. Training

For the training phase we used a standard CrossEntropy loss for ensemble networks, while for binary networks we use a combination of Binary CrossEntropy and Dice Loss. We use an Early stopping system with patience of 5 epochs and calculation of the validation loss in Moving Average ($alpha = 0.5$) to smooth the curve:

$$Loss_{validation} = \alpha L_{MA} + (1 - \alpha) L_{val}$$

For memory reasons the batch is set to 2. The initial Learning rate value is set to 0.001. The dropout rate of the segmentation networks has a value of 0.1. ReduceLROnPlateau was chosen as the scheduler for managing the learning rate, with parameters$= 0.001$, patience $= 4$ andweight decay $=$ 1e-7.

To choose the binary and multiclass networks to

be used in the ensemble experiments, we used the networks that appear to have the best performance in validation phase.
It is important to underline that the input of each single-organ network is the CT scan with Housenfield Units window optimal for the target organ.
The training of the binary models took place mainly on online hardware. In particular, we used an NVIDIA Quadro P4000 with 8GB of memory for single-class models and the ensemble models on a 6GB GeForce GTX 1060.

4.5. Evaluation

The metrics used for the evaluation and comparison of the output masks are the most used in literatures: Dice Similarity Coefficient (DSC) to measure the overlap between two sample, Hausdorff Distance 95 (HD95) is the 95th percentile of the maximum distance (in mm) of one set from the closest points of another, Precision and Recall are the metrics obtained from the Confusion Matrix.

5. Design of Experiments

The main part of this work is to test and compare the behavior of the ensemble architectures in settings relevant for the field of semantic segmentation. A total of 25 experiments were performed for as many trained ensemble networks, in addition to our baseline FCNs multi-class and argmax.
To facilitate the reading we assign a number to each experiment (Tx), where x is the test number. Table 2 shows the baselines of our experiments.

Model	Dice	HD95
Unet	0.8243 (0.779,0.874)	4.5729 (2.615,9.848)
SE-ResUnet	0.8172 (0.776,0.864)	6.3475 (2.528,16.469)
DeepLabV3	0.8512 (0.799,0.872)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	2.4454 (1.344,3.236)

Table 2: Baselines - Average performances on test set

T1 - Basic training: Pre-trained networks are used in a binary segmentation problem on a single organ, then combined in order to obtain the desired multiclass segmentation. This type of model is trained using the binary networks which have the best result on the validation set.

Model	Dice	HD95
Feature Fusion	0.8744 (0.836,0.913)	2.2607 (1.275,2.967)
1x1 convolution	0.8794 (0.851,0.913)	2.2941 (1.305,3.087)
Unet meta-model	0.8661 (0.828,0.898)	2.3665 (1.413,2.962)

Table 3: Experiment 1 - Average performances on test set

T2 - Multiple nets on same organ: This experiment is based on the fact that one of the main problems of semantic segmentation on medical images is the detection of small organs and tissues and, given the nature of CT scans, often with little contrast to the surrounding tissues. Using single nets trained on one organ can help alleviate this problem, but we want to see what happens if we use multiple binary nets trained on the same organ to ensemble in a multiclass problem. We used multiple nets for esophagus and trachea only, the most difficult organs to delineate. Table 4 and 5 show the results of this experiment.

Model	Dice	HD95
Feature Fusion	0.8807 (0.844,0.92)	2.1333 (1.138,2.982)
1x1 convolution	0.8824 (0.856,0.918)	2.133 (1.305,2.669)
Unet meta-model	0.8414 (0.804,0.863)	2.6101 (2.068,3.625)

Table 4: Experiment 2 (esophagus) - Average performances on test set

Model	Dice	HD95
Feature Fusion	0.8848 (0.848,0.918)	2.2529 (1.167,3.122)
1x1 convolution	0.8809 (0.845,0.912)	2.3353 (1.236,3.11)
Unet meta-model	0.8795 (0.843,0.905)	2.3714 (1.413,3.166)

Table 5: Experiment 2 (trachea) - Average performances on test set

T3 - Multiple nets trained on different dataset: SegThor dataset has some organs in common with StructSeg. We use networks trained on these common organs in our ensemble models, instead of networks trained on the same inference dataset. In this case we see how the ensemble architectures behave when one of the targets does not have a binary dataset on which to train the network, thus having to use an external dataset. Again, we tested only the most difficult organs i.e. trachea and esophagus. Tables 6 and 7 show the results of this experiment.

Model	Dice	HD95
ArgMax	0.6753 (0.647,0.739)	6.7791 (5.669,8.185)
Feature Fusion	0.8678 (0.837,0.901)	2.8196 (2.452,3.16)
1x1 convolution	0.7463 (0.717,0.778)	21.603 (20.971,22.046)
Unet meta-model	0.7419 (0.706,0.775)	21.6602 (21.04,22.264)

Table 6: Experiment 3 (esophagus) - Average performances on test set

Model	Dice	HD95
ArgMax	0.7578 (0.737,0.772)	7.6132 (5.419,10.289)
Feature Fusion	0.8158 (0.768,0.836)	4.412 (2.793,7.445)
1x1 convolution	0.7597 (0.74,0.775)	21.7617 (21.069,22.6)
Unet meta-model	0.7823 (0.769,0.803)	7.3663 (5.52,10.134)

Table 7: Experiment 3 (trachea) - Average performances on test set

T4 - Multi-class and multi-binary: The ease with which it is possible to add inputs to our ensemble architectures allows us to experiment with compositions of multiple segmentation networks, not limited to binary ones. In fact we also test the unique combination between the binary networks used for the base plus a multiclass network, chosen on the basis of the best performing one among those trained. Tables 8 shows the results of this experiment.

Model	Dice	HD95
Feature Fusion	0.88 (0.837,0.921)	2.0747 (1.122,2.623)
1x1 convolution	0.8815 (0.855,0.914)	2.1591 (1.305,2.831)
Unet meta-model	0.8668 (0.829,0.904)	2.2844 (1.305,3.007)

Table 8: Experiment 4 - Average performances on test set

T5 - Reduced training dataset: This is to test how the models behave on a hypothetical reduced ensemble dataset. The binary networks used are trained on the complete dataset, while the ensemble networks use 20% of the training dataset. This is to test simulation of a situation where you have large binary datasets available but a very limited number of labeled multiclass samples. Table 9 shows the results of this experiment.

Model	Dice	HD95
Feature Fusion	0.8681 (0.827,0.904)	2.3664 (1.236,3.597)
1x1 convolution	0.8793 (0.851,0.913)	2.2978 (1.305,3.082)
Unet meta-model	0.876 (0.848,0.911)	2.3085 (1.443,2.862)

Table 9: Experiment 5 - Average performances on test set

Table 10 shows the best FCNs used in the experiments.

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet + DeeplabV3
Trachea	DeeplabV3 + Unet
Heart	DeeplabV3
Spinal Cord	SE-ResUnet

Table 10: Best networks per organ

6. Results

In general, an ensemble architecture allows for better performances than the classic state-of-the-art FCN networks identified as baseline in this work.

Although very simple, the Argmax model has better results than multiclass FCN models. This is also due to the fact that the single model can better exploit the unique information of the corresponding organ, even if the spatial information on the mutual position of all the organs is lost. We think the result is also due to the finer-grained preprocessing, which allows to have images with different improvements depending on the target organ of the single-class network. In some cases, such as the T3 experiment on the esophagus, a worsening of a single input mask corresponding to one organ can also compromise the segmentation performance on different (and closely related) organs, such as the trachea in this case.

Using a Unet meta-model does not always give the best results: often the performances are equal to or lower than the argmax results, and there doesn't seem to be too much difference in using additional networks to improve their segmentation. In some cases, such as the T4 experiment, a worse input for a certain organ can also lead to a lower performance on another organ, apparently unrelated.

The 1x1 convolution network allows for good performances on average even if has a simple structure. Thanks to its intuitive structure and the few learnable parameters (49 for a network with 7 inputs and 7 outputs + biases), it has good results even in the case of a very small dataset. In some cases, however, the evaluation metrics drop to zero when there is not a good quality of the input from one of the binary networks (T3 trachea).

The architecture that on average performs better than the others is the feature fusion. It also performs well in the case of an input network trained on different data than the ones in test dataset, where it has much better results than the rest of the networks. In case of a small dataset it is always a good choice, but its performances are however slightly worse even than the simple argmax.

Finally, in Table 11 we see a comparison between the Dice Score of the best selection of our experiments (t and e are for trachea and esophagus) and the state of the art models presented by the winner of the MICCAI StructSeg2019 challenge [1], the source of the main dataset of this work. The network to be compared is called Cascaded-SE-ResUnet (C-SE).

	DLV3	**T1**	**T2-t**	**T2-e**	**C-SE**
1	96.88	97.05	97.66	96.72	97.01
2	96.52	96.65	96.70	97.00	96.03
3	91.74	93.27	92.91	92.77	94.49
4	61.70	71.78	76.46	71.73	80.69
5	77.98	79.69	79.44	81.99	84.98
6	85.89	89.92	88.32	88.21	91.00
Avg	85.12	87.94	88.07	88.48	91.56

Table 11: Comparison between our selected best models vs Cascaded-SE-ResUnet (C-SE), the StructSeg2019 winner [1]. DLV3 is our trained DeepLabV3, while the best nets by experiments are: 1x1 convolution (T1), Feature Fusion (T2-e, T2-t).

Organs are shown with their code: [1, 2, 3, 4, 5, 6] for [Left Lung, Right Lung, Heart, Trachea, Esophagus, Spinal Cord].

7. Conclusions

Generated masks are accurate and are able to capture well the structure of the target organs, with a few exceptions involving esophagus and trachea (known to be the most difficult in literature). In some cases the performances show values equal to or slightly lower than [1], the winning method of the StructSeg2019 challenge. Some types of tests with different settings were performed to demonstrate the applicability of these architectures in realistic situations. The results obtained are encouraging for a future development in the direction of specialized ensemble methods, particularly in absence of large, complete datasets with multi-organ labels. With this type of dataset we have shown how these methods are able to perform better than a simple combination of single single-organ networks despite the time and computational power limitations imposed by costs and logistics.

References

[1] Zheng Cao, Bohan Yu, Biwen Lei, Haochao Yinga, Xiao Zhang, Danny Z. Chen, and Jian Wu. Cascaded se-resunet for segmentation of thoracic organs at risk. 2020.

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking

atrous convolution for semantic image segmentation. 2017.

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. 2018.

[4] Trevor Darrell Jonathan Long, Evan Shelhamer. Fully convolutional networks for semantic segmentation. 2015.

[5] P. Fischer O. Ronneberger. U-net: Convolutional networks for biomedical image segmentation. 2015.

[6] David H Wolpert. Stacked generalization. 1992.

[7] Tongxue Zhou, Su Ruan, and Stephane Canu. A review: Deep learning for medical image segmentation using multi-modality fusion. 2019.

POLITECNICO
MILANO 1863

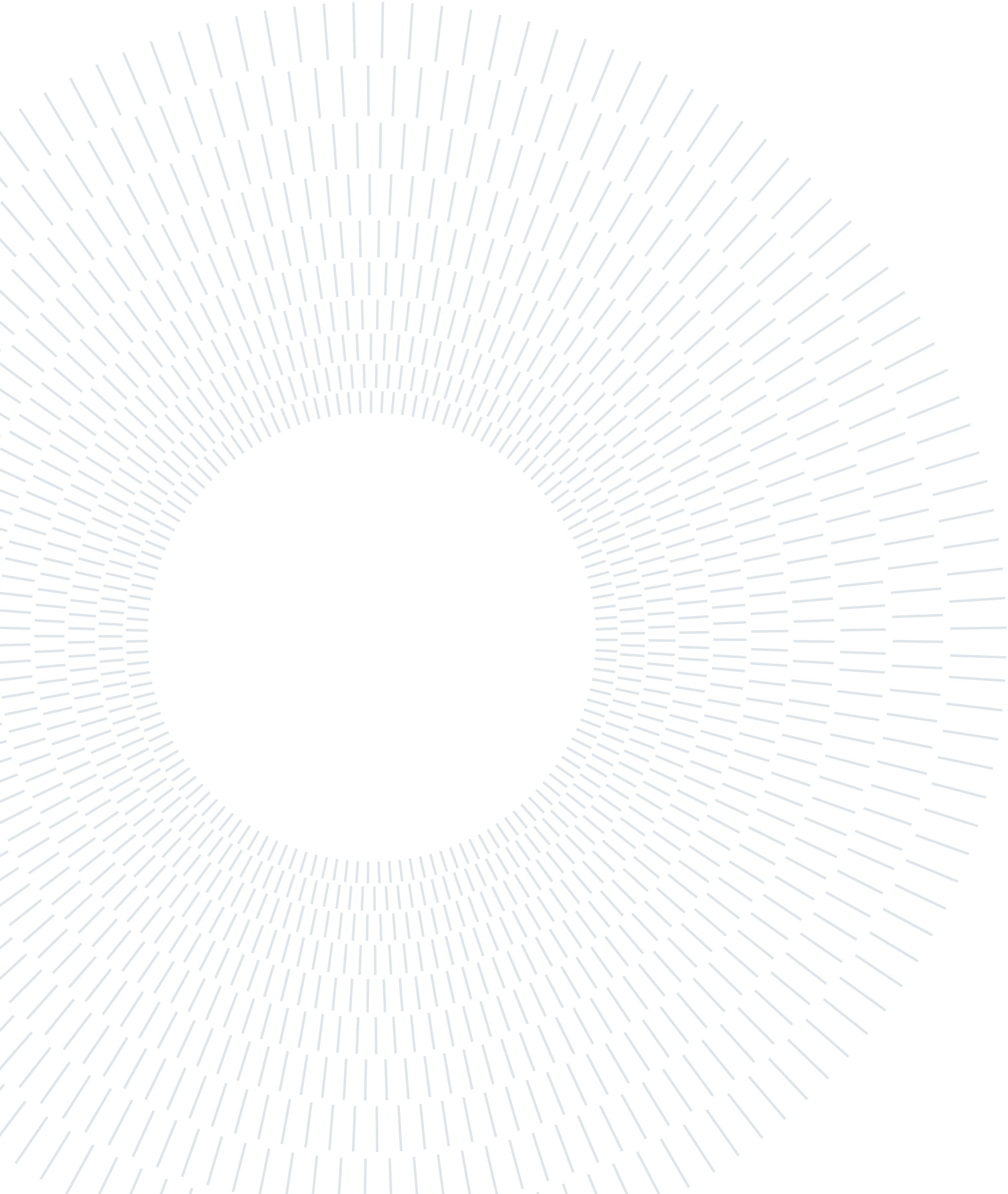
SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Ensemble methods for multi-organ semantic segmentation

Tesi di Laurea Magistrale in
Computer Science Engineering

Author: **Paolo Roncaglioni**

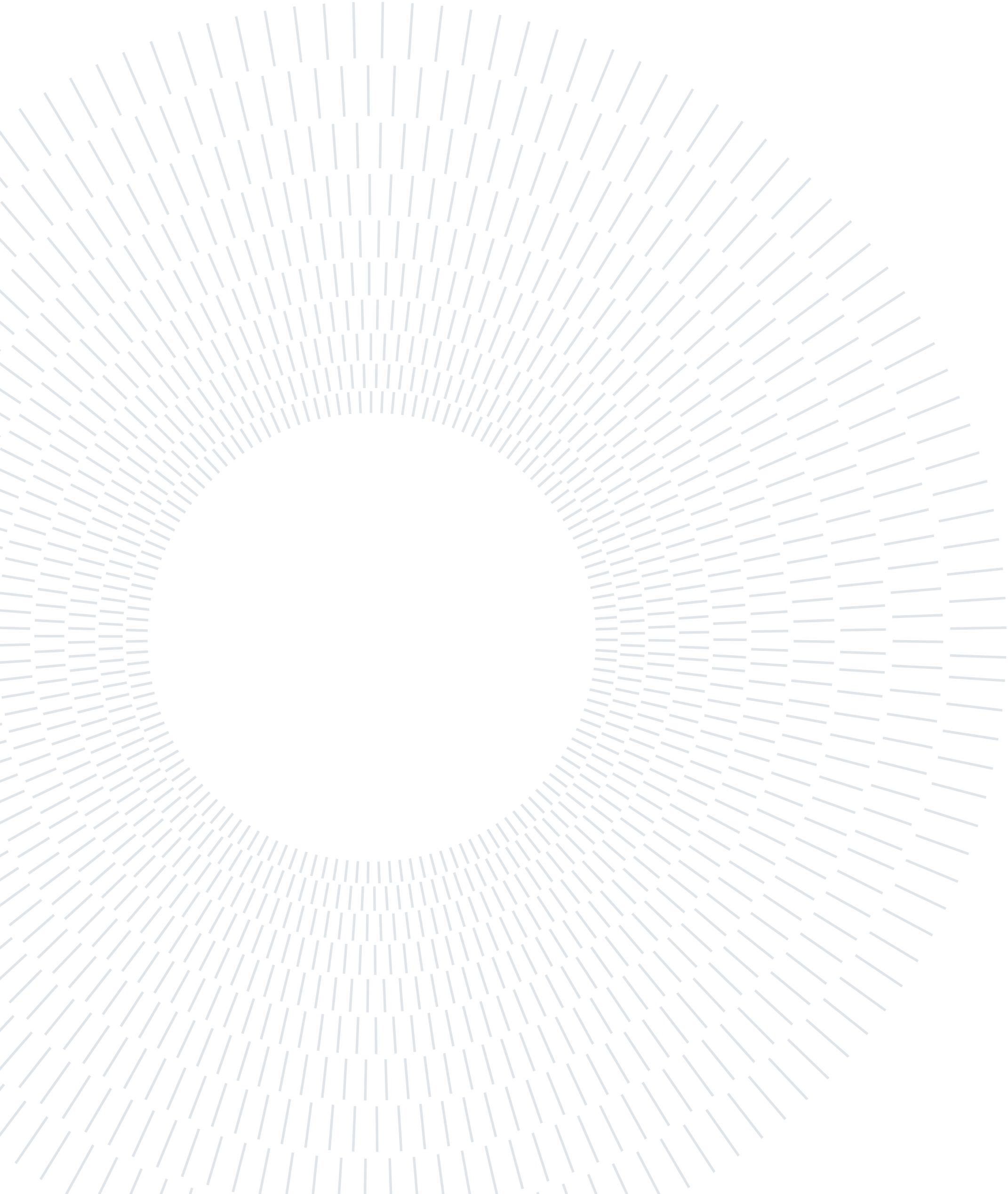
Student ID: 920555
Advisor: Prof. Daniele Loiacono
Co-advisors: Leonardo Crespi
Academic Year: 2021-22



Abstract

Multi-class segmentation has recently achieved significant performance in natural images and videos. These achievements have been made possible in large part by the public availability of large multi-class datasets. However there are certain domains, such as biomedical imaging, where obtaining sufficient multi-class annotations is laborious and often impossible, and therefore only single-organ datasets are available. While most of the existing research on segmentation in this domain uses large multi-class datasets or focuses on single-class segmentation, we propose a method that use multiple single-target datasets. We have implemented three multiple-input ensemble networks to use single-organ binary networks inference and obtain a complete multi-organ segmentation using images from 50 different patients. The metrics used for the evaluation and comparison of the different models with chosen baselines are Dice Score and Hausdorff Distance. Our work proposes a detailed study which aims to demonstrate the ability of ensemble architectures in organ segmentation. In particular, the experiments we perform concern the networks used in the ensemble: for organs that are difficult to delineate, such as the trachea and esophagus, we use more binary networks. We also see what happens if, in addition to binary networks for single organs, we use a multi-organ network. Finally we perform several tests with training of ensemble models with reduced dataset (in number of volumes). Our results show how increasing the number of networks per organ in ensemble architecture contributes to more accurate segmentation for the same organs, without compromising the performance of other organs. Furthermore, using a small number of samples in the training phase does not reduce overall performance substantially.

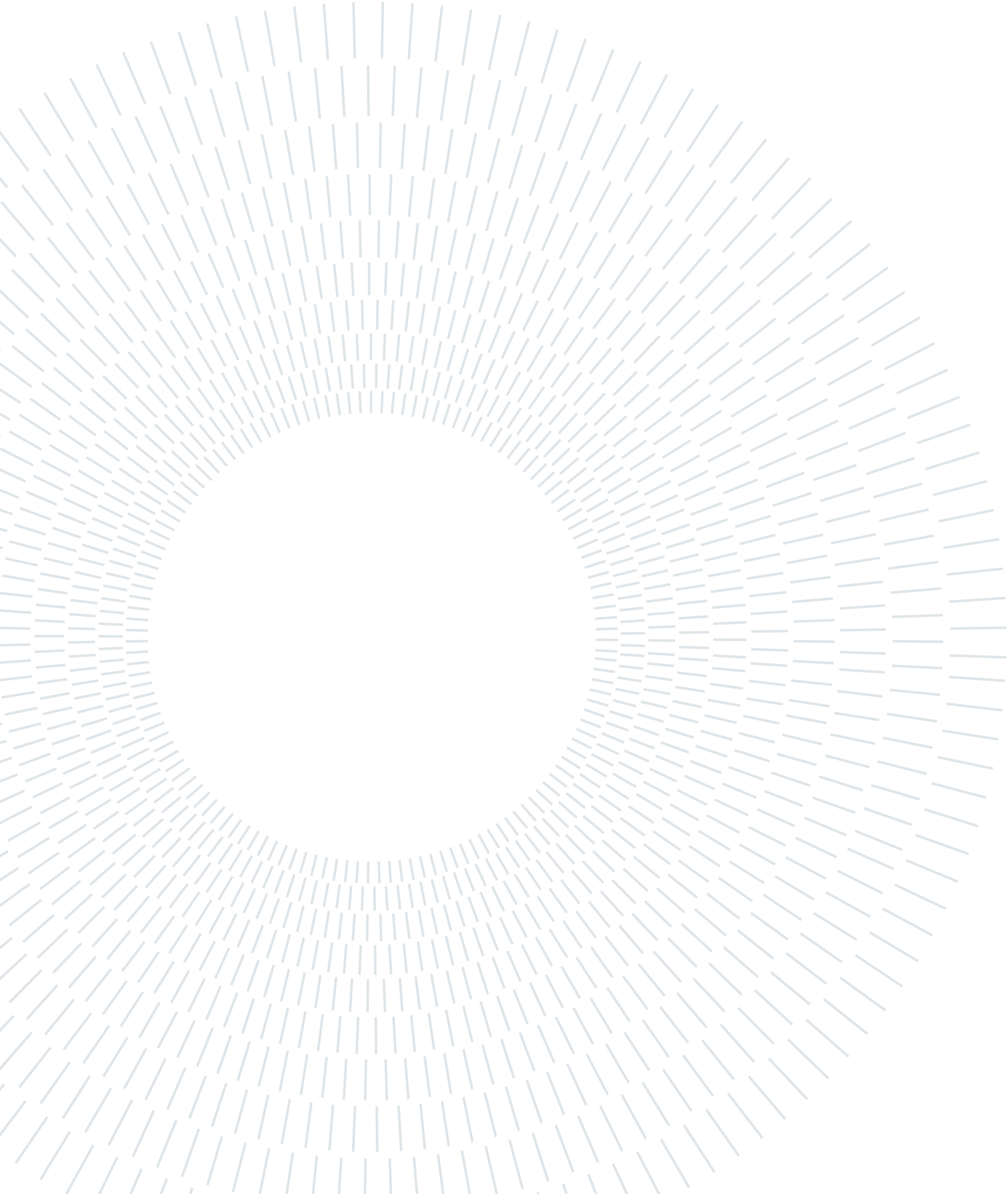
Keywords: Semantic segmentation, Ensemble, Multi-class, Organs at Risk, Deep Learning, Convolutional Neural Networks



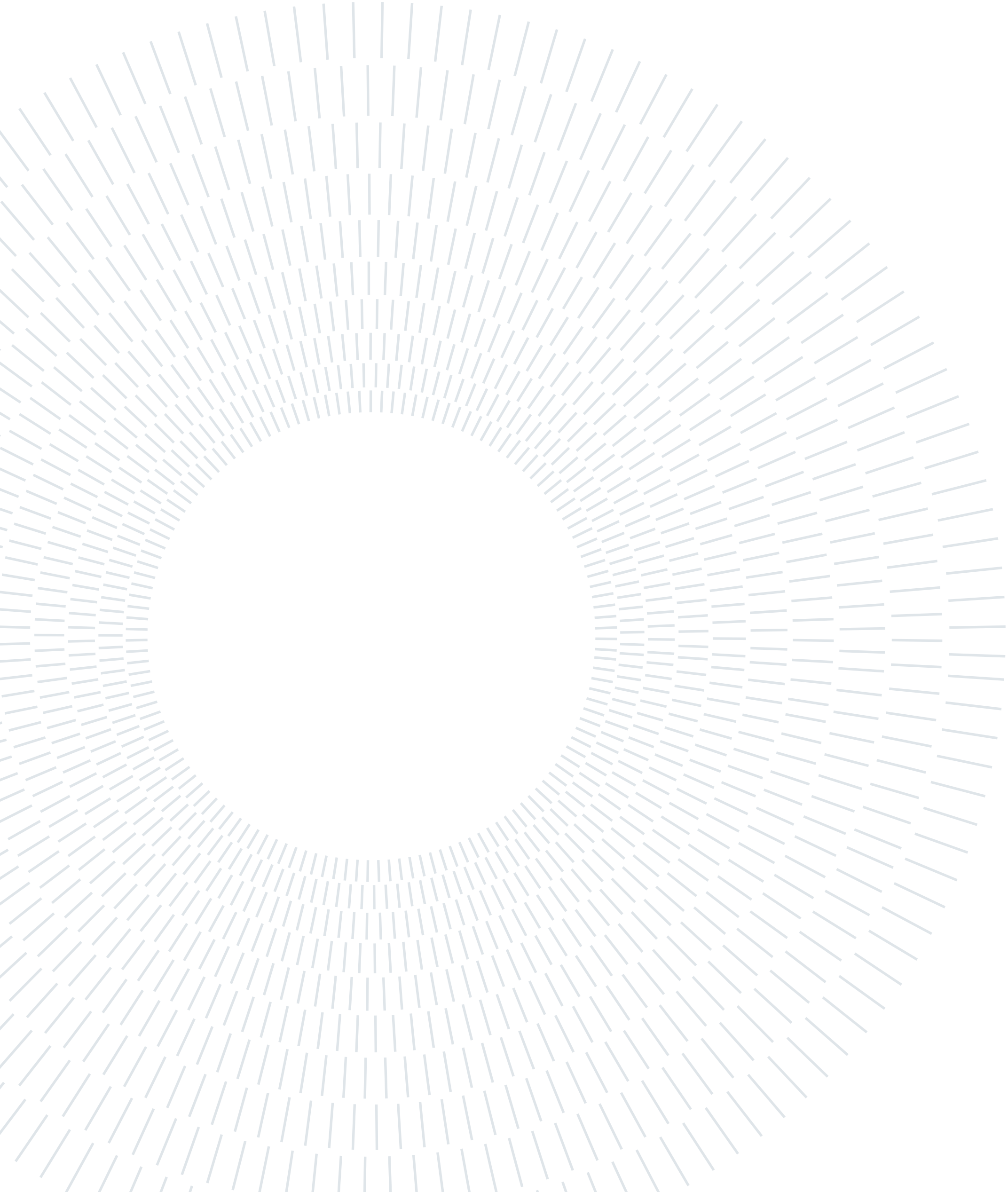
Abstract in lingua italiana

La segmentazione multi-classe ha recentemente raggiunto performance significative nelle immagini e video naturali. Questi traguardi sono stati possibili in larga parte alla disponibilitá pubblica di grandi dataset multi-classe. Tuttavia, ci sono certi domini, come quello delle immagini biomediche, dove ottenere annotazioni multi-classe sufficienti é laborioso e spesso impossibile, e quindi solo dataset con una singola classe di target sono disponibili. Mentre buona parte delle ricerche esistenti sulla segmentazione in questo dominio utilizzano grandi dataset multi-classe o si concentrano su segmentazione singola-classe, noi proponiamo un metodo che permette di utilizzare dataset multipli a singolo target. Noi abbiamo implementato tre multi-input ensemble network per utilizzare le inferenze di reti binarie a singolo organo ed ottenere una segmentazione multi-organo completa usando immagini provenienti da 50 diversi pazienti. Le metriche usate per la valutazione e la comparazione dei diversi modelli con le baseline scelte sono Dice Score e Hausdorff Distance. Il nostro lavoto propone uno studio dettagliato che mira a dimostrare l'abilitá delle architetture ensemble nella segmentazione di organi. In particolare, gli esperimenti che eseguiamo riguardano il tipo reti utilizzate nell'ensemble: per organi difficili da delineare, come trachea ed esofago, usiamo piú reti binarie. Vediamo anche cosa succede se, oltre le reti binarie per singoli organi, utilizziamo anche una rete multi-organo. Infine eseguiamo diversi test con addestramento dei modelli ensemble con dataset ridotto (nel numero di volumi). I nostri risultati mostrano come aumentare il numero di reti per organo nell'architetture ensemble contribuisca ad ottenere una segmentazione piú accurata per gli stessi organi, senza compromettere le prestazioni degli altri organi. Inoltre, utilizzare un numero ridotto di sample in fase di addestramento non riduce le prestazioni generali in modo sostanziale.

Parole chiave: Segmentazione semantica, Ensemble, Multi-classe, Organi a Rischio, Deep Learning, Reti neurali convoluzionali



Contents



1 | Introduction

Radiation therapy is a common choice for cancer treatment, as it can efficiently destroy cancer cells with external radiation rays. Therefore, developing an effective treatment plan is vital for radiation therapy. A good plan allows you to determine the appropriate distribution of radiation doses in target tumors and nearby organs called organs-at-risk (OaRs). To be effective, the radiation dose must be decided so that the cancer cells receive enough dose to be completely killed. On the other hand, the OaRs, which are very sensitive to radiation, must be protected to avoid harm to the patient in healthy tissues.

A reliable delineation and segmentation of these OaRs is therefore of great importance for proper treatment planning. Manual techniques involve radiologists or experts in a time-consuming job prone to subjective bias. In addition, due to the variability of the contour, the small size of the organs and the difficult contrast in CT scans, some organs such as Esophagus and Trachea are very difficult to delineate. To reduce treatment planning time and the total costs of radiation therapy is crucial to develop automated computer-aided OaRs segmentation methods.

The developments of recent years in the field of Machine Learning, more particularly in Deep Learning, have made it possible to make great strides in the field of Computer Vision. The competitiveness of neural networks in extracting and using information from images has favored their application in particular to the medical field.

However, almost all algorithms based on this technology have one thing in common: to work efficiently they need a huge amount of data annotated by specialists, i.e. images and segmentation masks in the field of semantic segmentation. This means that there will always be work by doctors or experts behind the annotation and delineation of organs in order to use them to develop a type of automatic segmentation.

Often, however, the images available in a dataset are not completely annotated: sometimes it is not necessary to segment all the organs for each patient or it is not useful for the therapy that is being used for treatment. It is therefore not unusual to have several different datasets available (even from the same source) with annotations of a single organ, but in a much smaller number images with complete multi-organ annotations.

It would be useful to develop a method that allows to use all the valuable information of multiple single-organ datasets in order to obtain a complete multi-organ segmentation.

1.1. Scope

Taking inspiration from multi-modal segmentation architectures [49], in which different medical image acquisition modes (e.g. CT or MRI) contribute to the final segmentation, we propose three ensemble models of multi-organ segmentation. These models combine the results of several networks of different architectures trained on single-organ or multi-organ dataset.

In particular, we propose a detailed study regarding the use of these architectures in settings that we think are relevant: availability of several trained networks (single class or multi-class), training dataset for some targets different from the test set, and very small multi-class dataset available.

We then evaluate all the results by comparing them with different metrics against methods we identified as baselines: three multi-class networks with Unet, SE-ResUnet and DeepLabV3 architectures.

The main contributions of this work are:

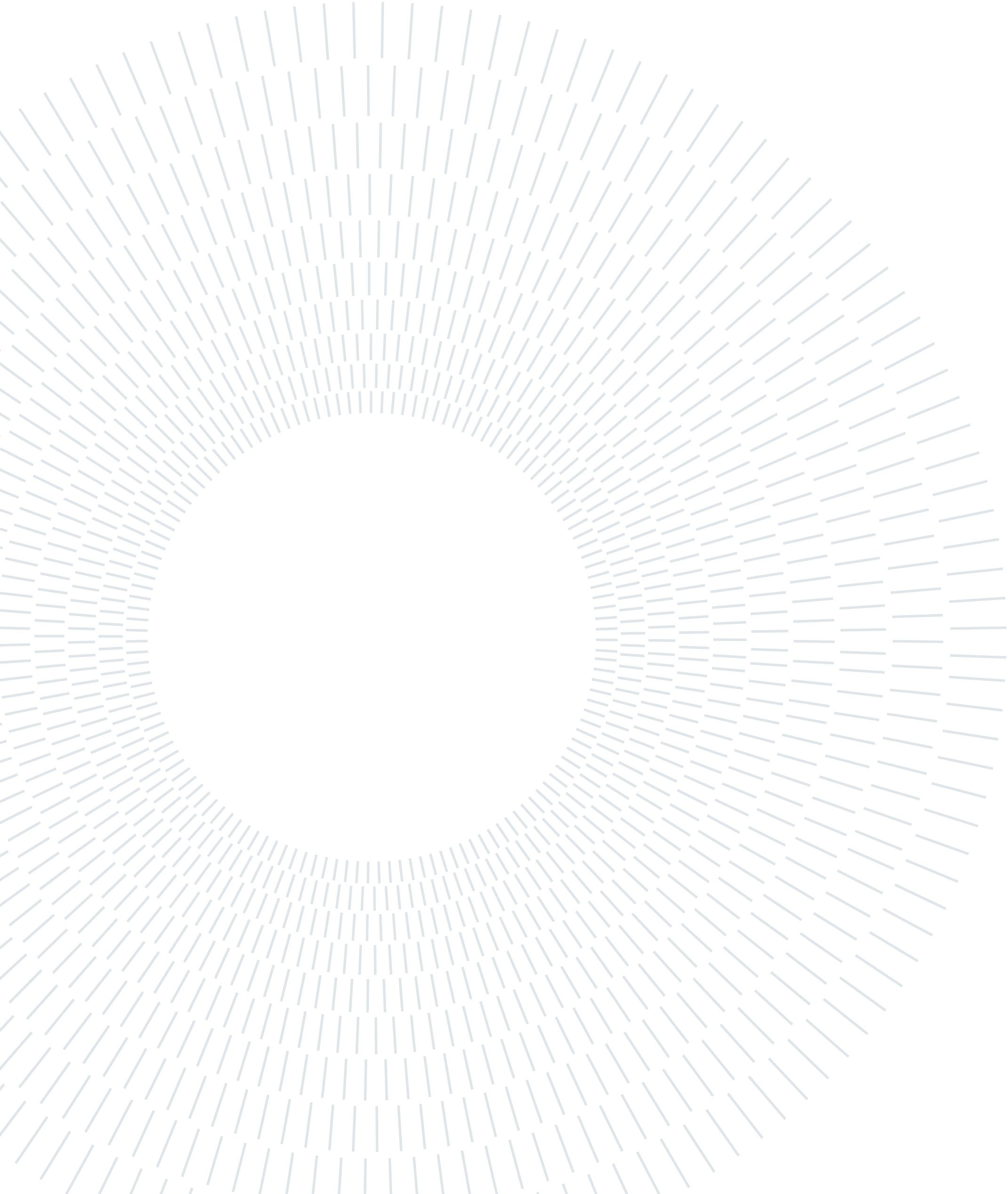
- Design and implementation of ensemble methods based on multi-modal works
- Evaluation of the behavior of the proposed methods on different settings: well-trained multiple single-class or multi-class models available, training dataset different from test set, reduced training dataset.

1.2. Thesis structure

The thesis is structured as follows:

- **Chapter 2** frames the state of the art in which the project falls and provides the reader with the appropriate knowledge to best understand the work that has been done.
- In **Chapter 3**, following a small introduction on CT scans, we present in detail the datasets on which this work was built, focusing on the training/test sets division and on the preprocessing steps performed.

- **Chapter 4** is the section where the used neural network architectures will be presented. Evaluation metrics and input pipelines are covered in this part, following some important implementation details for reproducibility and used Losses in training.
- **Chapter 5** explains the experiments were chosen and how they have been carried out using the described architectures.
- **Chapter 6** formalize the results of the experimental work.
- **Chapter 7** contains the conclusions about the work that has been done, focusing on the benefits and disadvantages of the proposed approaches and future possible improvements.



2 | State of the Art and Theoretical background

In this chapter we present a summary of the work considered to build this work. In the first part, in Section 2.1 after a brief overview of the history of artificial intelligence and Deep Learning in the medical field, we summarize the main architectures and their state of the art regarding the semantic segmentation of organs. We also describe the main problems of organ segmentation around which the greatest number of current research is clustered. We then move on to present methods and solutions regarding the ensemble and the combination of models. In Section 2.2 instead introduce all the theoretical aspects on which our work is based, giving the reader an overview of all the tools and techniques we have applied to our models, to better understand the settings in which our work took place.

2.1. Related works

2.1.1. Artificial Intelligence in Healthcare

The origins of thinking about using computer to simulate intelligent behaviour date back to 1950, when Alan Turing wondered if machines had the ability to think [16]. Only 6 years later, in 1956 during the workshop at Dartmouth College, John McCarthy officially used the term Artificial Intelligence to describe "the science and engineering of making intelligent machines [...]" [37].

The seventies saw a proliferation of AI research, especially on works regarded medicine. The international AI journal "Artificial intelligence in medicine" was founded and in 1980, the American Association for Artificial Intelligence was established with a subgroup on medical applications [45]. AI was explored and incorporated into clinical settings in the decades that followed. Fuzzy expert systems, Bayesian networks, artificial neural networks, and hybrid intelligent systems were among the new uses of AI in healthcare [15].

Among all the approaches used, one of the most interesting and most used - looking at the volume of publications - was the Artificial Neural Networks (ANN) [44]. First mentioned in 1943 by Walter Pitts and Warren McCulloch, they were a specific application of machine learning in which computation systems inspired by the biology of the neuron present in animals are used [38]. Their ability to recognize structures that are too complex or numerous to be extracted for a human, and their effectiveness in using complex relationships between different variables leads ANNs to be used in many clinical scenarios.

During the last 50 years AI has been used extensively for healthcare issues, with very rapid growth thanks largely to the increased computational power of machines and the available data that can be used to train these systems. Some, but not all, of the uses can be related to the fields of disease diagnosis, medical image analysis, prognosis, screening and drug interaction [10].

2.1.2. Deep Learning in Semantic Segmentation

As ANNs is a subfield of ML that is inspired by the organization of the brain using different layers of neurons with different weights and biases, Deep Learning (DL) is a term that identifies since 2006 the advancement of these ANNs in architecture and algorithms, specifically referred to ANNs with many hidden layers. As introuced in the Section 2.1, with the advent of processors with ever greater computing power and the arrangement of data and information like never before, Deep Learning acquired considerable importance as regards the analysis of images. Some macro-fields for this type of task include image classification, object detection, semantic segmentation, instance segmentation, and panoramic segmentation.

One of the most difficult problems in the last decade has certainly been the segmentation of images. Semantic segmentation is the understanding of an image pixel by pixel i.e. the assignment of a class for each pixel of the image. Before deep learning conquered the field of computer vision, some of the more interesting approaches involved decision trees [32] or random forest classifier [31] for segmentation using very large datasets. A popular early deep learning approach was the patch classification [21], where each pixel is separately classified using an image area around it.

In 2014 there was a real revolution with the introduction of Fully Convolutional Networks (FCN) by Long et al. [33], where the convolutional network architecture for dense predictions without fully connected layers became very popular. This allowed to generate segmentation maps for images of any size in a shorter time than even the patch classification approach. Many of the state-of-the-art approaches have since adopted this paradigm, and almost all of the state-of-the-art performances of public datasets have been achieved by deep-learning methods.

2.1.3. Deep Learning in Organ segmentation

Inspired by the success of DL in computer vision, researchers have attempted to extend these techniques to medical imaging. DL-based methods have been extensively explored in medical imaging for the purposes of segmentation, synthesis, enhancement, correction and registration.

Deep Learning-based multi-organ segmentation techniques represent a significant innovation in the delineation of guidelines in support of radiation therapy. In current practice, targets and organs at risk (OaRs) are normally delineated by experts based on CT scan images, which is a very tiring and time-consuming job. As much as the doctors who perform this labeling may have experience, it is a task often prone to errors or not totally exact delineation, especially with regard to organs such as the esophagus or trachea which are more difficult than the other organs. In the past few decades, scientists and clinicians have been engaged in a continuous effort to develop automatic contouring methods in the application of ever more accurate and consistent delineation.

Atlas-based method is certainly one of the most intuitive approaches for this kind of work, and also widely used outside the academic field, in commercial products. An atlas is defined as the combination of an image (template) and its segmented image (the atlas label). After adapting the atlas template and the target image by deformation through a process called registration, the label is propagated on the target. The accuracy of this method depends very much on the accuracy of image registration. Due to the morphological differences that different people can have for the same organ, an accurate recording is not always guaranteed [40].

DL-based multi-organ segmentation methods can be categorized into different aspects, based on properties such as network architectures, the type of training (supervised, semi-supervised, unsupervised, transfer learning), the size of the input (2D, 3D, whole volume, patch) and so on. Among the different approaches currently in the state of the art, we distinguish 6 different categories of network based on the architecture as shown in Figure 2.1: Auto-encoders (AE), convolutional (CNN), fully convolutional (FCN), generative adversarial networks (GAN) and Regional-CNN. There are also other methods called hybrids that allow you to use multiple networks for different purposes, such as image enhancement and subsequent segmentation.

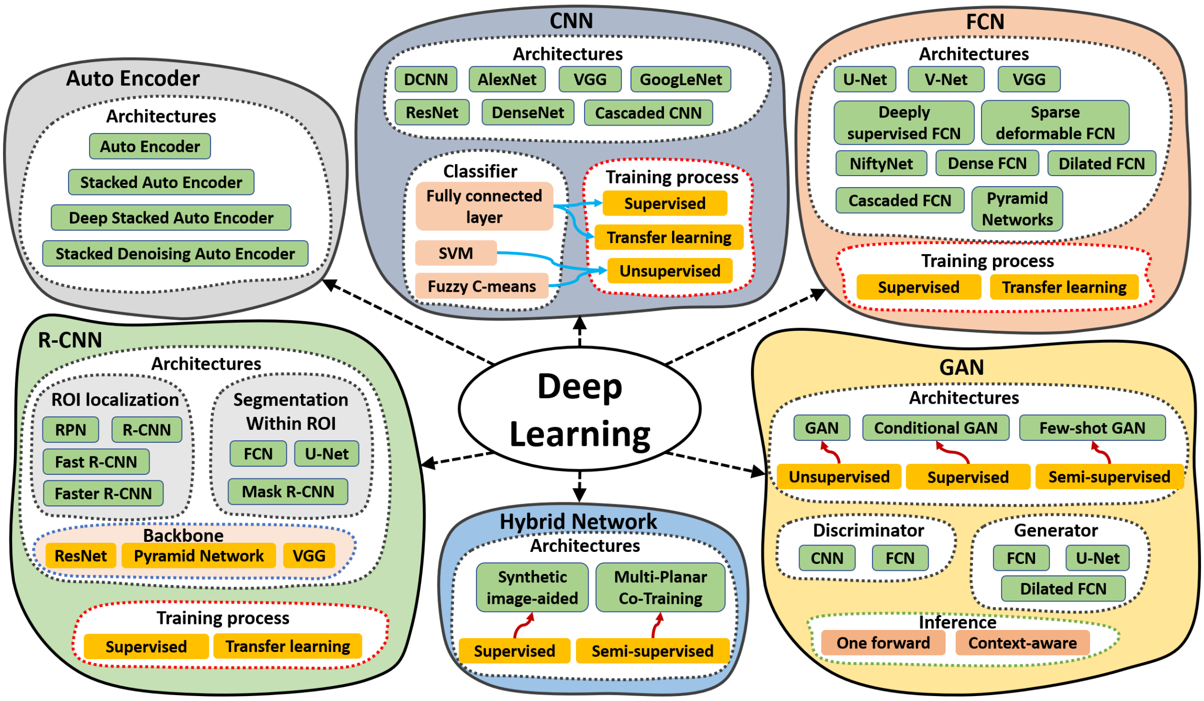


Figure 2.1: Semantic segmentation architectures overview

AE and its variants have been and continue to be extensively studied and used in medical image analysis. It is an architecture made up of 2 main parts: a neural network with an encoder layer that transforms the input into a latent representation minimizing the reconstruction error between input and output values, and a decoder layer that reports this latent representation of a few dimensions in the original input. In essence, AEs learn the representation (encoding) for a data set, learning to ignore insignificant points (noise). One of the most used and popular type of model is stacked AE (SAE) which consists of multiple AEs aggregated into multiple layers where the output of one layer is directly linked to the input of the next layer [28]. Denoising AE (DAE) is another variant in which we try to alleviate the problem of learning a trivial solution to the model by letting the network learn to rebuild a clean image starting from noise or other types of corruption [17]. In addition to the fact of not being able to have very deep networks due to the increase in computational complexity, the main problem of this kind of approaches appears to be due to the regularity of the data: irregular lesions with high internal variability are difficult to code for AEs.

CNN attribute their name to the type of hidden layer from which they are composed. The layers of this network in fact consist mainly of convolutional layers, max pooling, batch normalization, dropout, fully connected and normalization. The last layer, called a totally connected layer, typically depends on the type of task that the network has to solve: sigmoid or softmax for classification or tanh for regression. Due to the composition of this last layer in which all output neurons are connected to all input neurons, the totally connected layer tends to cause a loss of spatial information. This kind of architecture can't be used for segmentation, if you are working in a huge space of data (e.g. unconstrained real images). Fully connected layers can still do segmentation if you are restricted to a relatively smaller space (like a handful of object categories with limited visual variation), such that the FC activations may act as a sufficient statistic for those images [47].

Fully Convolutional Network with a last convolutional layer was adopted by Shelhamer et al. [33]. Thanks to the improvement of deconvolutional kernels used to up-sample the feature map, FCN allows you to output a dense voxel-wise prediction even from an entire volume. One of the most famous state of the art structures for segmentation is the U-net, proposed by Ronnenberg et al. 2015, in which the concept of deconvolution is used [39]. By using an encoding and a decoding path together with the design of the long skip connections between layers of equal resolution in the two parts of the network, the trade-off between organ localization and use of the context is expected to be overcome. V-net extends this concept by using residual blocks as short skip connections between early and later convolutional layers and replacing max-pooling operations with convolutional layers [25]. Possible cascade approaches were also studied in which two different U-nets, the first to identify the Region of Interest (ROI) to be given as input to the second where the real segmentation of the organs takes place [41].

The Generative adversarial networks have attracted a lot of attention in recent years in the field of medical image analysis thanks to the particular ability to generate data without explicitly modeling the probability density function. A typical GAN architecture consists of two competing networks, a generator and a discriminator. While the generator is trained to artificially generate data that approximates the target distribution, the discriminator is trained to distinguish artificial data from real data. The discriminator then encourages the generator to generate realistic data by penalizing unrealistic predictions by learning in a zero-sum game. At the end of the process, the discriminator is discarded as it is no longer able to recognize real data from artificial ones [30]. The main idea of this type of approach is to be able to have a satisfactory learning of the model using a very limited dataset, without sacrificing the performance of supervised training. In this regard, Xue et al. used a GAN (U-Net-GAN) in the task of multi-organ segmentation for thoracic CT images using a series of U-Net networks as generators and FCN networks as discriminators [22]. The peculiarities of GAN methods make it ideal to be used in the segmentation problem since patient movements such as translation and rotation do not change the relative position of the patients, even if it is difficult to balance the loss function between different organs of different sizes.

The R-CNNs follow an intuitive approach to the segmentation problem: 2 tasks are included in the training, one is the object detection which aims to classify and locate the objects of interest in the image using the bounding box called Region of Interest (ROI) and the second that carries out the actual segmentation on these found regions [19]. To this end, the R-CNN networks perform a selective search to extract approximately 2000 candidate regions from the image and by warping to a same size, these regional proposals were then fed into a CNN to extract a 4096-dimensional feature vector as output [27]. Due to the 2000 ROIs to be found for each 2D slice, these models need a lot of time and computational power to train. To solve this main problem, several architectures have been proposed: Fast R-CNN by Girshick et al. [14] which finds the ROIs by first passing the image into an FCN and subsequently generating Bounding boxes, and the subsequent evolution Faster R-CNN by Ren et al. [42] which, thanks to an algorithm of object detection, eliminates selective research by making the network learn also the region proposals.

Finally, the hybrid design usually involves two or more networks for different functional propose. An intuitive example could be a model where one network aims to enhance the image quality, and the other one to segment the OARs from the enhanced image. One of the approaches studied to treat low-contrast organs in CT scans without having MRI images available includes using a CycleGAN to estimate structural MRIs from CT images, and then training a second FCN network on segmentation based on the sMRIs produced [34]. Another interesting approach is that of the Deep Multi-Planar CoTraining (DMPCT) network based on the student-teacher architecture to increase the variance of the training data, in which pseudo-labels to unlabeled data are generated by a "teacher" architecture, which are used by the "student" network in conjunction with the manually labeled data to increase the variance of the training [50].

Problems and challenges with organ segmentation

The challenges of multi-organ semantic segmentation are very similar to the problems that are faced for generalized segmentation, with some emphasis on problems typical of medical analysis [48].

One of the main problems concerns the scarcity of data, which usually leads to an overfitting of the model on the training dataset with poor performance on new data. This is certainly due to the fact that a large number of training labels are not always available due to manual delineation by experts, often time-consuming and laborious. To date, many studies focus on training on multiple datasets or multimodal datasets, to increase the size of the dataset and take advantage of other types of information to increase performance.

Another big problem concerns class imbalance. For example, if the segmentation concerns large organs such as the lungs together with very small organs such as the trachea or esophagus, this will be biased towards the larger organ class. Working on the right loss function is the way to alleviate the problem.

The computational complexity is due to the fact of working on large medical images on network structures and learnable parameters that can be very high in numbers. Reducing the number of layers or parameters and focusing on methods that artificially increase the number of training data is one of the solutions.

In the real world, not all datasets are the same. Images taken by different machines, labeled by different experts or even by the same doctor at different times, lead to an error in the segmentation of the network to be biased towards that particular situation. These are possible scenarios that lead to errors called intra-observer and intra-observer, which however is a limitation that is expected to exist in all methods based on supervised learning.

Noise in medical images has always been a problem. If you add to this low contrast between the organs, this leads to a low quality image on which it is more difficult to achieve good performance.

2.1.4. Ensemble and fusion methods

Ensemble learning techniques combine different individual models in order to obtain better results than individual models. Deep ensemble learning models combine the advantages of both deep learning models and ensemble learning in order to have a final model with a better generalization performance.

The most famous ensemble strategies that have evolved over time are categorized as [26]:

- **Bagging**: Also known as bootstrap aggregating, is one of the standard techniques for generating the ensemble-based algorithms. The main idea of bagging is to reduce variance by averaging the output of multiple models built on different datasets. Specifically, each model is fitted using a dataset D_i built on uniform sampling and replaced by the total dataset D.
- **Boosting**: Boosting technique is used for converting multiple weak learning networks into a learning model with better generalization. The techniques such as

majority voting in case of classification problems or a linear combination of weak learners in the regression problems results in better prediction as compared to the single weak learner.

- **Stacking**: Ensembling can be done either by combining outputs of multiple base models in some fashion or using methods to choose the “best” base model. Stacking is one of the integration techniques where the meta-learning model is used to integrate the output of base models [46]. The Figure 2.2 quickly schematizes a basic architecture of ensemble stacking: the same input passes through a series of networks that create the single predictions. A meta-model, which can be a simple decision tree or a more sophisticated neural network, then takes these predictions and uses them to learn the optimal combination of all the networks considered.

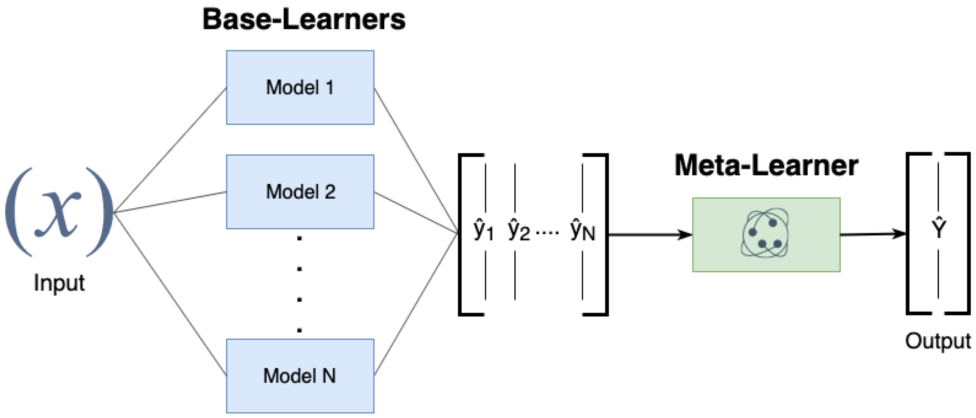


Figure 2.2: Base schema of ensemble stacking

The literature cite different ways of understanding the fusion of models based on the height of the data fusion point: input-level, layer-level and decision-level fusion strategy [49]. These types of approaches are commonly used in models involving training on multi-modal data (e.g. models trained on CT scan and MRI scan simultaneously) even on different datasets. This work relies heavily on the same blending methods applied to different classes as well.

As shown in Figure 2.3 , in the layer-level fusion strategy the images are used as the single input to train individual segmentation network. These learned individual feature representations are merged at some level (more or less depth) in the layers of the network, and finally the result are fed to the decision layer to obtain the final segmentation result. The layer-level fusion network can effectively integrate and fully leverage multiple different networks if features can be extracted from the model.

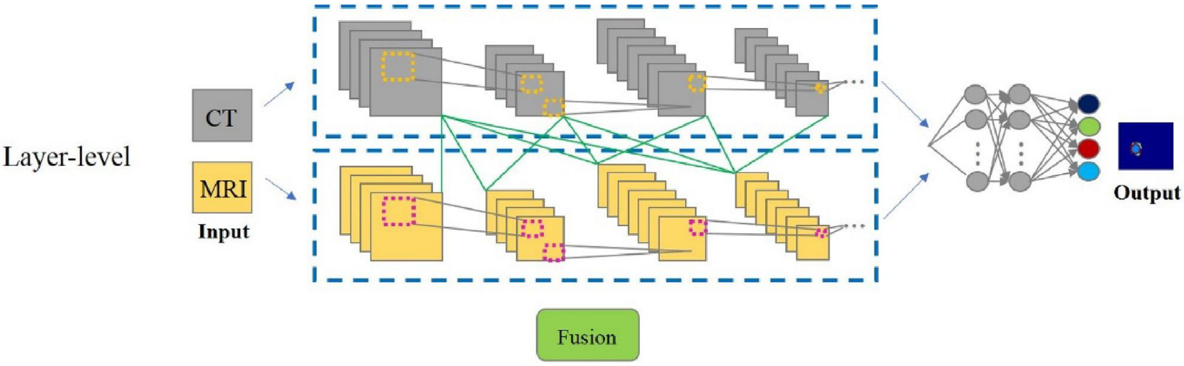


Figure 2.3: Layer-level fusion of 2 networks trained on different modalities

In decision-level fusion (also called "late fusion") segmentation network each modality image is used as the single input of single segmentation network. The outputs of the individual networks is then integrated to get the final segmentation result. Figure 2.4 describes the generic network architecture of layer-level fusion segmentation work.

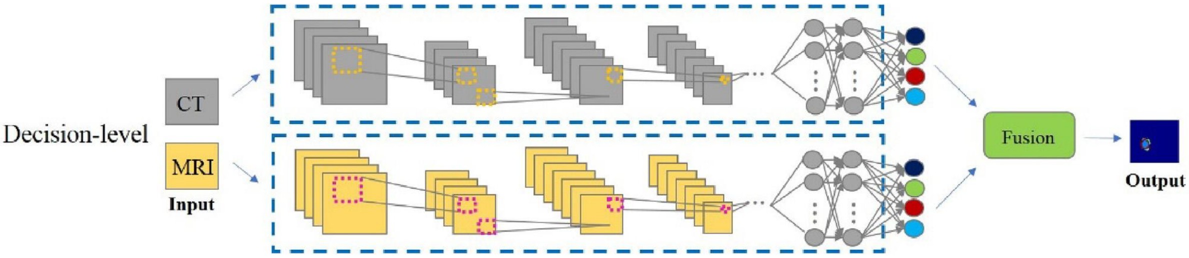


Figure 2.4: Decision-level fusion of 2 networks trained on different modalities

2.2. Theoretical background

The architectures we will use in our work are mostly based on convolutional networks. Some important notions about this type of models are briefly described in this section.

2.2.1. Artificial Neural Networks

Artificial Neural Network is an efficient computing system whose central theme is borrowed from the analogy of biological neural networks. As shown in Figure 2.5, ANN is a system made up of a large amount of units (neurons), linked together in a way that allows them to communicate mathematically.

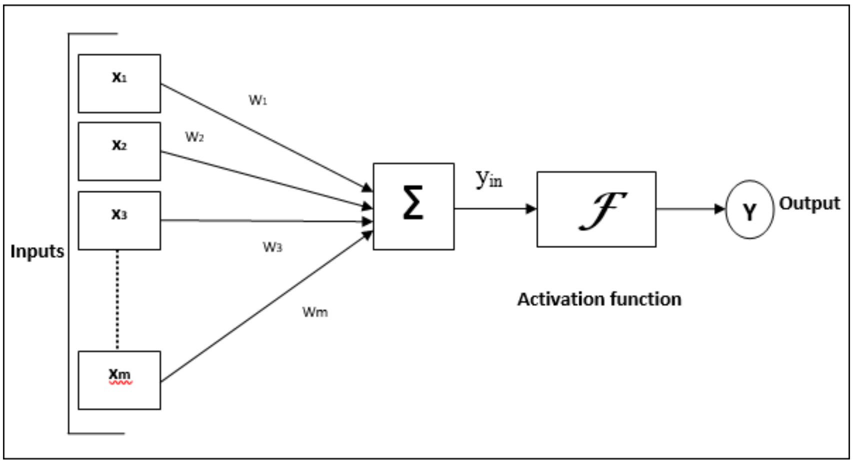


Figure 2.5: Very basic model of an Artificial Neural Network

Every neuron is connected with next layer neurons through a connection link. Each connection link is associated with a weight, the learnable parameter, that has information about the input signal. Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

2.2.2. Convolutional neural networks

The architecture of a convolutional NN is a multi-layered feed-forward neural network, made by stacking many hidden layers on top of each other in sequence. It is this sequential design that allows convolutional neural networks to learn hierarchical features. A simple network that helps visualize the core principles could be the early CNN LeNet-5 present in Figure 2.6, capable of recognizing handwritten characters.

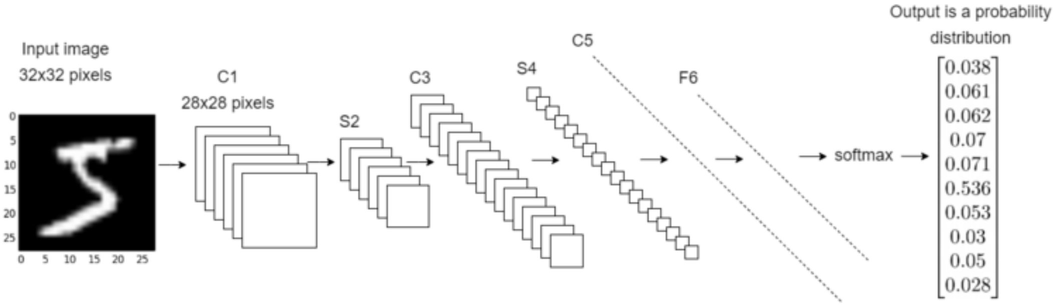


Figure 2.6: LeNet-5, an early and simple Convolutiona Neural Network capable of recognizing handwritten characters

The hidden layers are typically convolutional layers followed by activation layers that hold an activation function (tipically sigmoid or ReLU), some of them followed by pooling layers. While pooled layers simply downsample the image in a reduced dimensionality, convolutional layers perform a dot-product shifting of the whole image against filters. This is where the majority of computation occurs.

2.2.3. Fully Convolutional neural networks

The most obvious difference between FCN and CNN is their output. Ordinary convolutional neural networks can be used for classification to determine the class of some images and for object localization via regression while FCNs output the segmented images in their original input size. Like CNNs, FCNs can have different layers within them that operate on the image at different sizes and produce different feature maps. A quick overview of the application in medical segmentation is present in Figure 2.7 where the typical layers of a FCN are observed.

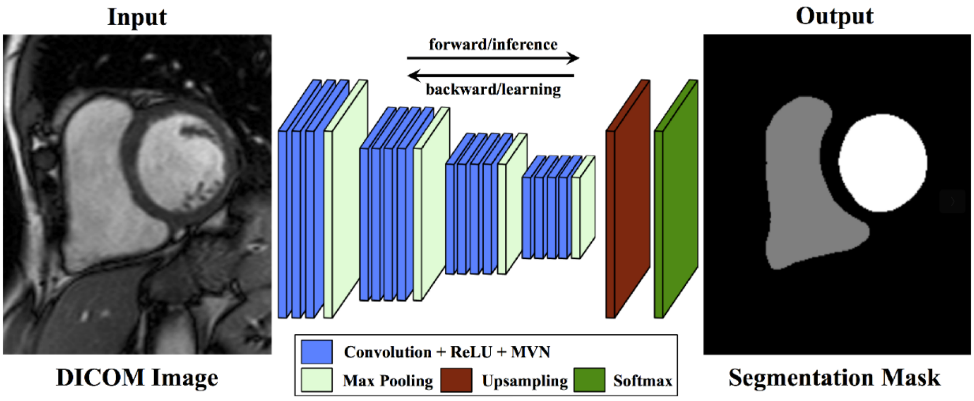


Figure 2.7: Structure and layers of basic FCN

1x1 Convolution

In this work we make a lot of reference to this particular type of convolution, used in most of the image-to-image segmentation models in the final layer. As the name suggests, the 1x1 convolution is a subset of the normal convolution with a filter of size 1x1, with zero-padding and a stride of 1.

Given a convolutional layer that outputs a shape tensor (B, K, H, W) where:

- **B** is the batch size
- **K** is the number of convolutional filters or kernels
- **H,W** are the input dimensions

and applying it to a 1x1 convolution with F filters, we will have a shape output (B, F, H, W). Figure 2.8 shows an example of this operation.

The 1x1 convolution can be used to address this issue by offering filter-wise pooling, acting as a projection layer that pools information across channels and enables dimensionality reduction by reducing the number of filters whilst retaining important, feature-related information [1].

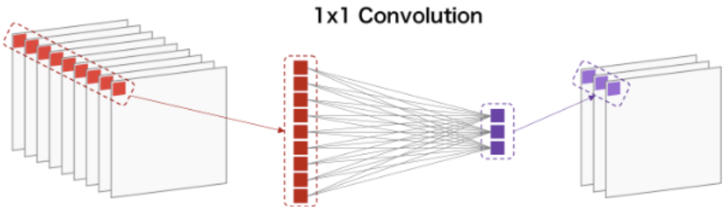


Figure 2.8: 1x1x3 Convolution. Each pixel (red) of the input is weighted and summed channel-wise.

2.3. Summary

After a brief introduction on the history of artificial intelligence in the medical field, citing some of the most important works, we moved on to talk about artificial neural networks and their evolution in the field of deep learning for semantic segmentation. We then gave an overview of the main works and studies in the medical field regarding the specific semantic segmentation of organs, also describing some important problems that are important to consider in this field. We moved on to describe the works in the field of ensemble stacking, on which this work relies heavily. Finally, we spent a few words to review the main theoretical concepts on which the networks used in this work are based. In the next chapter we will give an overview of the composition of the data we are going to work on, and how we have manipulated it to help the performance of our models.

3 | Datasets

In this chapter we first give some details about Computed Axial Tomography (Section 3.1) and what the Housenfield scale is. Then we describe the datasets used in this thesis (Section 3.2), with emphasis on the main features and differences of the data used in the work and the analysis made on the division of training and test sets. Finally, Section 3.3 is used to give a detailed description of the preprocessing steps applied in order to prepare images before being used in our models.

3.1. Computerized Axial Tomography

Since its introduction in 1972, X-ray computed tomography (CT) has evolved into an essential diagnostic imaging tool for a increasing variety of clinical applications. In order to scan a portion of the body, a narrown 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 (slices) the body [4]. The more dense a tissue, the more X-rays it absorbs.

Pixels in an image obtained by CT scanning are displayed in terms of relative radiodensity. In these scans the white regions indicate few x-rays absorbed, therefore denser tissues (such as bones), and the darker regions indicate that not as many x-rays will be absorbed, hence an area of air. The pixel itself is displayed according to the mean attenuation of the tissue(s) that it corresponds to on a scale from *+3.071* (most attenuating) to *-1.024* (least attenuating) on the Hounsfield scale.

Hounsfield units (HU) are a dimensionless unit universally used in computed tomography (CT) scanning to express CT numbers in a standardized and convenient form. Consequently, by ordering the types of tissue according to the HU value obtained from a scan, we obtain:

$$Air < Fat < Fluid < SoftTissue < Bone < Metal$$

Generally, CT images use 12-bit images able to store values between *-1.024* to *3.071*. How these values are displayed is then determined by a process called windowing, where the appearance of the picture is adjusted to highlight particular structures. The parameters to manipulate are the window level to adjust the brightness and window width to adjust the contrast. This is very important to underline as, in this work, we often make use of different CT window and level to improve the vision of specific organs. Table 3.1 lists an example of Hounsfield values for different organs and tissue types [18].

Substance	Housenfield Unit
Air	-1000
Lungs	-500
Fat	-100 to -50
Heart	20 to 50
Bones	45 to 3000
Water	0

Table 3.1: Housenfield values according to tissue type or substance

A good advantage of this approach is that though the use of ionizing radiations sometimes restricts its use owing to its adverse effects (even if the risk is generally small), CT can be used in patients with metallic implants or pacemakers where MRI is contraindicated [3].

3.2. Dataset Organization

Although more datasets were evaluated, our choice fell on two widely used datasets for the segmentation of thoracic organs at risk:

- The task 3 of **Structseg 2019** from the "Automatic Structure Segmentation for Radiotherapy Planning Challenge", part of the MICCAI 2019 [13].
- **SegTHOR 2019** from the "IEEE international Symposium on Biomedical Imaging 2019" (Challenge 4) [12].

Both datasets come with a training and a test set, although we will only use the annotated images from the first set for our work. In the following sections we give some details about the composition of these two sets, included some references to papers or other works where results about these sets are given.

3.2.1. Structseg

StructSeg 2019 is composed by fully anonymized 50 whole-volume chest CT images from lung cancer patients. Even if 10 additional patients are made available as test sets, they are not used in this work as they are not annotated. The dataset contains also the Gross Target Volume (GTV) annotations of the same patients (task 3) but, since we are interested only in the Organs segmentation (Task 4), we are not using them. Six OARs of each 3D chest CT image were annotated and checked by experienced oncologists and include Left lung, Right Lung, Heart, Spinal Cord, Trachea and Esophagus.

Images There are a total of 50 volumes in this dataset, with a number of slice that goes from 80 to 127 per volume and every axial slice has a voxel dimension of 512 × 512. A total of 3861 images are used. The images are grayscale so, instead of multiple color channels, there is just one channel that carries information and each value of a pixel represents only an amount of light. In Figure 3.1 is shown an example slice along with its annotation.

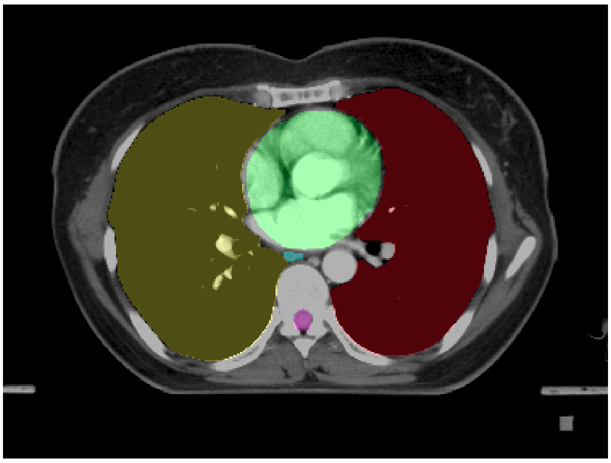


Figure 3.1: Structseg slice with: left lung (red), right lung (yellow), heart (green), spinal cord (pink) and trachea (light blue)

3.2.2. SegTHOR

SegTHor contains 60 CT scans from patients with lung cancer or Hodgkin's lymphoma. Like StructSeg, this dataset focus on 4 thoracic organs, which are heart, aorta, esophagus and trachea. These organs have varying shapes and appearances. The esophagus is the most difficult organ to contour due to its shape and position, which vary greatly from one patient to another and is almost invisible.

Images The training and testing data include 40 (annotated) and 20 (not annotated) patients, respectively. The CT scans have 512 x 512 pixels in-plane size in grayscale. The number of slices varies from 150 to 284 with a total number of 7390 images. We used only the published 40 CT scans as training and test data. In Figure 3.2 an example of a slice along with its annotation is shown.

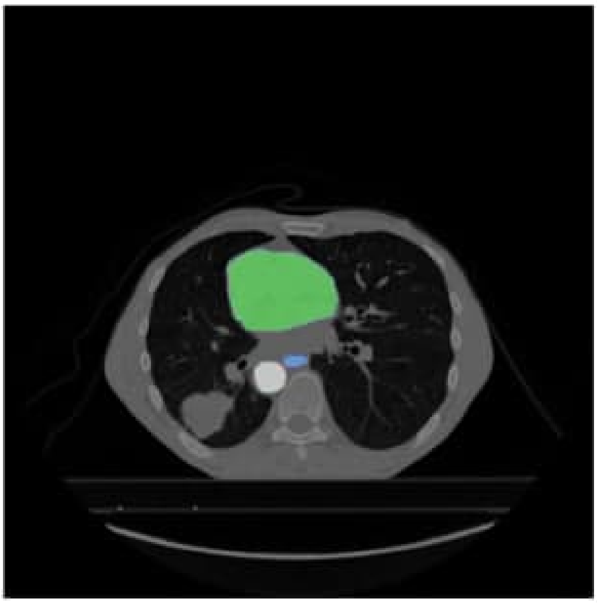


Figure 3.2: SegThor slice with annotations: heart (green), aorta (white) and trachea (blue)

3.2.3. Further analysis

The division into training datasets, used in the training process, and test sets was carried out trying to maintain the distributions between organs and HU values. Keeping the percentage of 20% widely used in the bibliography, we decided to divide the database based on the volumes for structseg and SegThor in 40/10 and 32/8 respectively. In Figure 3.3 we confirm how the distributions of the mean of the HU value per single pixel are kept as similar as possible between test and training set for both datasets.

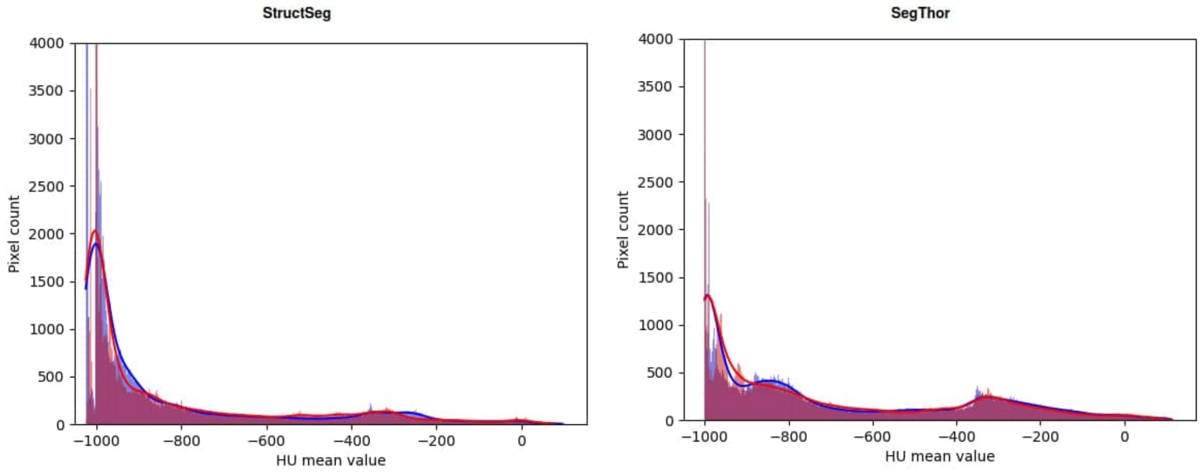
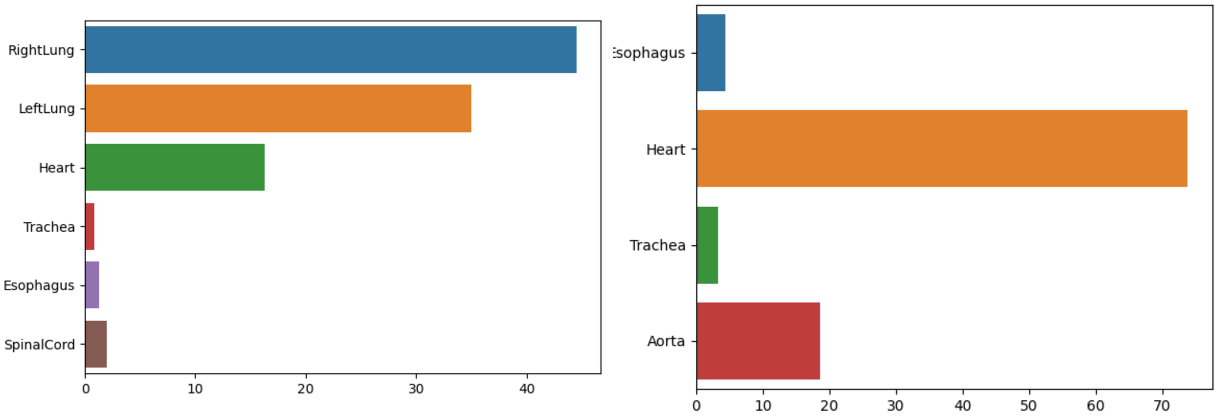


Figure 3.3: Structseg and SegThor average pixel HU distribution with test set in blue and training set in red

We also underline the fact that the distribution between background and organs is very unbalanced in both datasets, since it is the preponderant part of most of the images. Removing the background, even among the same organs we can see, Figure 3.4 a strong class imbalance in favor of the larger organs. This applies to both structseg and segthor.



(a) StructSeg

(b) SegTHOR

Figure 3.4: Organ percentages

3.3. Preprocessing

In any work on deep learning there is almost always some sort of preprocessing on the datasets required to clean or transform the data in a way that suits the experiments. Below are the transformations used for all datasets, although as explained in Section 4.1.1 online augmentations techniques are used to further improve performance.

3.3.1. Hounsfield scale window

As mentioned in chapter 3.1, in order to highlight a specific tissue/organ a window on the Hounsfield values can be set to look at certain area of interest.

To choose a window, a 'level' and a 'width' is defined to adjust respectively the brightness and the contrast of the sample so that, for example, a window with a level of *0* HU and a width of *400* HU will have a range of -*200* HU to *+200* HU. Any tissue with a density of -*200* HU or less will be black, and any tissue with a density of *+200* HU or more will be white so that values between -*200* HU to *+200* HU will be spread between the whole grayscale range.

A small range of tissue density is represented by a full greyscale spectrum from black to white, thus making subtle density differences within the specified range easier to see. Although this varies somewhat from institution to institution and vendor to vendor, window width and centers are generally fairly similar. In the Table 3.2 are shown the typical width and level values in HU, that largely coincide with the ones we used. For multi-class networks we use the same values as for the lungs, which is the maximum useful interval [19].

Tissue	**Width**	**Level**
Lungs	1500	-600
Heart	350	50
Bones	1800	400
Esophagus	300	80
Trachea	1200	-440
Brain	80	40
Spinal Cord	600	0

Table 3.2: Hounsfield window according to tissue type

In Figure 3.5 we observe the difference of the used window between various organs in our datasets.

For the implementation we apply this calculation widely used in the literature to find the minimum and maximum of the range of values, given Window Level and Window Width:

$$min = \frac{2WinLevel - WinWidht}{2} + 0.5$$

$$max = \frac{2WinLevel + WinWidht}{2} + 0.5$$

To find the multiplication factor for the pixels in (0, 255) range:

$$d = \frac{255}{max - min}$$

And to elaborate the final image:

$$finalImage = (rawImage - min) * d$$

Then all pixels greater than 255 are set to the value of 255 and for the ones less than 0 set to 0 [11].

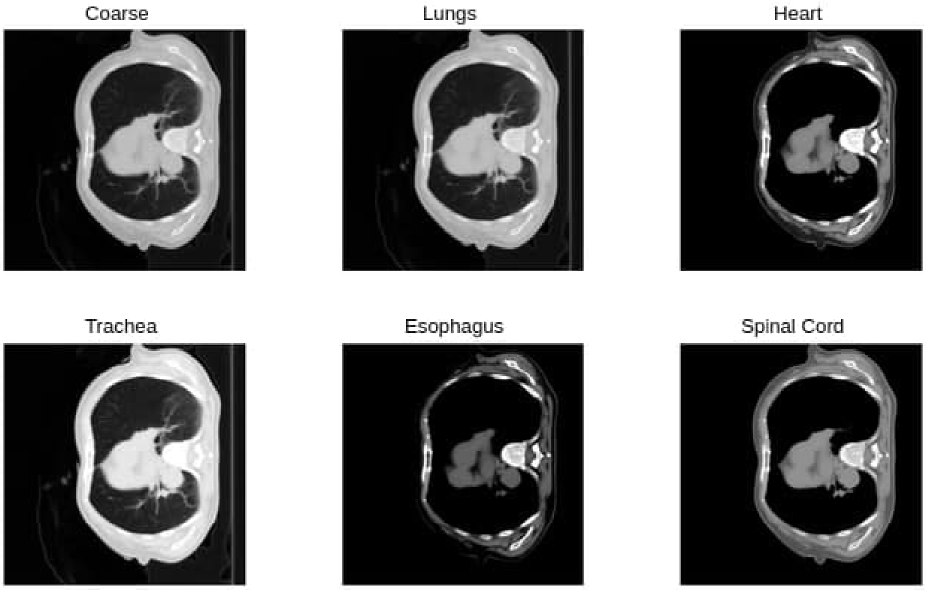


Figure 3.5: An example of all 6 organs in StructSeg highlighting different areas according to HU window

3.3.2. Normalization

A common point for all types of Neural Networks is the need for normalization i.e. to make our range of values in a meaningful range for the task. The main reason to use this transformation on images before feeding them to our network comes from the fact that in many machine learning algorithms it is required to have the same dynamic range in input, which means to have the same difference between the minimum pixel value and the maximum for each image. It is therefore possible to avoid unwanted effects by converting the values of the images into a different interval using in our case a type of rescaling called min-max scaling or min-max normalization where the transformation concerns a single color channel having to do with images in grayscale . Min-max is one of the simplest forms of normalization that allows you to bring the difference into an interval [a, b], using the formula:

$$x^{'} = a + \frac{(x - min(x))(b - a)}{max(x) - min(x)} \tag{3.1}$$

In our case we wanted to bring all the images to lie within the range [0, 1] by using the Formula 3.1 that with a = 0 and b = 1 becomes:

$$x^{'} = \frac{x - min(x)}{max(x) - min(x)} \tag{3.2}$$

where, given a volume of a subject, x is the original intensity value and x' is the new normalized voxel value. Considering that we already used the windowing transformation (Section 3.3.1), min and max values were set to 0 and 255 so that the final transformed image is simple a division for 255.

3.3.3. Cropping

As shown in Section 3.2 of this chapter, most of the pixels present are black i.e. empty pixels. To alleviate this problem, all images in the dataset were first center cropped: the outermost part of each slice was removed. In our case, we found it optimal to use a crop area of [320, 320] in the CT scans of both datasets, so that no useful information is removed. This has brought considerable advantages in terms of memory occupied, speed in training and reading from disk without compromising the performance of the models.

Figures 3.6 and 3.7 shows an example of 5 cropped slices against normal ones for both structseg and segthor datasets.

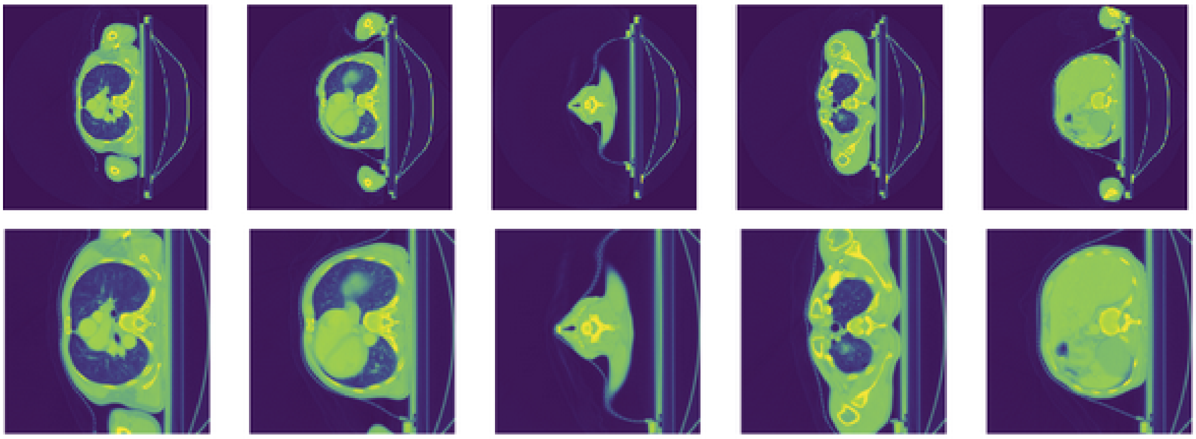


Figure 3.6: StructSeg example of cropping: the first row for the normal images, the second for 5 random slices of a volume after cropping it to a width of 320 and height of 320

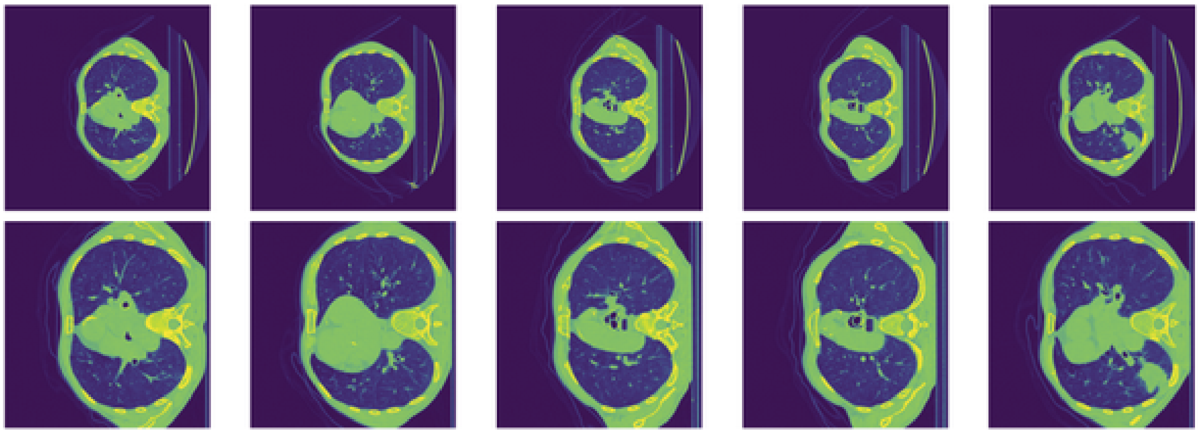
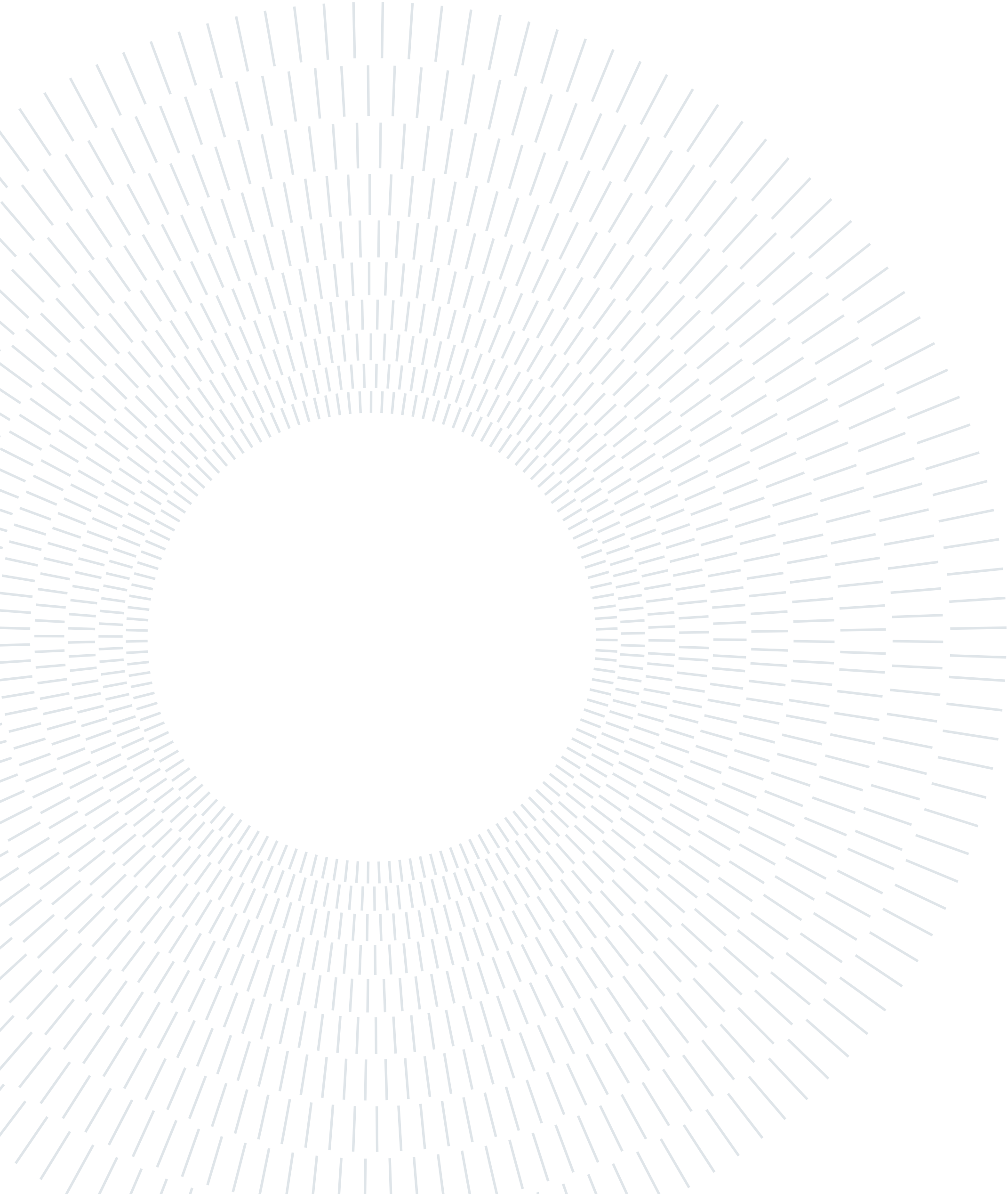


Figure 3.7: Segthor example of cropping: the first row for the normal images, the second for 5 random slices of a volume after cropping it to a width of 320 and height of 320

3.4. Summary

In this chapter we have given a smattering of Computerized Axial Tomography, in order to understand the types of data we need to use i.e. Housenfield value. We have also given a detailed description about the datasets we use in this work, showing how it is organized and what kind of images it is composed of, with special regard to the type of organs present. We also focused on the fair division of the datasets into train and test, so that this affects the possible conclusions as little as possible.

Finally, we thoroughly described the types of preprocessing used before using the images for actual training on the models described in the next chapter.



4 | System design and overview

This chapter aims to give a detailed description of how the data is processed before being passed to the networks for the training phase (Section 4.1.1). Section 4.2.1 describes in detail the architectures used in the basic models of segmentation, used by the ensemble architectures present in Section 4.2.2. Section 4.3 presents the technical information on the implemented algorithms and the loss functions used in the training phase. Finally, in the Section 4.4 the evaluation metrics used in our work are shown in order to be able to make comparisons between the same models of this work as of models present in the literature.

4.1. Input pipeline

The datasets are composed of files in DICOM format (Digital Imaging and COmmunications in Medicine), one of the most used standards for this type of application. There are 2 files for each patient: one representing the volume made up of CT scan slices and the second the labels associated with this volume. Before starting with the preprocessing phase, the DICOM files are transformed into HDF5 format, suitable for containing and organizing large amounts of data. In fact, using this type of file greatly simplifies the organization of the hierarchy, uploading files and it is possible to use it directly with well-known libraries in data analysis (e.g. Numpy, Pandas). The single HDF5 file is then saved on disk, ready to be loaded into memory during the training phase.

We have implemented different types of data loaders, depending on the transformations or the type of ensemble network in which it is to be used. In all loaders, however, there is always a first preprocessing phase which consists of the transformations discussed in the 3.3 section: in order, HU windowing, data normalization, and slice crop. The training set is shuffled since Gradient Descent works best when the instances are independent and identically distributed. Batching is not relevant in this work as only batches of size 1 are used for physical memory reasons.

All training images are then passed through random transformations in order to increase performance on test sets.

4.1.1. Augmentations

Modern machine learning models often have a very high number of parameters that allow them to generalize well when trained on a large number of labeled data. In particular, especially in the medical field, this large labeled dataset is not always available for training, which leads to a high risk of overfitting. Data Augmentation is a widely used technique to increase both the size and diversity of the tagged dataset using various transformations on the input.

Since in this work we use a fairly limited dataset, we use data augmentation by applying elastic deformations to the available training images. This allows the network to learn a kind of invariance to these transformations, without the need to add these transformations to the real dataset. This is particularly important in the segmentation of biomedical images as the deformations that can be simulated are the most common variations of organs and tissues.

The approach used in this work regarding augmentations is online: every time an image is loaded, it is passed through the transformation pipeline. These transformations have a 50% chance of actually being used with range-defined variable parameters. In practice we use Pytorch Albumentation library which provides many basic transformations and tools for this purpose [23].

Elastic Transform

When it comes to working on highly variable images i.e CT body scan, shape differences between different people are expected. For that, in medical imaging problems non-rigid transformations can help to augment the data. We use this elastic transformation to vary the images and the masks in order to increase the variability of the shapes of the various organs with parameters that do not allow to generate unsuitable or excessively distorted shapes. In Figure 4.1 is shown an example of elastic transform.

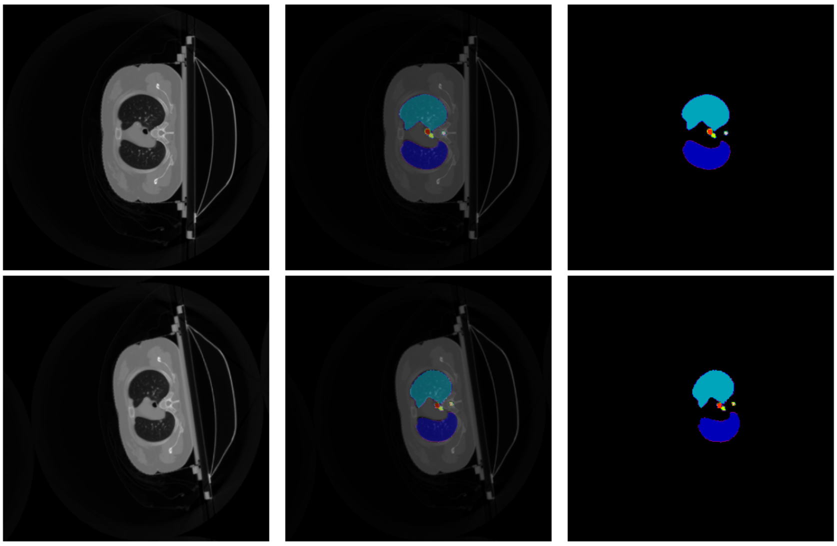


Figure 4.1: Example of elastic transformation (below) with parameters $\alpha = 30$, $\sigma = 48$, $\alpha_{affine} = 48$

Grid Distortion

This transformation allows you to create rigid grids on the image and change the measurements of the cells to distort the dimensions. As usual, parameter values are used in order to not compromise too much the shapes of the organs. In Fig. 4.2 is shown an example of elastic trasform.

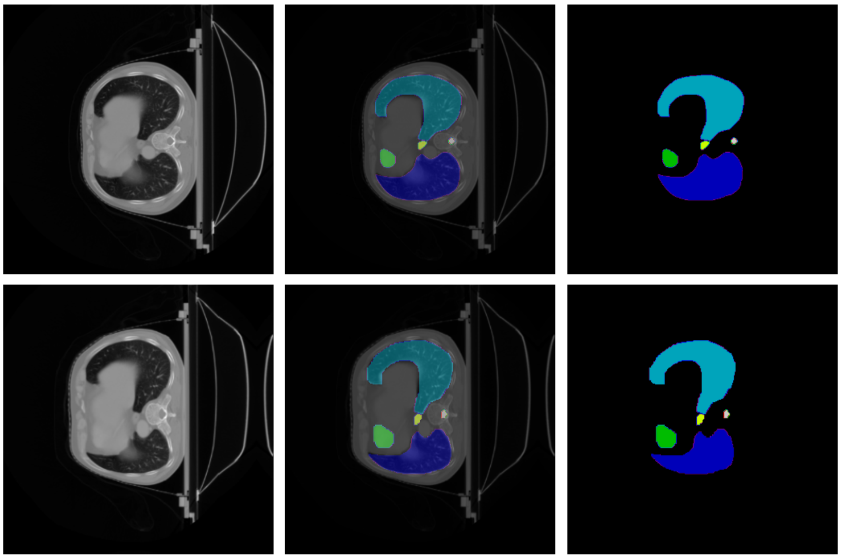


Figure 4.2: Example of grid distortion (below) with parameters $n_{steps} = 5$ and $DistortionLimit = 0.3$

Rotation

Widely used in any field of segmentation, rotation allows you to obtain copies of the same image rotated at different angles. In our case we have chosen a rotation in a range of 10°, so as not to alter the position of the organs in the image too much, but enough to simulate a possible rotation due to the patient's movement.

4.2. Architectures

The base of our architecture are the segmentation FCNs. These are used in combination with each other in the ensemble to refine the final segmentation masks. In this section we first describe the convolutional network architectures used, then move on to the ensemble models showing the differences from the literature.

4.2.1. Segmentation networks

In this work we used three Fully Convolutional Neural Networks: the classic Unet [39], a Squeeze-and-Excitation block version of Unet called SE-ResUnet [19], And the network developed by google DeepLabV3 which introduces the Atrous Spatial Pyramid Pooling mechanism [20].

To build our ensemble methods we use binary segmentation models well known in the literature. The logic has not been changed in the implementation compared to the standard used described in the relative papers mentioned above.

Unet

The Unet network is a Fully Convolutional Network based on an U-shape Encoder-Decoder structure, from which it takes its name.

The main idea of this approach is to downsample the input progressively to the bottleneck layer, then do the reverse process and pass the information through multiple upsample layers. Links, called "skip connections", are used between pairs of contraction/expansion layers to solve the problem of the loss of spatial information of the bottleneck layer. This preserves the structural integrity of the image, which greatly reduces distortion of the image.

As shown in Figure 4.3, this architecture consists of three sections: the contraction, the bottleneck, and the expansion section.

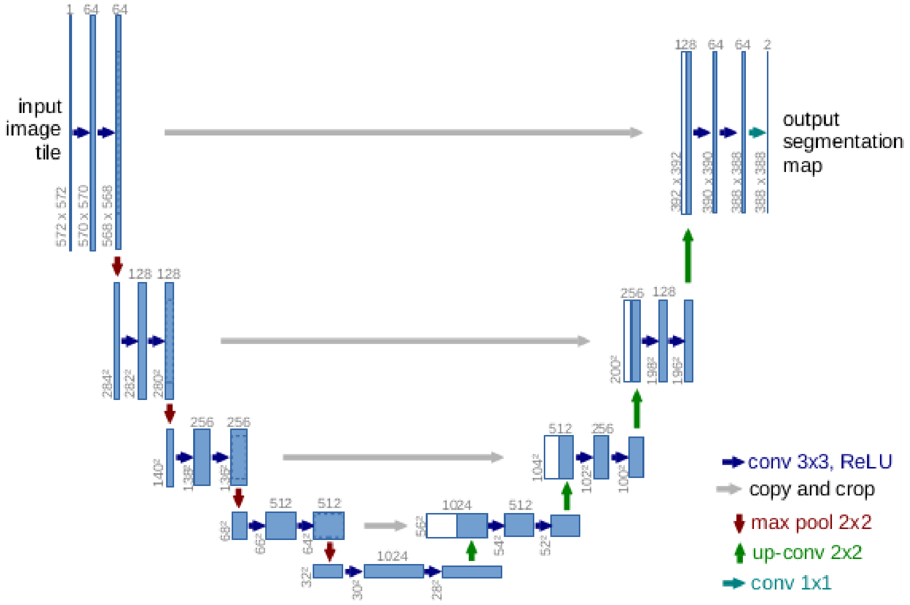


Figure 4.3: Example of Unet architecture with 32x32 input size.

The contracting path follows the typical architecture of a convolutional network. It consists in the repetition of blocks formed by 3x3 convolutions (unpadded), each followed by a batch normalization operation and a rectified linear unit (ReLU), ending with a max 2x2 pooling with stride 2 for each downsampling. With each downsampling we double the number of features in the channels. The purpose of this contracting path is to capture the context of the input image in order to be able to do segmentation.

The bottleneck is the conjunction between encoder and decode, and consists of a vector of small dimensions with a high number of feature maps.

The expansion part consists of a similar structure, coming to be almost symmetrical with the other part: repetition of blocks of 3x3 convolutions (unpadded), each followed by Batch normalization and ReLU. The upsampling part, in which the number of feature channels is halved, is carried out by a 2x2 Transposed Convolution (with stride 2). The main difference with the other way is in the fact that after each upsampling a cut part of the feature map is concatenated to the result by the contracting path through the skip connections. A final layer with a 1x1 convolution is used to map each of the 64 feature vector components into the desired number of classes. A standard Unet has 23 convolutional layers, and a total of 17M learnable parameters [8].

SE-ResUnet

SE-ResUnet [19] is a variant of a Unet that employs squeeze-and-excitation blocks to enable the network to perform dynamic channel-wise feature recalibration [29]. Figure 4.4 shows how the structure is similar to Unet, with the same contracting/expansion and up/down block paradigm.

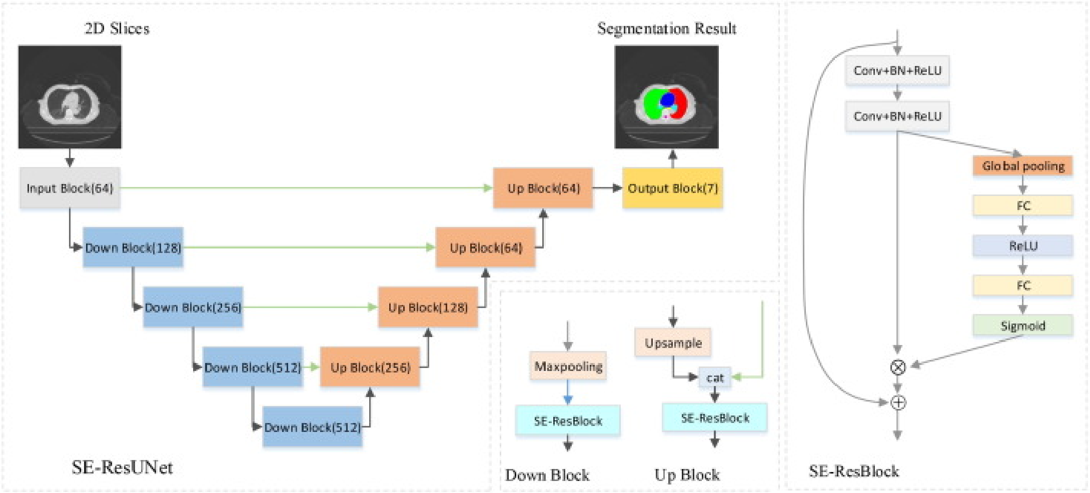


Figure 4.4: Example of SE-ResUnet architecture, with detailed upsample and downsample blocks [19] and basic SE-ResBlock.

The SE-ResBlock takes a convolutional block as input. Each channel is squeezed into a single value using average pooling, after which a dense layer followed by a ReLU adds nonlinearity and the output channel is reduced by a certain ratio. Another dense layer followed by a sigmoid gives each channel a smooth gating function, and finally each convolution feature map is weighted based on the side network in the "excitement" phase. In Figure 4.5 we see a simplified version of the block where the excitement of the output is emphasized.

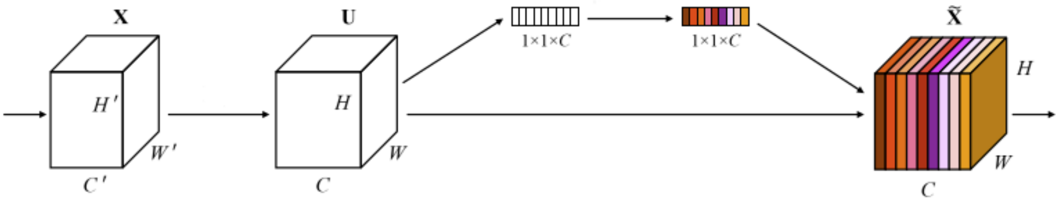


Figure 4.5: SE block diagram: H and W represent the size of the different inputs Xs, C the number of channels used. The different colors underline the influence of the individual parts of the vector on the convolution block.

DeeplabV3

DeepLab is a state of the art semantic segmentation model conceived and designed by Google in 2016 [20]. The network is formed by the usual 2 phases already seen in section 4.2.1 i.e. an enconding (contraction) phase essential to extract information from the image, and a deconding (expansion) phase used to reconstruct an output of the appropriate size.

This architecture combines some powerful concepts known in the field of computer vision, such as Spatial Pyramic Pooling (SPP) [7] and Atrous Convolution, in a structure called Atrous Spatial Pyramic Pooling (ASPP).

Spatial Pyramic Pooling keeps spatial information in local spatial bins with a fixed number and size. In each container each feature is then pooled and subsequently aggregated together with the others. Figure 4.6 shows an example of this structure in a part of a convolutional network: the feature maps are passed through several bins in parallel with a fixed pooling size, to then be combined and passed to the next fully-connected layer.

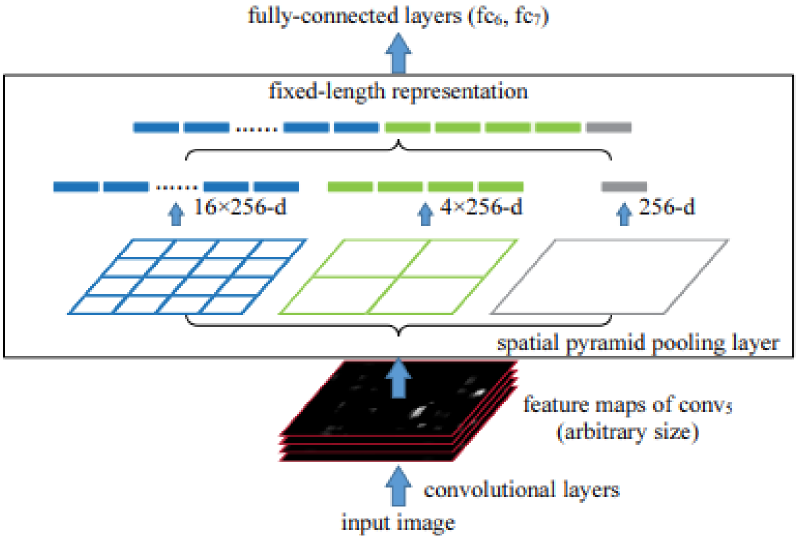

fully-connected layers (fc6, fc7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of conv5
(arbitrary size)
convolutional layers
input image

Figure 4.6: Spatial Pyramid Pooling with 3 bins of size (1,4,16) and 256 feature maps filters

SPP uses multiple instances of the same architecture, which leads to an increase in the computational complexity and memory that the network requires for training. To address this problem, DeepLabV3 introduces the concept of atrous convolutions (Figure 4.7), a generalized form of convolution operation. The atrous convolution uses an additional parameter with respect to the normal convolution, called rate, which is used to control the field of view of the convolution and allows to capture long-range information. The general form of the convolution is given by:

$$y[i] = \sum_k x[i + r * k]w[k]$$

where r is the rate and k the size of the convolution. Note that when $r = 1$ it is just a standard convolution.

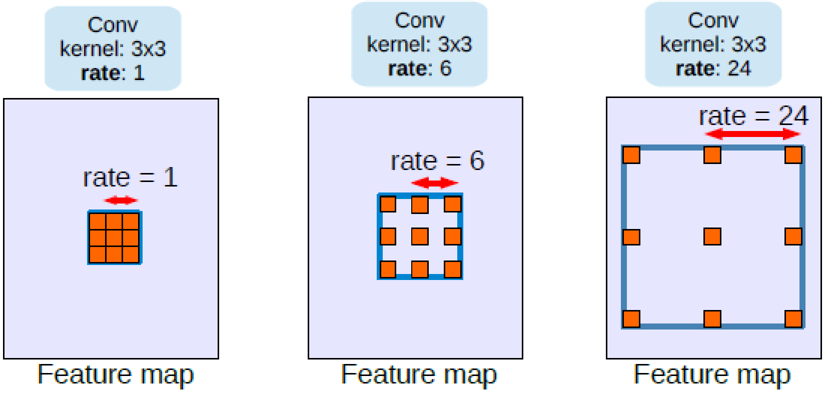


Figure 4.7: Atrous Convolution with different rates r

DeepLab uses a combination of these two methods in particular in the last block of the network i.e. when the output stride reaches size 16: the ASPP structure uses one 1x1 convolution and three atrous convolutions with rates (6, 12, 18), for all 256 filters and batch normalization.

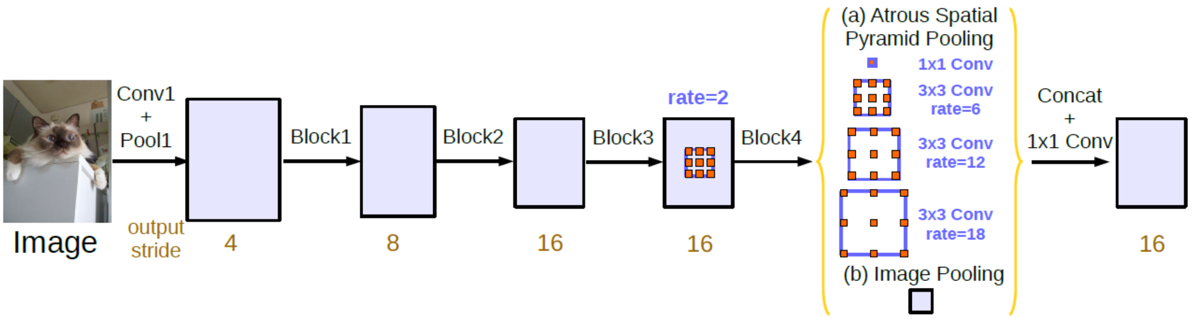


Figure 4.8: Parallel modules with atrous convolution (ASPP).

4.2.2. Ensemble networks and methods

Multiclass segmentation has recently achieved significant performances in images and video. This result is mainly attributable to the availability of large public multi-class datasets. However, there are fields, such as biomedical images, where obtaining a number of annotated multi-class data is laborious and often impossible, and only single-class datasets are available (perhaps even from the same source). One of the problems of having single networks trained on single-class datasets is to fail to embed the spatial correlations among organs and thus are likely to be overfitted for each particular single-class dataset.

Most of the methods that single-class dataset uses are specialized in the segmentation of a particular organ. Most of the ensemble methods in the literature use different networks with different architectures to obtain a greater generalization of the prediction but using the same multiclass dataset [24]. There are studies on the classification of images in which we can see approaches closer to this work: starting from models trained on datasets with different targets, the results are merged [35].

The approach we use in this work can be seen used in literature for multi-modal segmentation problems [49], in which semantic segmentation starting from multiple modal inputs e.g. CT scan and MRI are used.

Assuming that a single well-trained binary network can better exploit the information of the corresponding organ, we have come to think of several ensemble models that include the use of binary networks to have a more accurate final multi-organ segmentation.

Argmax of binary outputs

First approach of decision-fusion network. As the name suggests, for this model we intuitively use a single network for each of the n target in order to have n output logit masks at the end. Each individual mask is then passed into a sigmoid function and then combined together using the maximum value among all channels as the final result, but keeping only those above a certain threshold. The operation is therefore an argmax of C classes for each *(x, y)* pixel on the set of masks M:

$$\arg\max_{C} M_c(x, y)$$

The result is a final multi-class segmentation mask complete with all desired targets. We use this simple combination of results as a baseline to evaluate the performance of more advanced ensemble models as this is the most basic way of combining results from different networks. Since the combination takes place in the "decision" phase i.e. with the output of the individual networks, it is possible to use a heterogeneous variety of models and datasets. We emphasize the possibility of using different preprocessing depending on the target, as shown in Figure 4.9.

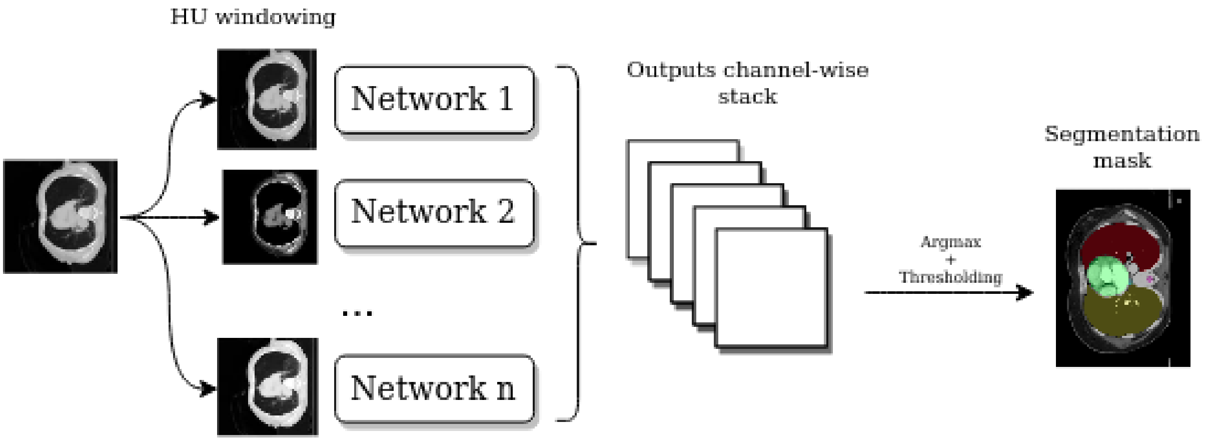


Figure 4.9: ArgMax combination scheme of different networks the same image but different preprocessing as input (HU windowing in this case as in Section 3.3.1)

Obviously this type of network does not need to be trained if you already have well-trained models available on the target.

1x1 convolution ensemble stacking

As shown in Section 2.2, the 1x1 convolution is a tool for reducing the size of filters while retaining feature-related information. Therefore, using this tool on top of the stacked binary networks output, it is possible to weigh each mask according to the target. The weighted masks are then combined pixel-wise and thresholded. As in the case of ArgMax, it is possible to use this type of ensemble with heterogeneous segmentation models.

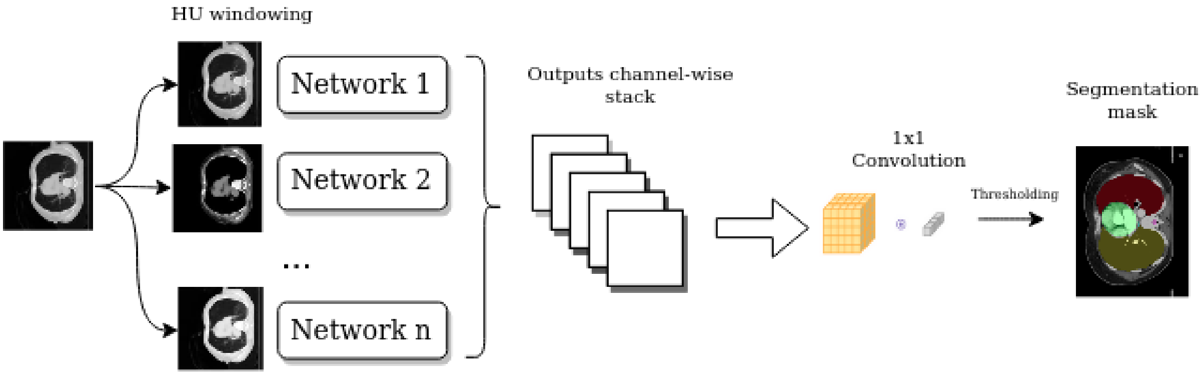
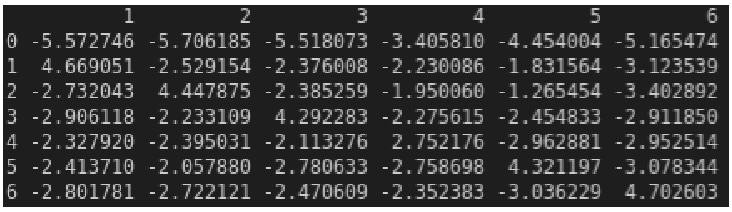


Figure 4.10: 1x1 convolutional ensemble stacking

This type of structure has the advantage of being more meaningful than the others: for each target it is possible to check the weights associated with each input mask and understand from which network is more dependant. An example is shown in Figure 4.11, where the complete weights of a 1x1 convolution on 6 classes with 6 input networks are shown: as expected, for each target, the network with the most "weight" is the one trained on that class.



```
          1         2         3         4         5         6
0 -5.572746 -5.706185 -5.518073 -3.405810 -4.454004 -5.165474
1  4.669051 -2.529154 -2.376008 -2.230086 -1.831564 -3.123539
2 -2.732043  4.447875 -2.385259 -1.950060 -1.265454 -3.402892
3 -2.906118 -2.233109  4.292283 -2.275615 -2.454833 -2.911850
4 -2.327920 -2.395031 -2.113276  2.752176 -2.962881 -2.952514
5 -2.413710 -2.057880 -2.780633 -2.758698  4.321197 -3.078344
6 -2.801781 -2.722121 -2.470609 -2.352383 -3.036229  4.702603
```

Figure 4.11: 1x1 convolution weights example. The rows are the target of the convolution (0 = background) and the columns are the outputs of the binary networks trained on that class. In this case the heaviest weights are on the diagonal, which represents the networks trained on the same target organs.

Unet ensemble stacking

Stacking (also called meta-ensembling) is a model ensembling technique used to combine information from multiple predictive models to generate a new model. In Section 2.2 we talked about how an approach used in ensemble models involves the use of a meta-model [49] that combines the outputs of single binary architectures. In this work we use an FCN network, specifically a Unet, to obtain a multiclass segmentation starting from the logit masks of the single models used, as shown in Figure 4.12.

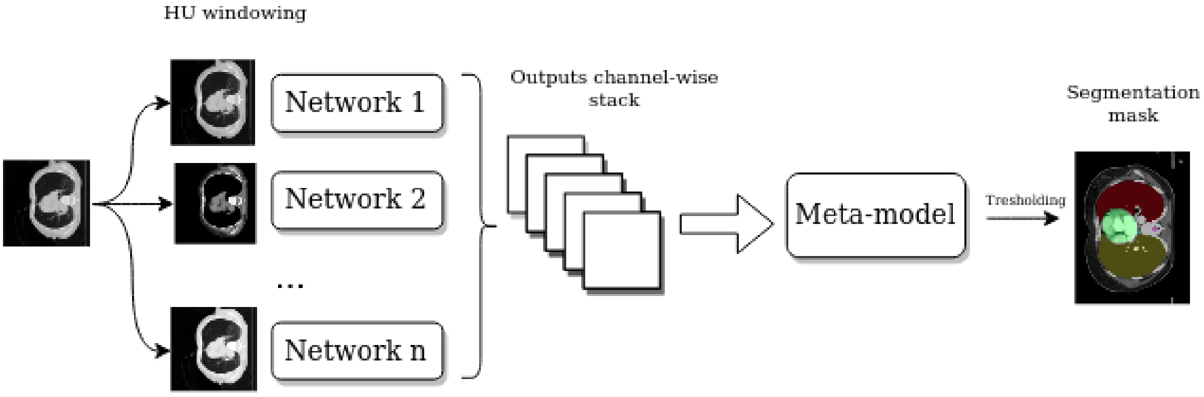


Figure 4.12: Binary ensemble meta-model schema

Last-layer Feature fusion

This method is part of layer-level fusion strategies, in which individual well-trained networks are combined through the concatenation of the learned individual feature representations of a specific layer. Like most of the ideas in this work, inspiration was drawn from multimodal multi-class segmentation structures and studies [49].

A fundamental difference with multimodal examples, or with most of the literature concerning the combination of models, is that we try to solve a multi-class problem starting from models trained on single classes.

Depending on the network, the features are taken from what we have identified as the last layer before the decision and fed to a 1x1 convolution for dimensional reduction. This leads to having a number of learnable parameters equal to the number of concatenated features * the number of classes in output (+ biases). In Figure 4.13 we see a schematic of the architecture.

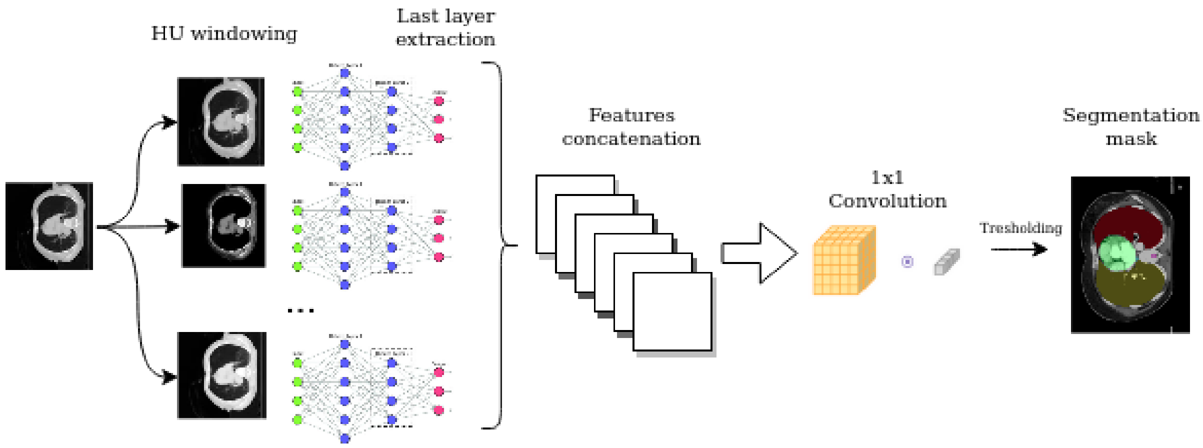


Figure 4.13: Last layer Feature Fusion

The identified fusion level is the last layer of a generic FCN, which changes according to the networks considered. In our case, Unet and SE-ResUnet share the same point, having very similar architectures, which is located just before the final 1x1 convolution (reference in Figure 4.3). For DeepLabV3 instead, it was necessary to take the features before the 1x1 convolution and the upsampling operation that occurs at the end of the network. For this reason, we take the identified features of DeepLab and use the same Upsample operation before concatenating them to the other features.

Finally, due to the different structure of the architectures, the number of features that are extracted from Unet and SE-ResUnet networks amounts to 64 for both, and 256 for

DeepLab.

4.3. Training

This section presents the detailed training process, together with the hyperparameters and Loss functions used. The evaluation metrics fundamental to understanding the performances of the studied models are also described in detail. We also give a technical description of the process implementation.

While using the same algorithm, the training of binary and ensemble networks changes in some fundamental aspects. In ensemble models the selected pre-trained binary convolutional networks must first be loaded. Depending on the target classes, the output is preprocessed and passed to the single network.

As optimizer is used the Adaptive Moment Estimation, also called Adam.

4.3.1. Losses

The loss function, which aim to measure the dissimilarity between the ground truth and the predicted segmentation, is a critical component in deep learning-based segmentation methods. Especially for medical image problems where there is often a strong class imbalance, it is necessary to choose a function that does not penalize too much the tissues or the less present targets.

Given the diversity of problems, the chosen approach consists of using a weighted CrossEntropy for multi-class training as seen in [19], while for the binary problem a combination of Dice and CrossEntropy for greater robustness [36].

CrossEntropy Loss

Cross entropy (CE) is derived from Kullback-Leibler (KL) divergence, a measure of dissimilarity between two distributions.

For a generic CNN segmentation task, CrossEntropy is defined as:

$$L_{CE} = -\frac{1}{N} \sum_{c=1}^{C} \sum_{i=1}^{N} w_c g_i^c \log s_i^c$$

where g_i^c i is the ground truth binary indicator of class label c of voxel i, and s_i^c is the corresponding predicted segmentation probability. w_c is a parameter of the Weighted-CE that serves to weigh each class C in order to penalize some targets. It is usually inversely proportional to the frequency of the classes, penalizing the most frequent classes in the dataset.

In our implementation the CrossEnstropy is combined with the SoftMax function, generally used in multi-class problems to normalize the scores for a given class. This type of loss is also called Softmax Loss, and is equivalent to:

$$L_S = -w_y \log \frac{e^{x_y}}{\sum_{c=1}^{C} e^{x_c}}$$

where x is the input, y is the target, w is the weight, C is the number of classes

Dice Loss

Dice loss can directly optimize the Dice Similarity Coefficient (DSC) which is the most commonly used segmentation evaluation metric. Unlike weighted cross entropy, Dice loss does not require class re-weighting for imbalanced segmentation tasks.

$$L_{Dice} = 1 - \frac{2\sum_{c=1}^{C}\sum_{i=1}^{N} g_i^c s_i^c}{\sum_{c=1}^{C}\sum_{i=1}^{N} g_i^c + \sum_{c=1}^{C}\sum_{i=1}^{N} s_i^c}$$

In standard Dice Loss, s_i^c should be in binary form [0,1]. Due to the nature of the backpropagation process in training, a version of Dice Loss called Soft-Dice Loss is used which uses the probability s_i^c with no threshold.

Composite Loss

As suggested in literature [36], for a more robust training it is recommended to use a combined Loss. These losses use two or more known loss functions to take advantage of both features.

In this work we use the weighted combination of Dice and CE loss, defined as:

$$L_{DiceCE} = w_{CE} L_{CE} + w_{Dice} L_{Dice}$$

4.3.2. Early Stopping

Early stopping is a form of regularization used to avoid overfitting during the training phase. This is a method that allows you to specify an arbitrary number of training epochs and stop training once the model performance stops improving on a holdout validation dataset.

To regularize the training loss and validation curve, we use calculating values with the Moving Average method to ensure that we also give importance to the history of the loss:

$$Loss_{validation} = \alpha L_{MA} + (1 - \alpha) L_{val}$$

Where alpha represents the weight of the previous Loss values. This value is also used to save the best model to use in the inference phase.

The MA of the loss of the training is instead used to calculate the count of the epochs without improvements. An epoch without improvement is such if the loss of the latter does not exceed the value of the previous epoch plus a minimum threshold: $L_{trainMA} + \epsilon$.

Also, the training does not end until at least a certain value of the learning rate has been reached.

4.3.3. Implementation details

Hyperparameters are shared by binary networks and multi-class ensembles. For memory reasons the batch is set to 2. Using an approach taken from Leslie N. Smith's article [43], an optimal learning rate was found with an initial value of 0.001. The dropout rate of the segmentation networks has a value of 0.1. ReduceLROnPlateau was chosen as the scheduler for managing the learning rate, with parameters $\epsilon = 0.001$, patience = 4 and weight decay = 0.0000001.

The α parameter for the training MA and validation (seen in Section 4.3.2) is 0.5. This means that history is given equal importance to the new loss values. The threshold ϵ for the early stopping count is 0.001, while the patience is 5.

Following the tests shown in Section 5.1, the most performing weights for CrossEntropy Loss are all 1. The weights of the composite loss are also set to 1, in order to give an equivalent weight to both single losses.

For DeepLab we use a rate for the atrous convolution of (12, 24, 36), and for the backbone a resnet34. In all networks, the reference literature [19, 20, 39] was followed. Table 4.1 shows in detail the input and output dimensions of all the models used.

Model	Input	Output
Unet	[2, 1, 320, 320]	[2, 7, 320, 320]
Se-ResUnet	[2, 1, 320, 320]	[2, 7, 320, 320]
DeepLabV3	[2, 1, 320, 320]	[2, 7, 320, 320]
1x1 conv stacking	[2, X, 320, 320]	[2, 7, 320, 320]
Unet ensemble	[2, X, 320, 320]	[2, 7, 320, 320]
Feature Fusion	[2, Y, 320, 320]	[2, 7, 320, 320]

Table 4.1: Input and output size used during training by models. 2 is the batch size we have decided to use, while the second number represents the channels we have decided to use. X is the number of input networks we use for a specific problem, while Y is the number of chained features $(64 * N_{Unet} + 64 * N_{SeResUnet} + 256 * N_{Deeplab})$

Python 3.8.10 is the main programming language of this implementation, using Pytorch 1.10 as the main library. We used [9] as a basis for the Unet implementation, [6] regarding Se-ResUnet and [2] for DeepLavV3. Reference has been made to the logic of the implementation of nnUnet [5] regarding the training component.

The training of the binary models took place mainly on the Paperspace Gradient hardware, a cloud computing service that provides several video cards for this kind of task. In particular, we used an NVIDIA Quadro P4000 with 8GB of memory. The ensemble models, given the memory limit imposed by the service, were trained on local hardware on a GeForce GTX 1060 with 6GB of memory.

4.4. Evaluation metrics

In this section, we present the metrics used for model evaluation and comparison. These are all widely used metrics in the literature regarding the semantic segmentation task. In addition to the well-known Precision and Recall, we use region-based metric functions that calculate the overlap regions or the difference in boundaries. The Dice Similarity Coefficient is the metric we refer to most in this work.

Precision and Recall

In the context of binary classification, the confusion matrix is defined as a table of four different combinations of predicted and real values. We say True Positive / Negative (TP / TN) to indicate a value that is rightly classified as positive / negative and False Positive

/ Negative (FP / FN) for values that are wrongly classified.

We then define the Recall metric, which evaluates among the positive classifications how many are actually positive, as:

$$P = \frac{TP}{TP + FP}$$

and the Precision metric, which computes the number of positive samples rightly classified as positive, is:

$$R = \frac{TP}{TP + FN}$$

Dice Similarity Coefficient

The Dice Similarity coefficient (DSC), also called the Sørensen index, is a statistic used to measure the overlap between two samples (typically the target and the prediction). The value is in the range of 0 to 1, where 1 denotes a complete and perfect overlap. Recalling the confusion matrix of the previous subsection, the DSC can be written as:

$$DSC = \frac{2TP}{2TP + FP + FN}$$

Hausdorff Distance 95

The maximum Hausdorff distance (HD) is the maximum distance (in mm) of one set from the closest points of another set. A graphical insight is shown in Figure 4.14.

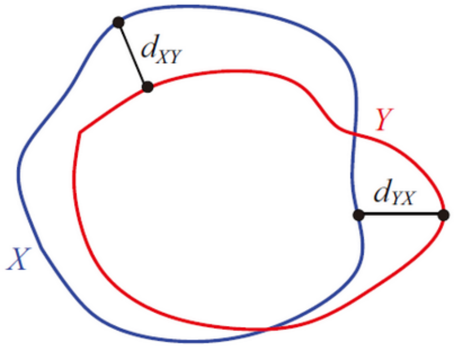


Figure 4.14: Hausdorff distance

Formally, HD from set X to set Y is defined as:

$$d_H(X,Y) = \max(\max_{x\in X}\min_{y\in Y} d(x,y), \max_{y\in Y}\min_{x\in X} d(x,y))$$

95% HD is similar to maximum HD. The only difference is that it is calculated on the 95th percentile of the distance between the boundary points in X and Y. The purpose of this is to eliminate the impact of a small subset of outliers.

4.5. Summary

In this chapter we have seen in general how it works and what are the bases on which our work is carried out. We first saw how the input pipeline and the initial transformations work. The architectures used were presented and exposed in detail, focusing mainly on our ensemble implementations.

We have given an overview of the training algorithm and the optimizations used, of the loss functions used in the update step and of all the details necessary to allow the reproducibility of our work.

At the end of the chapter we have described the important evaluation metrics to measure as accurately as possible the quality of the segmentation of the different architectures and compare the results.

In this paper we will use abbreviations for each ensemble network: feature fusion (Last-layer feature fusion), Unet meta-model (Unet ensemble stacking) e 1x1 convolution (1x1 convolution ensemble stacking).

5 | Design of Experiments

In this section we see how the experiments concerning the architectures described in Section 4.2.2 were conducted. In particular, it is evaluated how the specific type of ensemble performs with respect to the baseline models taken into consideration.

First is shown, in Section 5.2, the trained binary networks divided by architecture and organ, which are used in ensemble models. Section 5.3 contains details on the implementations that we consider baseline, i.e. binary networks trained on a single organ and multiclass networks trained on all organs simultaneously. Section 5.4 instead contains the description of the experiments carried out to observe the behavior and performance of our ensemble models on different hypotheses and composition of networks or datasets. As a matter of convenience, each experiment is listed with a *TX* name, where X is the number of the test, for quicker citation.

Each experiment is associated with a unique ID, whether it has been driven on online platforms or on local PCs. For each training, all the information necessary to recognize the type of system used was saved: training dataset, organs to be segmented, pre-trained models (if used), the various training loss and validation loss for each epoch together with the calculated Moving Average , optimizer used, all main hyperparameters. Depending on the type of architecture used, more information may have been saved.

To choose the binary and multiclass networks to be used in the ensemble experiments, we used the networks that appear to have the best performance in validation phase.

A total of 25 ensemble experiments were performed, for as many trained ensemble networks.

5.1. Parameter Tuning

We have performed some tests to check the performance as some parameters in the training change. Tests include using different CE loss weights and a different type of optimizer.

Small tests were performed on the type of loss function for binary models (dice, composite, CE, focal) to be used and on the optimizer (Adam, RMSprop). We decided to use a composite loss and Adam optimizer.

CE Weights Intuitively, the CE loss should be weighted (see Section 4.3.1) so as not to penalize the organs present in fewer numbers in the dataset. The networks tested are the last-layer feature fusion model used in this work.

Below are the weights tested in the format $(x_0, x_1, x_2, x_3, x_4, x_5, x_6)$ where x_i represent the i-th target in StructSeg in the order [Background, Right Lung, Left Lung, Heart, Trachea, Esophagus, Spinal Cord]:

- (1, 1, 1, 1, 1, 1, 1): the standard CE application
- (1, 1, 1, 1, 2, 2, 2): to give more weights to the smaller organs
- (1, 1, 1, 1, 15, 5, 10): as suggested in [19] with the same dataset
- (0.01, 1.2, 0.92, 2.51, 30, 48, 21): the inverse of the total number of pixels per organ

Looking at the evaluation metrics for each body, the best result is obtained by the standard CE. For detailed results see Appendix B.

5.2. Binary networks

Binary networks serve as the basis for ensemble that are trained later, which means that the result of the subsequent ones strongly depends on these.

We trained each implemented architecture described in Section 4.2.1 with every organ in single-class problem. As for the StructSeg dataset the training was carried out on all organs, while for SegTHOR we focused on the organs in common with the first dataset i.e. Heart, Trachea and Esophagus. An overview of all the combinations is shown in Tab. 5.1.

Organ	Model	Dataset
Left Lung	Unet, SE-ResUnet, DeeplabV3	StructSeg
Right Lung	Unet, SE-ResUnet, DeeplabV3	StructSeg
Esophagus	Unet, SE-ResUnet, DeeplabV3	StructSeg, segTHOR
Trachea	Unet, SE-ResUnet, DeeplabV3	StructSeg, segTHOR
Heart	Unet, SE-ResUnet, DeeplabV3	StructSeg, segTHOR
Spinal Cord	Unet, SE-ResUnet, DeeplabV3	StructSeg

Table 5.1: Binary networks, all the trained combinations

We have chosen the best validation performing network (Dice score) to be used in ensemble training. The best architectures for each organ of each dataset are shown in Table 5.2.

Organ	Net	Loss
Left Lung	SE-ResUnet	0.0319
Right Lung	SE-ResUnet	0.0397
Esophagus	SE-ResUnet	0.1744
Trachea	DeeplabV3	0.2395
Heart	DeeplabV3	0.0661
Spinal Cord	SE-ResUnet	0.1113

Organ	Net	Loss
Esophagus	DeeplabV3	0.1346
Trachea	DeeplabV3	0.1154
Heart	DeeplabV3	0.0405

Table 5.2: Best binary networks in validation: StructSeg (left) and SegTHOR (right)

5.3. Baseline

This section describes the implementations used as a baseline to compare the performance of subsequent ensemble and combination results. Specifically, we think that the basic multiclass segmentation networks and the ArgMax combination of the results of the binary networks is a good basis on which to work to compare the evaluation metrics.

5.3.1. Multiclass

All architectures were trained for both datasets with Crossentropy Loss (described in Section 5.1). Table 5.3 shows the trained single-model multiclass networks, along with the best Loss validation result.

Model	Loss
Unet	0.0182
SE-ResUnet	0.0192
DeeplabV3	0.0143

Table 5.3: Multiclass networks and their best validation loss

5.3.2. Binary outputs Argmax

Evaluating this approach allows us to understand how well the single trained binary networks are able to perform well using the simplest result combination method: select the maximum result for each pixel with respect to the class. In table 5.4 we see all the network tests that are used to validate the subsequent results. In these combinations the networks that perform best in validation in the training phase are used.

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet
Trachea	DeeplabV3
Heart	DeeplabV3
Spinal Cord	SE-ResUnet

Table 5.4: Single components of the considered ArgMax

5.4. Ensemble analysis

Relying on the idea that a single network can better exploit the unique information of the corresponding organ, a pipeline of experiments has been built regarding our ensemble architectures. For each type of experiment proposed below, in addition to describing the details, the reasons that led us to direct the research in that direction are explained. Each experiment was carried out for all 3 ensemble networks considered: 1x1 convolution ensemble stacking, Unet ensemble stacking and Last-layer feature fusion.

5.4.1. T1 - Basic training

Pre-trained networks are used in binary segmentation problem on the single organ, then combined in order to obtain the desired multiclass segmentation. This type of model is trained using the binary networks which have the best result in the validation in the training phase. The networks shown have been trained on StructSeg datasets only. Table 5.5 show the used trained models.

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet
Trachea	DeeplabV3
Heart	DeeplabV3
Spinal Cord	SE-ResUnet

Table 5.5: Single components of basic ensemble training on StructSeg

Given the peculiarity of the feature fusion ensemble network, we are also able to experiment by "unfreezing" the gradient of several networks and leaving it free to fine-tune. In this case we test how the metrics change in the case of fine-tuning on the trachea and esophagus.

5.4.2. T2 - Multiple nets on same organ

This experiment is based on the fact that one of the main problems of semantic segmentation on medical images is the detection of small organs and tissues and, given the nature of CT scans, often with little contrast to the surrounding tissues. Using single nets trained on one organ can help alleviate this problem, but we want to see what happens if we use multiple binary nets trained on the same organ to ensemble in a multiclass problem. In this case we use the same networks seen in the experiment in Section 5.4.1, adding a second model among those that perform best for each organ that is more difficult to segment i.e. Trachea and Esophagus in this case. Tables 5.6 are a list of the used already-trained binary networks.

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet + DeeplabV3
Trachea	DeeplabV3
Heart	DeeplabV3
Spinal Cord	SE-ResUnet

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet
Trachea	DeeplabV3 + Unet
Heart	DeeplabV3
Spinal Cord	SE-ResUnet

Table 5.6: Basic ensemble training with multiple nets on most difficult organs: Esophagus (left) and Trachea (right)

A similar type of experiment is done with both additional networks in the same training.

5.4.3. T3 - Multiple nets trained on different dataset

SegThor dataset has some organs in common with StructSeg. We use networks trained on these common organs in our ensemble models, instead of networks trained on the same inference dataset. In this case we see how the ensemble architectures behave when one of the targets does not have a binary dataset on which to train the network, thus having to use an external dataset. Again, the trachea and esophagus are used. The list of networks is present in the Table 5.7.

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet
Trachea	DeeplabV3 (SegThor)
Heart	DeeplabV3
Spinal Cord	SE-ResUnet

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet (SegThor)
Trachea	DeeplabV3
Heart	DeeplabV3
Spinal Cord	SE-ResUnet

Table 5.7: Basic ensemble training with multiple nets trained on different db on most difficult organs: Esophagus (right) and Trachea (left)

5.4.4. T4 - Multi-class and multi-binary

The ease with which it is possible to add inputs to our ensemble architectures allows us to experiment with compositions of multiple segmentation networks, not limited to binary ones. In fact we also test the unique combination between the binary networks used for the base plus a multiclass network, chosen on the basis of the best performing one among those trained (Section 5.3.1). We expect to have a better performing segmentation as the multiclass network adds information about the disposition of the targets that is not present in the usual binary networks. Table 5.8 shows the composition of the networks used.

Organ	Model
Left Lung	SE-ResUnet
Right Lung	SE-ResUnet
Esophagus	SE-ResUnet
Trachea	DeeplabV3
Heart	DeeplabV3
Spinal Cord	SE-ResUnet
Multiclass	DeepLabV3

Table 5.8: Multiclass and Binary networks ensemble training on StructSeg

5.4.5. T5 - Reduced training dataset

This is to test how the models behave on a hypothetical reduced ensemble dataset. The binary networks used are trained on the complete dataset, while the ensemble networks use 20% of the training dataset. This is to test simulation of a situation where you have large binary datasets available but a very limited number of labeled multiclass samples.

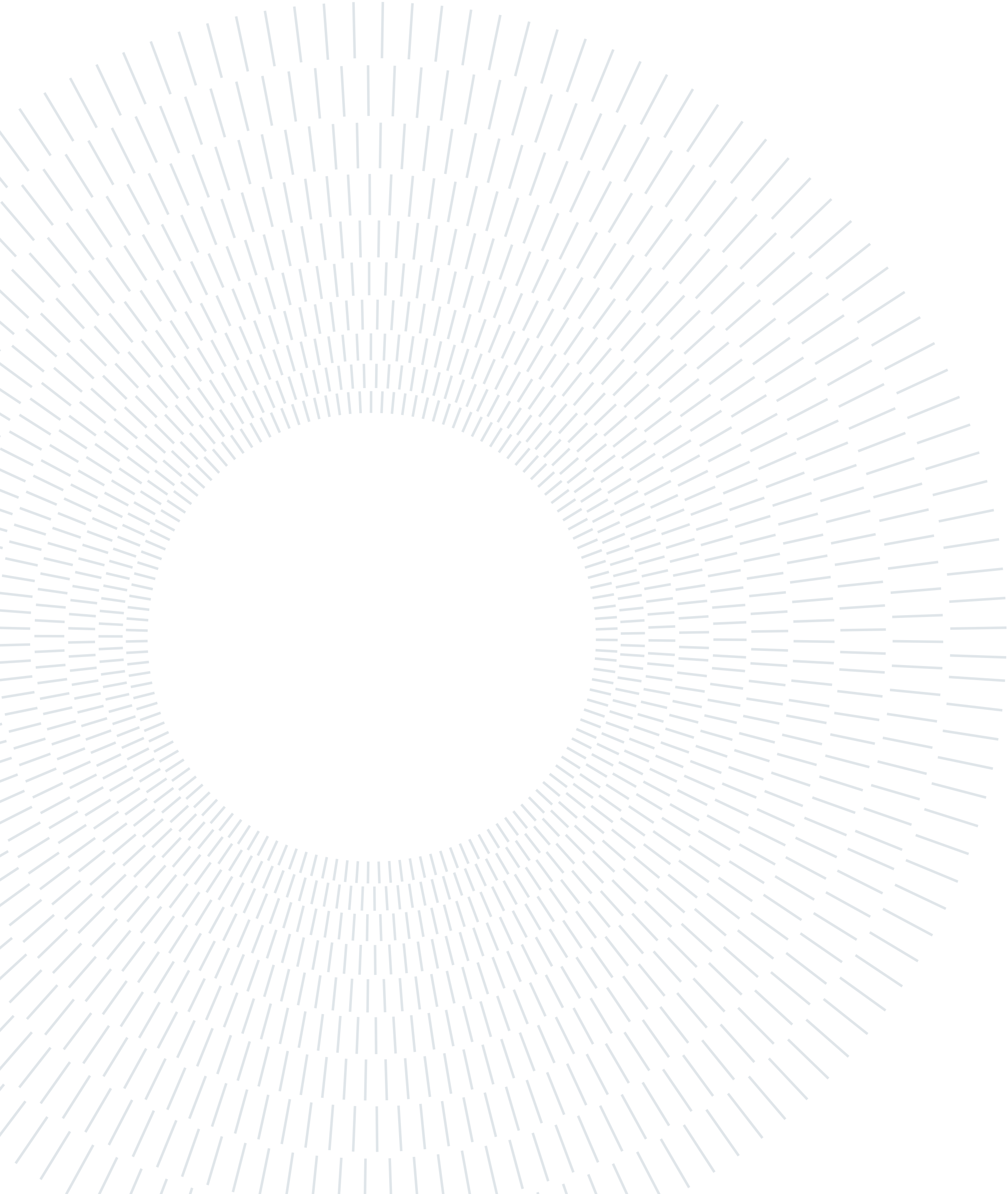
The binary models used are the same as those present in Section 5.4.1.

5.5. Summary

In this chapter we have presented the main experiments performed in this work.

Starting from the observations on some technical tests to evaluate different hyperparameters, the experiments of the basic networks that were used for the subsequent tests are shown. After reporting the baseline experiments performed for comparison, we came to organize the ensemble experiments into main categories, depending on the type of network combination used.

In the next chapter we will provide the results of these experiments complete with evaluations and comments on them.



6 | Results

In this chapter we report the results of our work. In Section 6.1 we summarize the performances obtained by all the trailed models divided by the type of architecture. In the section 6.2 we show the quality of the segmentation from a visual point of view of the same experiments.

Finally, in Section 6.3, we discuss in a general overview the results obtained. As in Section 5, the results obtained are shown with the name of the related test i.e. TX is "Test number X".

6.1. Quantitative results

In this section we show the quantitative results of the experiments described in Section 5. Since our aim is to evaluate and compare the performance of the different ensemble architectures described in Section 4.2.2 under different conditions, for each network we show the mean value of the metric and the (worst, best) case by volume. To aid comparison, we also report the performance of our baseline models in each experiment.

The boxplot graphs help to make a finer evaluation organ by organ. The metrics shown are Dice and HD95, as they are more significant for the segmentation problem. In addition to the results of the ensemble networks, these graphs also show the performance of our baseline to have a good overall picture of the situation.

All reported nets refer to nets trained on multi-organ dataset.

Baseline

Table 6.1 shows the result of the evaluation metrics described in Section 4.4 on test datasets. This includes all single architecture multiclass networks described in Section 5.2 and the ArgMax configuration described in Section 5.3, that are our baselines.

Argmax certainly shows a better result than FCN multiclass networks, followed shortly by DeepLabV3. The biggest difference is shown in Figure 6.1, where the argmax far outweighs the other models in smaller organs.

Model	Dice	Precision	Recall	HD95
Unet	0.8243 (0.779,0.874)	0.87 (0.768,0.921)	0.8051 (0.709,0.885)	4.5729 (2.615,9.848)
SE-ResUnet	0.8172 (0.776,0.864)	0.8403 (0.732,0.902)	0.8182 (0.71,0.888)	6.3475 (2.528,16.469)
DeepLabV3	0.8512 (0.799,0.872)	0.8556 (0.79,0.909)	0.8536 (0.767,0.894)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	0.8581 (0.791,0.902)	0.9079 (0.867,0.946)	2.4454 (1.344,3.236)

Table 6.1: Baselines - Average performances on test set

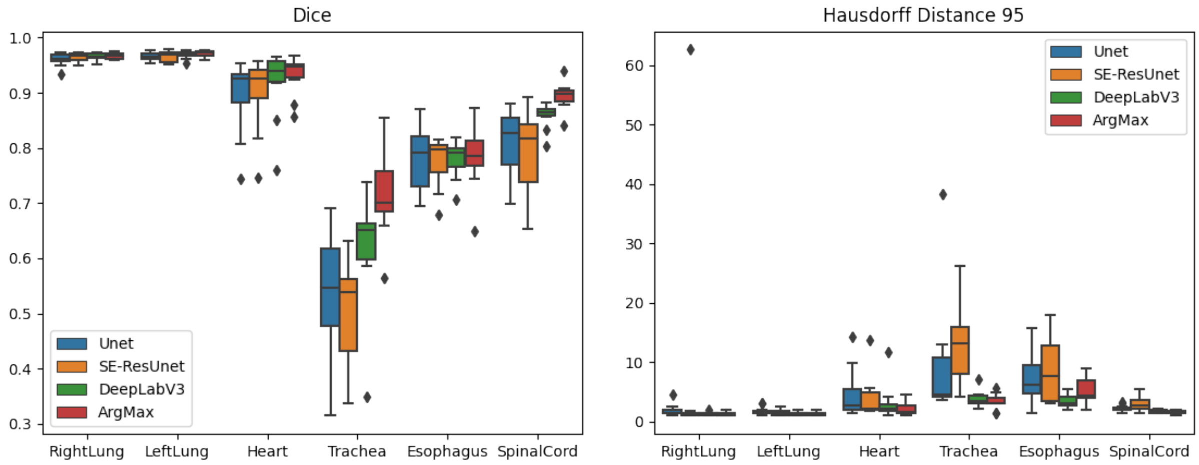


Figure 6.1: Baselines - Metrics visualization

T1 - Basic training

In basic training we see the ensemble networks to make a first direct comparison with the simple combination of outputs. AS we see in Table 6.2, the simpler network, 1x1 convolution of the outputs (we call it "1x1 convolution" for short) has a slightly better overall result than the other networks, including the baseline.

For single organs, we see in Figure 6.2 that there is no significant difference in performance in the segmentation of the largest (lungs and heart). For the little ones, however, the difference is mainly noticeable in the esophagus: 1x1 convolution and feature fusion are able to make a more accurate segmentation and make less significant errors.

Model	Dice	Precision	Recall	HD95
DeepLabV3	0.8512 (0.799,0.872)	0.8556 (0.79,0.909)	0.8536 (0.767,0.894)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	0.8581 (0.791,0.902)	0.9079 (0.867,0.946)	2.4454 (1.344,3.236)
Feature Fusion	0.8744 (0.836,0.913)	0.8812 (0.816,0.93)	0.8748 (0.818,0.919)	2.2607 (1.275,2.967)
1x1 convolution	0.8794 (0.851,0.913)	0.8689 (0.806,0.912)	0.8986 (0.856,0.939)	2.2941 (1.305,3.087)
Unet meta-model	0.8661 (0.828,0.898)	0.9005 (0.852,0.944)	0.8457 (0.792,0.895)	2.3665 (1.413,2.962)

Table 6.2: Experiment 1 - Average performances on test set

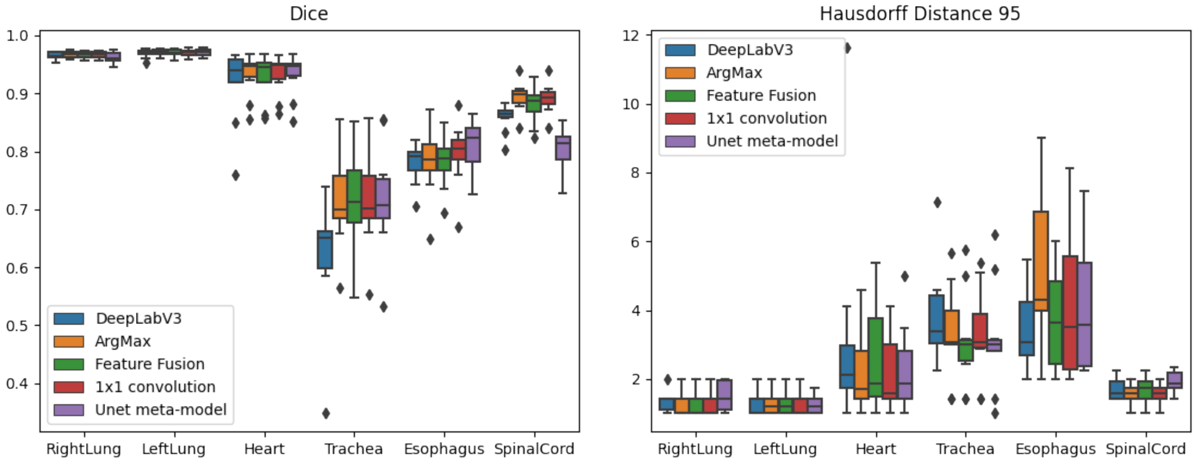


Figure 6.2: Experiment 1 - Metrics visualization

T2 - Multiple nets on same organ

By using multiple networks for the same target, we expect a better generalization of the model [24, 26]. In this experiment we use specific networks for different organs, therefore they should perform better on test sets than the model without additional networks. Specifically, we use the additional networks on the organs that are more difficult to segment, i.e. esophagus and trachea.

First, experiments are performed on individual organs, then both together.

Esophagus

In Table 6.3 we see how the 1x1 convolution and the feature fusion still have a higher result than the baseline, which increases slightly compared to the experiment without the additional network. On the other hand the average result of the network with Unet meta-model decreases by more than 2%.

Taking the organs individually, in Figure 6.3 we see how the increase in average performance is potentially due to the better result for each organ. While the situation does not change much for trachea, in the esophagus there is a visible improvement in the main metrics regarding 1x1 convolution and feature fusion. For Unet meta-model the situation is different: despite the additional network, the results is decreased both in the Dice and in the HD95: the difference in precision and recall change confirms the greater difficulty of the model in correctly classifying organs, increasing false positives.

Model	Dice	Precision	Recall	HD95
DeepLabV3	0.8512 (0.799,0.872)	0.8556 (0.79,0.909)	0.8536 (0.767,0.894)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	0.8581 (0.791,0.902)	0.9079 (0.867,0.946)	2.4454 (1.344,3.236)
Feature Fusion	0.8807 (0.844,0.92)	0.8861 (0.825,0.931)	0.881 (0.819,0.931)	2.1333 (1.138,2.982)
1x1 convolution	0.8824 (0.856,0.918)	0.8806 (0.817,0.919)	0.8899 (0.842,0.935)	2.133 (1.305,2.669)
Unet meta-model	0.8414 (0.804,0.863)	0.9268 (0.878,0.961)	0.7897 (0.729,0.828)	2.6101 (2.068,3.625)

Table 6.3: Experiment 2 (esophagus) - Average performances on test set

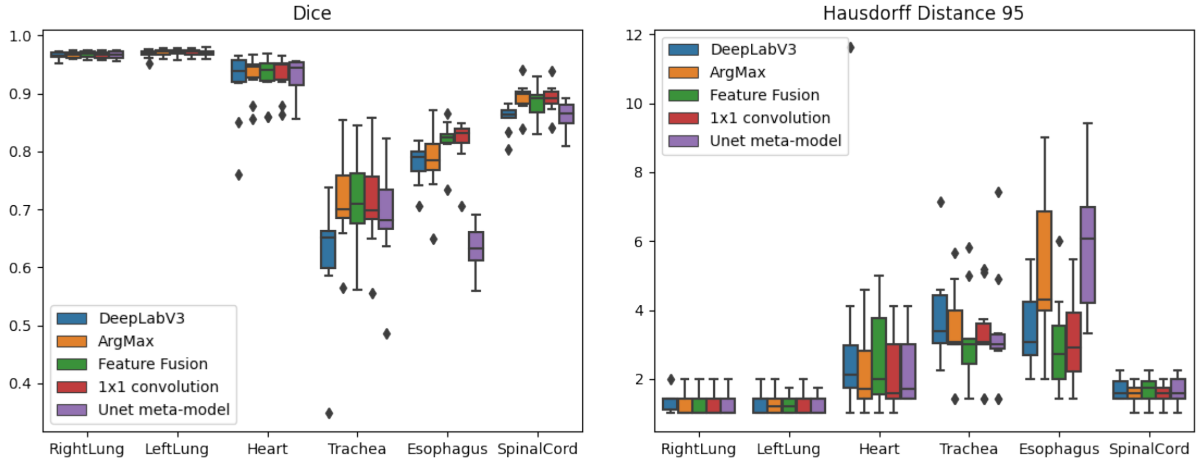


Figure 6.3: Experiment 2 (esophagus) - Metrics visualization

Trachea

In Table 6.4 there is a slight improvement in the average metrics for all networks compared to the base test. The fusion feature improves the dice by 2% on the trachea (Figure 6.4), while the other models remain almost unchanged. Even though the HD95 has a slight improvement for each network, we notice a substantial increase in variance. The results of the other organs are mostly unchanged.

Model	Dice	Precision	Recall	HD95
DeepLabV3	0.8512 (0.799,0.872)	0.8556 (0.79,0.909)	0.8536 (0.767,0.894)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	0.8581 (0.791,0.902)	0.9079 (0.867,0.946)	2.4454 (1.344,3.236)
Feature Fusion	0.8848 (0.848,0.918)	0.8902 (0.829,0.934)	0.8877 (0.825,0.931)	2.2529 (1.167,3.122)
1x1 convolution	0.8809 (0.845,0.912)	0.8934 (0.835,0.926)	0.8817 (0.831,0.916)	2.3353 (1.236,3.11)
Unet meta-model	0.8795 (0.843,0.905)	0.9053 (0.85,0.936)	0.8674 (0.816,0.903)	2.3714 (1.413,3.166)

Table 6.4: Experiment 2 (trachea) - Average performances on test set

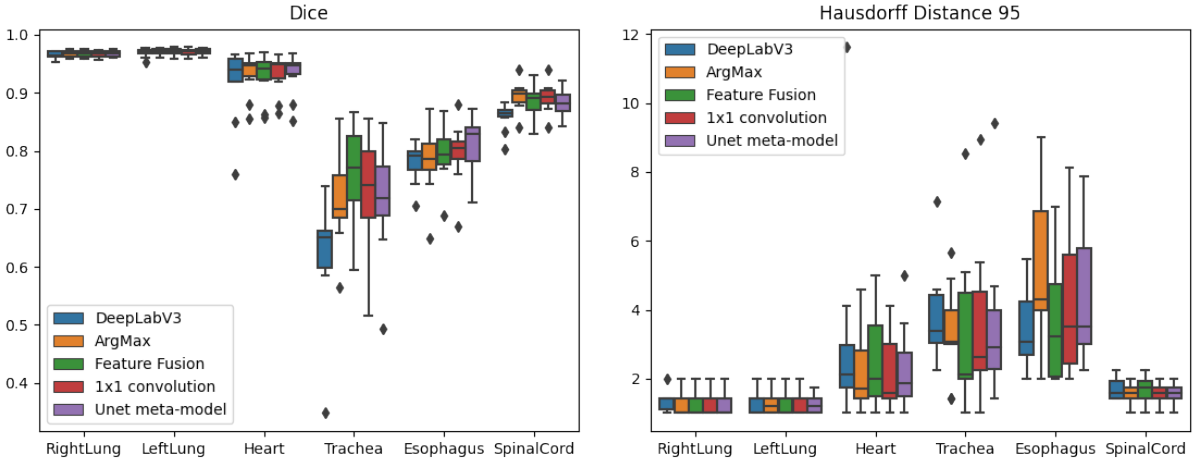


Figure 6.4: Experiment 2 (trachea) - Metrics visualization

Esophagus and Trachea

Table 6.5 shows a good improvement for all the models in all the metrics compared to the base test. The best result for the Dice metrics is the feature fusion, while for the HD95 metric is the 1x1 convolution. Especially for the latter, the improvement is remarkable. Compared to the test with the two networks for the esophagus, the unet meta-model remains stable but in any case does not improve compared to the basic test. In general, for both dice and HD95, it seems an overlap of the results obtained in the previous T2 tests. The other organs remain with almost unchanged results.

Model	Dice	Precision	Recall	HD95
DeepLabV3	0.8512 (0.799,0.872)	0.8556 (0.79,0.909)	0.8536 (0.767,0.894)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	0.8581 (0.791,0.902)	0.9079 (0.867,0.946)	2.4454 (1.344,3.236)
Feature Fusion	0.885 (0.844,0.917)	0.9117 (0.855,0.943)	0.8675 (0.798,0.902)	2.2071 (1.069,3.452)
1x1 convolution	0.8843 (0.85,0.918)	0.9058 (0.846,0.931)	0.8727 (0.817,0.911)	2.1716 (1.236,2.724)
Unet meta-model	0.8774 (0.834,0.919)	0.898 (0.841,0.932)	0.8698 (0.807,0.908)	2.2924 (1.276,3.482)

Table 6.5: Experiment 2 (Esophagus + Trachea) - Average performances on test set

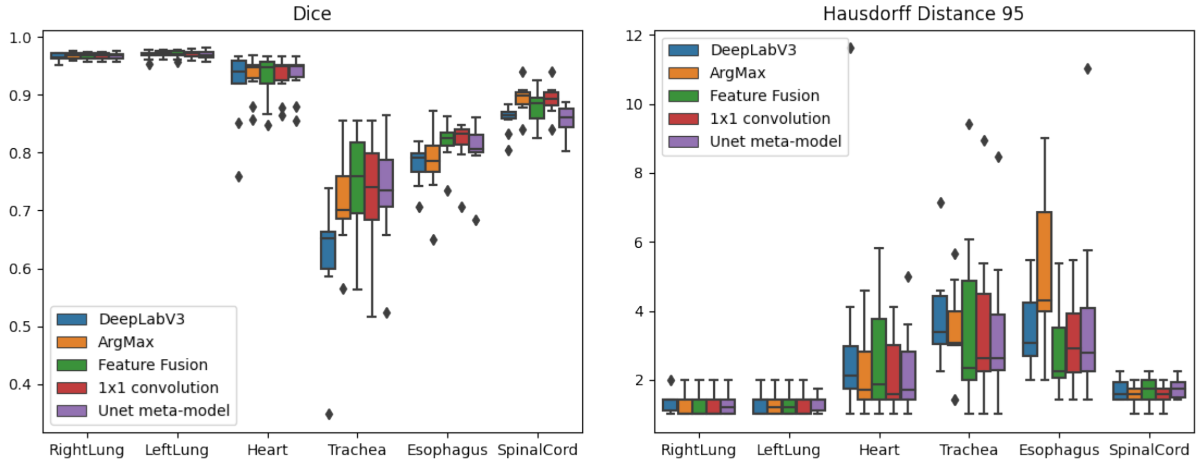


Figure 6.5: Experiment 2 (Esophagus + Trachea) - Metrics visualization

T3 - Multiple nets trained on different dataset

With this settings we want to control the behavior of the ensemble networks when some of the networks have been trained on a different dataset than the test one. In particular, we want to see if the networks manage to balance the poor result of the individual networks with information from other results. Unlike the other experiments, in this subsection are shown the results of the argmax architecture using networks trained on SegTHOR datasets.

Esophagus

Table 6.6 and plot 6.6 show the results of the experiment with the esophagus. The difference is clear: on average only the fusion feature manages to maintain acceptable results. In the individual organs we understand why: the networks have a lot of difficulty in segmenting the esophagus. In dice and in HD95 only the fusion feature, among the ensemble networks, shows good results while the rest does not fully recognize the organ.

This result also greatly affects the trachea: for the argmax the results decrease a lot, a sign that the trachea and esophagus could have overlapping results. For the other networks, however, the results are on average very similar but with lower variance.

Model	Dice	Precision	Recall	HD95
ArgMax	0.6753 (0.647,0.739)	0.7029 (0.636,0.747)	0.6649 (0.624,0.736)	6.7791 (5.669,8.185)
Feature Fusion	0.8678 (0.837,0.901)	0.874 (0.825,0.916)	0.8694 (0.808,0.911)	2.8196 (2.452,3.16)
1x1 convolution	0.7463 (0.717,0.778)	0.7456 (0.69,0.782)	0.75 (0.695,0.795)	21.603 (20.971,22.046)
Unet meta-model	0.7419 (0.706,0.775)	0.7665 (0.718,0.797)	0.7219 (0.66,0.763)	21.6602 (21.04,22.264)

Table 6.6: Experiment 3 (esophagus) - Average performances on test set

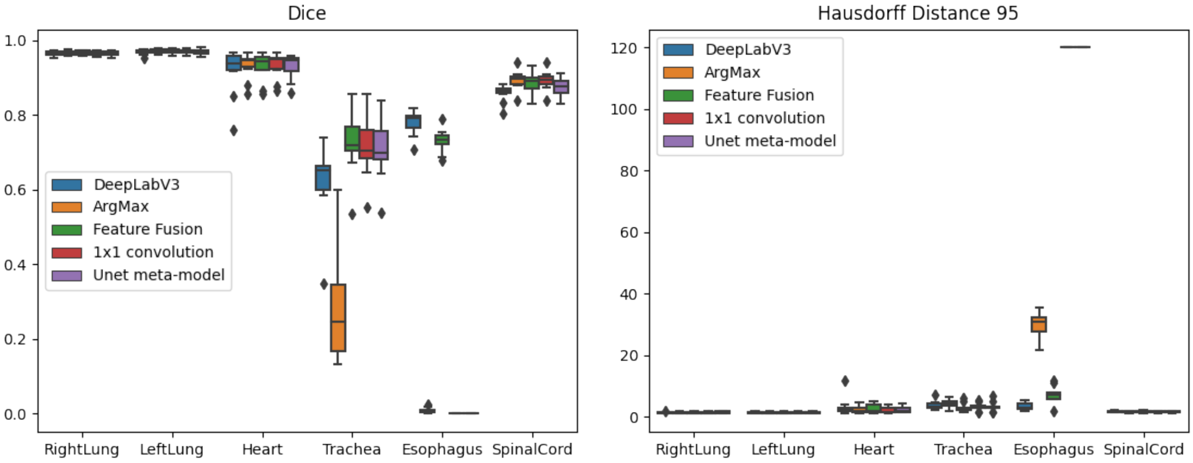


Figure 6.6: Experiment 3 (esophagus) - Metrics visualization

Trachea

Compared to the same experiment (T3) on esophagus, the results are better on average (Table 6.7), but the feature fusion no longer has an excellent result compared to other architectures. In Figure 6.7 we see that argmax and 1x1 convolution fail to segment the trachea at all, while for the Unet meta-model the dice results are slightly better (around 20% total), but still not satisfactory. The fusion feature allows to obtain better results on the organ, reaching almost 50% in dice, and under 20 for the HD95.

No other organs are affected than in the base case.

Model	Dice	Precision	Recall	HD95
ArgMax	0.7578 (0.737,0.772)	0.7478 (0.693,0.777)	0.7791 (0.758,0.795)	7.6132 (5.419,10.289)
Feature Fusion	0.8158 (0.768,0.836)	0.8379 (0.786,0.878)	0.8103 (0.76,0.847)	4.412 (2.793,7.445)
1x1 convolution	0.7597 (0.74,0.775)	0.7552 (0.707,0.783)	0.7713 (0.749,0.79)	21.7617 (21.069,22.6)
Unet meta-model	0.7823 (0.769,0.803)	0.858 (0.821,0.878)	0.7613 (0.741,0.791)	7.3663 (5.52,10.134)

Table 6.7: Experiment 3 (trachea) - Average performances on test set

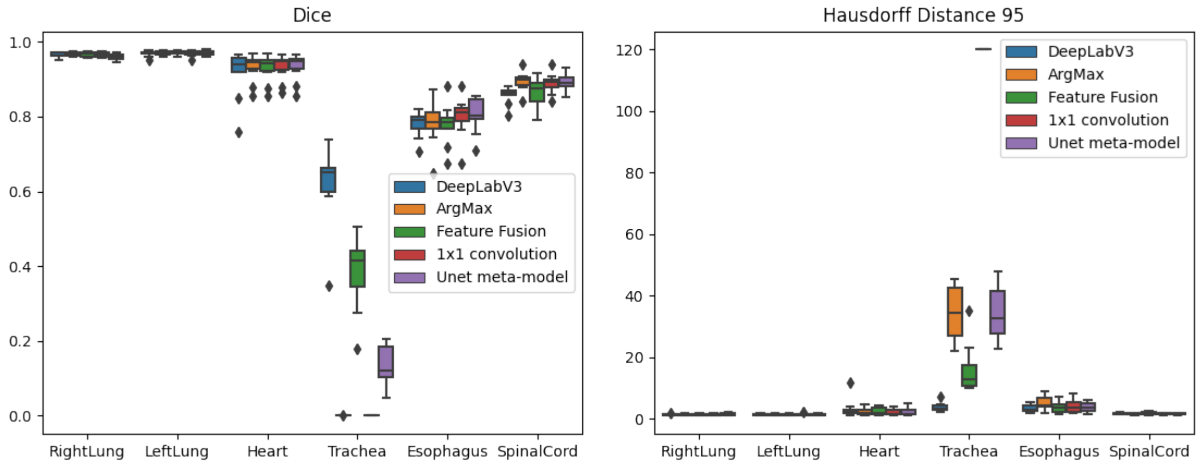


Figure 6.7: Experiment 3 (trachea) - Metrics visualization

The results of this Section are disappointing but expected since the training database of the single networks considered (trachea and esophagus) is different from the test one. We can hypothesize that the distribution of the two datasets is so different that training only on one of the two is not sufficient.

T4 - Multi-class and multi-binary

The addition of a multiclass network in the ensemble architectures of our studio should be useful for binary networks to have a better generalization of all organs on test sets.

Looking at Table 6.8 there is a slightly better performance of all networks for all metrics. Less in the dice, more pronounced in the HD95. Even for single organ the changes are not high: in the dice there is a reduced variance but no significant change in the mean, the same in HD95 except for the esophagus where the situation is very similar to the T2 experiment with the additional network of the esophagus.

Model	Dice	Precision	Recall	HD95
DeepLabV3	0.8512 (0.799,0.872)	0.8556 (0.79,0.909)	0.8536 (0.767,0.894)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	0.8581 (0.791,0.902)	0.9079 (0.867,0.946)	2.4454 (1.344,3.236)
Feature Fusion	0.88 (0.837,0.921)	0.8914 (0.823,0.939)	0.8757 (0.804,0.92)	2.0747 (1.122,2.623)
1x1 convolution	0.8815 (0.855,0.914)	0.8685 (0.801,0.915)	0.9025 (0.857,0.942)	2.1591 (1.305,2.831)
Unet meta-model	0.8668 (0.829,0.904)	0.8888 (0.839,0.935)	0.86 (0.806,0.902)	2.2844 (1.305,3.007)

Table 6.8: Experiment 4 - Average performances on test set

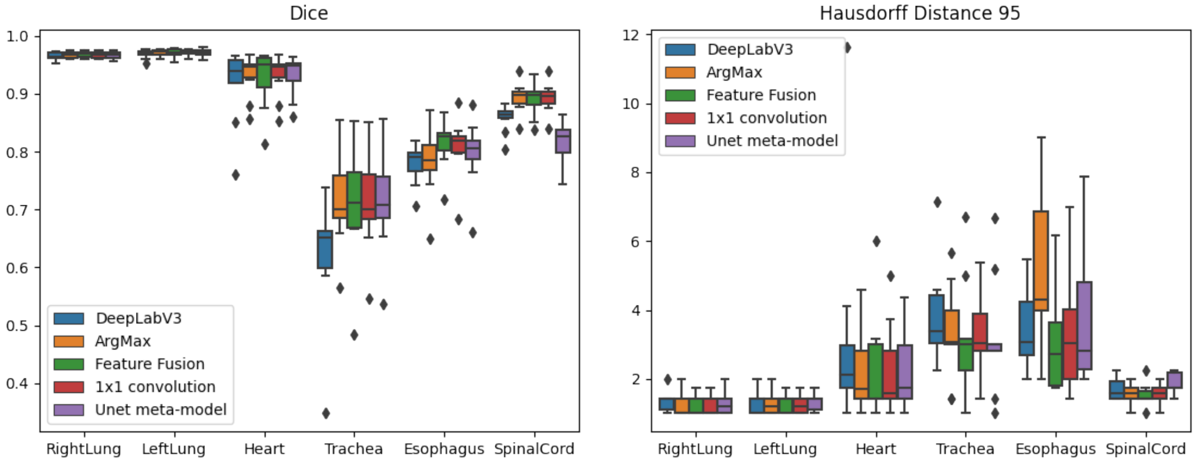


Figure 6.8: Experiment 4 - Metrics visualization

T5 - Reduced dataset

Finally, with this kind of settings we expect a worse situation in ensemble dataset respect to the base experiment. In this case we want to see how worse the segmentation can be and if it outperforms the argmax output, indicating that it is still possible to use a segmentation type with ensemble models that require training even in the presence of a very poor dataset.

In Table 6.9 we see that this time it is the feature fusion model that has the worst performances, as opposed to all the experiments carried out with complete datasets where on average they have the best performances. Although the 1x1 convolution has the best performance with all metrics, it is the Unet meta-model that has the most marked improvement compared to the base case, while the 1x1 convolution actually gets slightly worse.

Figure 6.9 shows the results for each organ. The results with dice metric are comparable with the argmax method, but for the HD95 there is still a predominance of the ensemble methods. Especially for the usual difficult organs, esophagus and trachea, the HD95 improves considerably on average. The results are also comparable with the full dataset training, where there is a slight reduction in the mean of metrics but also a reduction in variance. We highlight also that the spinal cord in the Unet meta-model architecture is segmented with a much better performance if it is trained with a reduced dataset. This indicates a strong possibility of overfitting in the case of a complete dataset.

Model	Dice	Precision	Recall	HD95
DeepLabV3	0.8512 (0.799,0.872)	0.8556 (0.79,0.909)	0.8536 (0.767,0.894)	2.4528 (1.644,3.615)
ArgMax	0.8775 (0.848,0.91)	0.8581 (0.791,0.902)	0.9079 (0.867,0.946)	2.4454 (1.344,3.236)
Feature Fusion	0.8681 (0.827,0.904)	0.8662 (0.809,0.917)	0.881 (0.822,0.941)	2.3664 (1.236,3.597)
1x1 convolution	0.8793 (0.851,0.913)	0.8686 (0.806,0.912)	0.8988 (0.856,0.939)	2.2978 (1.305,3.082)
Unet meta-model	0.876 (0.848,0.911)	0.903 (0.848,0.941)	0.8565 (0.803,0.9)	2.3085 (1.443,2.862)

Table 6.9: Experiment 5 - Average performances on test set

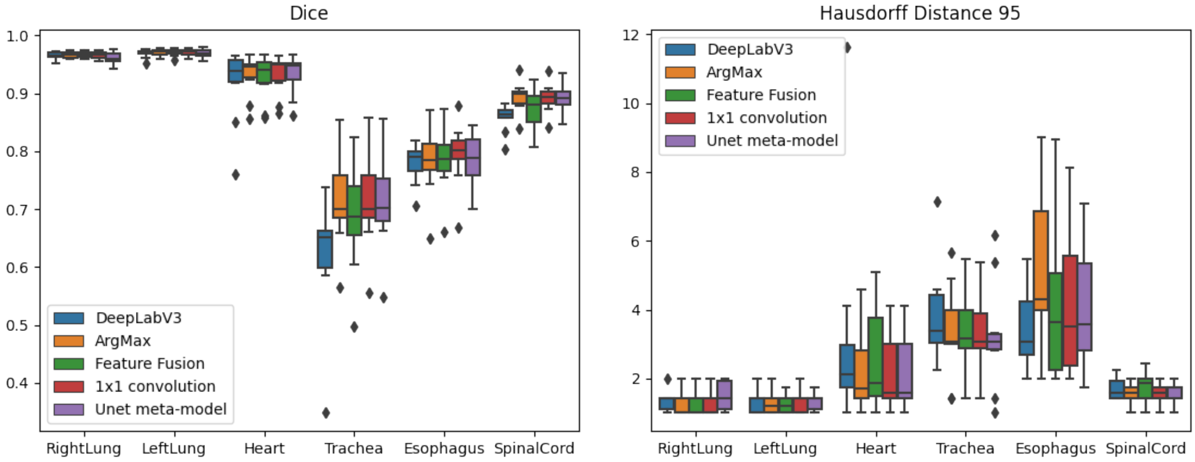


Figure 6.9: Experiment 5 - Metrics visualization

6.2. Qualitative results

This section presents the qualitative result of the segmentation to complete the analysis. The subdivision is by type of experiment i.e. the same used in Section 5.4.

Each image (6.10, 6.11, 6.12, 6.13, 6.15, 6.14, 6.16, 6.17) shows the Ground Truth of the slice in the middle of the volume of a specific patient, next to the outputs of the networks used in this work seen in Section 4.2.2.

T1 - Basic training

Each organ is present and recognized: right in yellow the spinal cord, the trachea divides into two ways that enter the lungs and the esophagus is the small spot in the center.

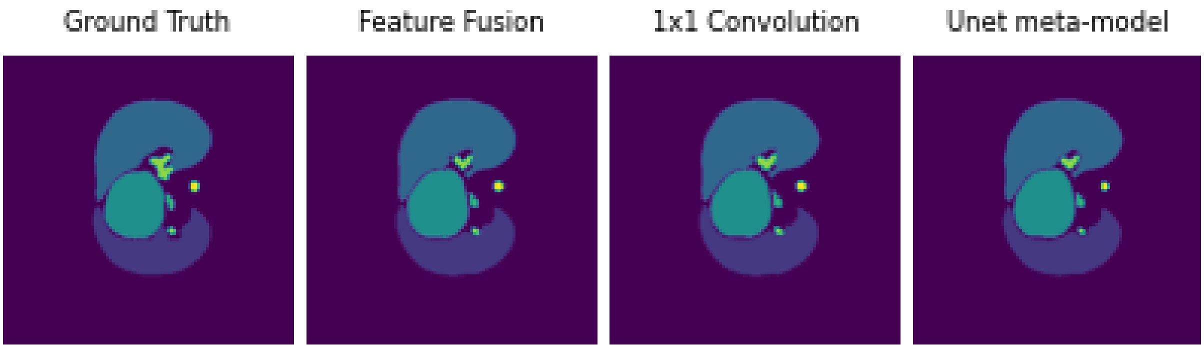


Figure 6.10: Comparison between Ground Truth and experiment T1 predictions

T2 - Multiple nets on same organ

Although the numerical results are better for small organs, visually there is a negative difference from the baseline. This is probably because in these experiments the height of the considered slice is not considered important for the quality of the general segmentation.

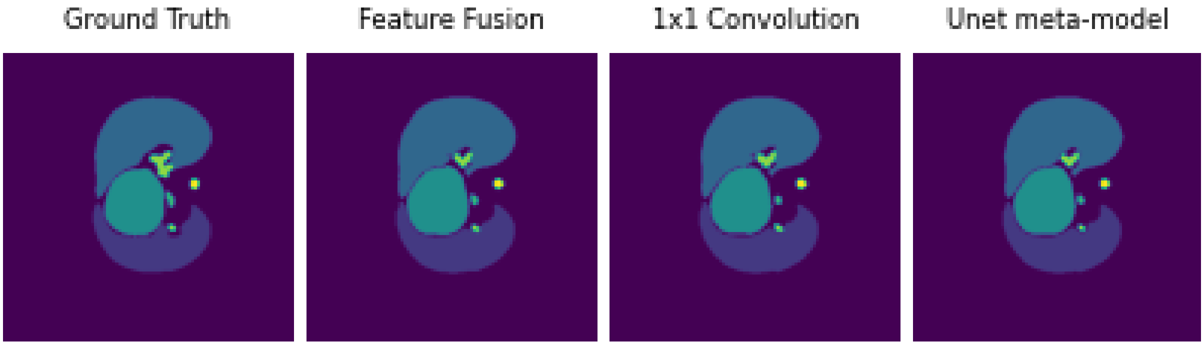


Figure 6.11: Comparison between Ground Truth and experiment T2 predictions (trachea)

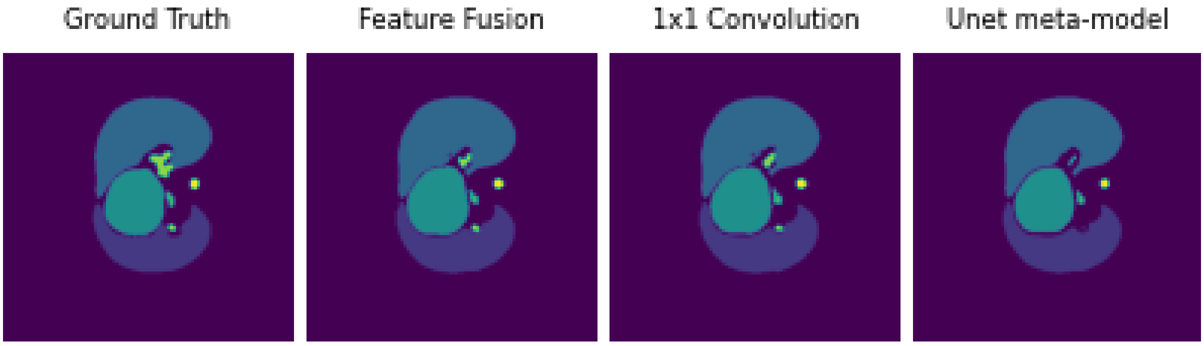


Figure 6.12: Comparison between Ground Truth and experiment T2 predictions (esophagus)

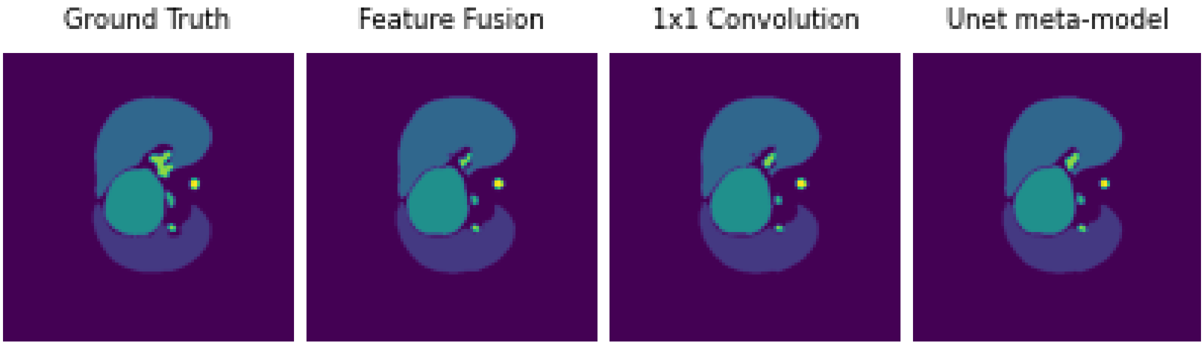


Figure 6.13: Comparison between Ground Truth and experiment T2 predictions (esophagus + trachea)

T3 - Multiple nets trained on different dataset

The results are in line with what was expected from the quantitative analysis. The esophagus and trachea organs are almost completely absent in the final segmentation.

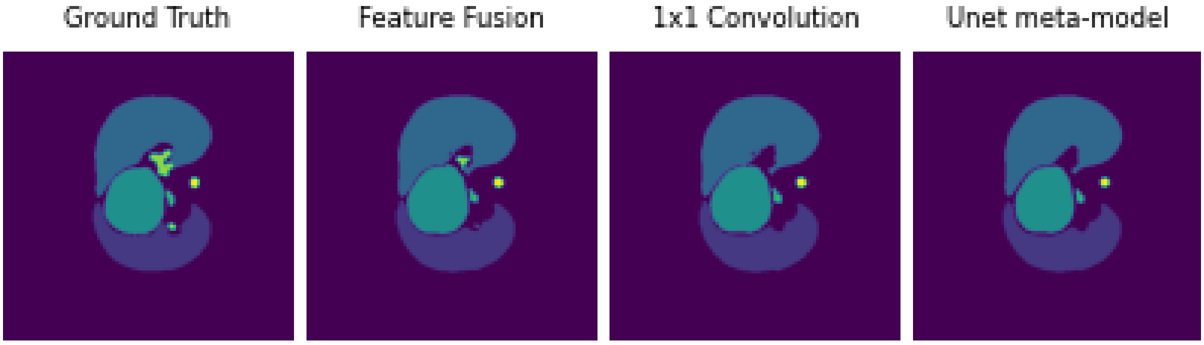


Figure 6.14: Comparison between Ground Truth and experiment T3 predictions (esophagus)

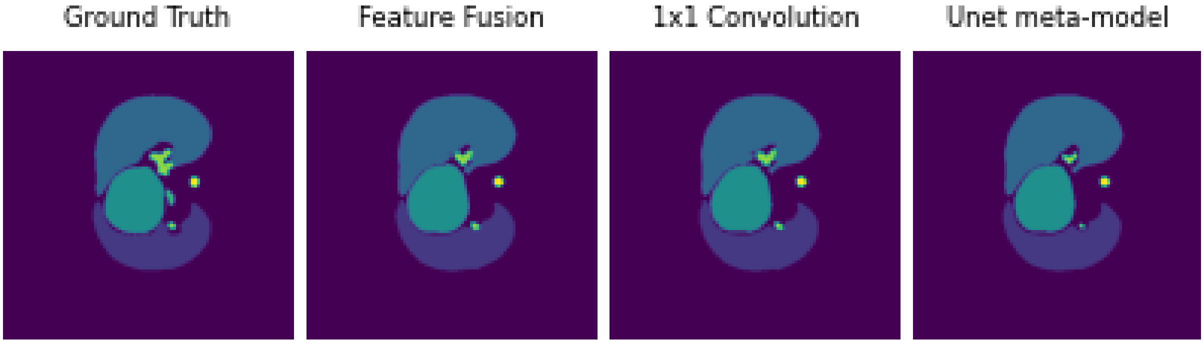


Figure 6.15: Comparison between Ground Truth and experiment T3 predictions (trachea)

T4 - Multi-class and multi-binary

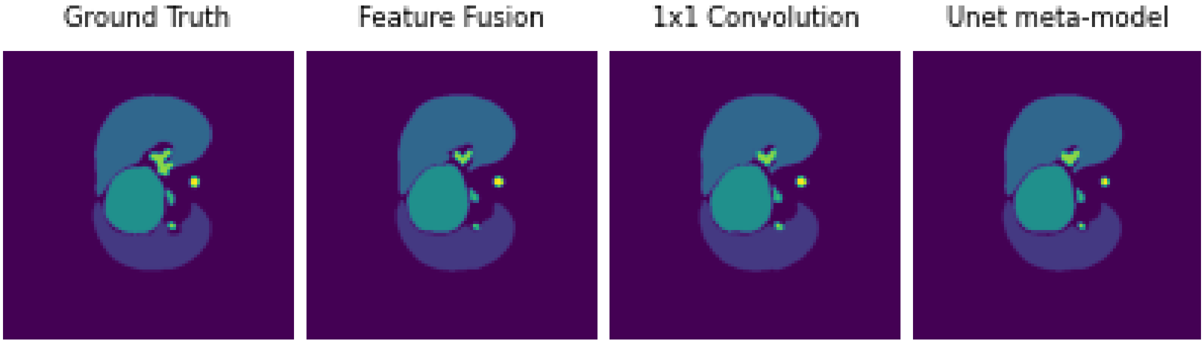


Figure 6.16: Comparison between Ground Truth and experiment T4 predictions

T5 - Reduced dataset

Even though a smaller dataset is used than the other experiments, visually it looks very similar to T1.

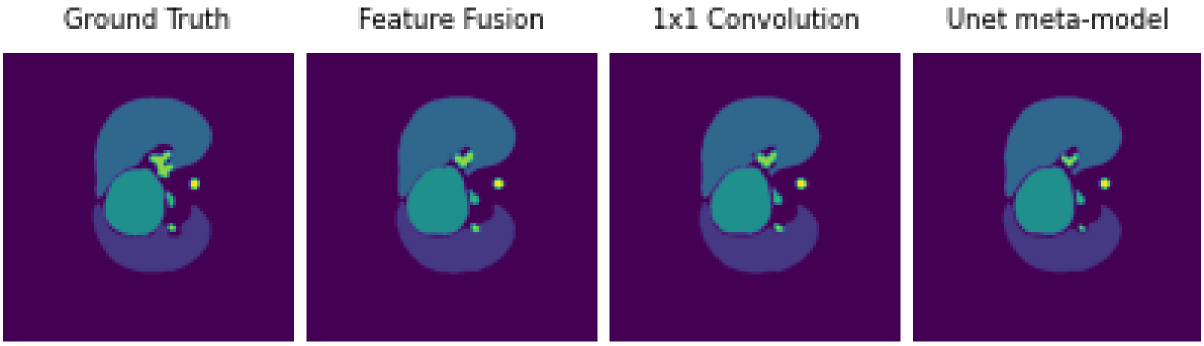


Figure 6.17: Comparison between Ground Truth and experiment T5 predictions

Finally, in Table 6.10 we show the total training time (in epochs) used for each model, before the stop by the early stopping.

Model	Training Epochs
Feature Fusion ($T1$)	11
1x1 convolution ($T1$)	37
Unet meta-model ($T1$)	19
Feature Fusion (T2-e)	13
1x1 convolution (T2-e)	37
Unet meta-model (T2-e)	13
Feature Fusion (T2-t)	13
1x1 convolution (T2-t)	36
Unet meta-model (T2-t)	19
Feature Fusion (T2-et)	17
1x1 convolution (T2-et)	37
Unet meta-model (T2-et)	19
Feature Fusion (T3-e)	13
1x1 convolution (T3-e)	41
Unet meta-model (T3-e)	18
Feature Fusion (T3-t)	12
1x1 convolution (T3-t)	38
Unet meta-model (T3-t)	19
Feature Fusion ($T4$)	12
1x1 convolution ($T4$)	33
Unet meta-model ($T4$)	19
Feature Fusion ($T5$)	14
1x1 convolution ($T5$)	83
Unet meta-model ($T5$)	21

Table 6.10: Total training epochs of trained models

6.3. Results Evaluation

In general, an ensemble architecture allows for better performances than the classic state-of-the-art FCN networks identified as baseline in this work.

Although very simple, the Argmax model has better results than multiclass FCN models. This is also due to the fact that the single model can better exploit the unique information of the corresponding organ, even if the spatial information on the mutual position of all the organs is lost. We think the result is also due to the finer-grained preprocessing, which allows to have images with different improvements depending on the target organ of the single-class network. In some cases, such as the T3 experiment on the esophagus, a worsening of a network corresponding to one organ can also compromise the segmentation performance on different (and closely related) organs, such as the trachea in this case.

Using a Unet meta-model does not always give the best results: often the performances are equal to or lower than the argmax results, and there doesn't seem to be too much difference in using additional networks to improve their segmentation. In some cases, such as the T4 experiment, a worse input for a certain organ can also lead to a lower performance on another organ, apparently unrelated.

The 1x1 convolution network allows for good performances on average even if has a simple structure. Thanks to its intuitive structure and the few learnable parameters (49 for a network with 7 inputs and 7 outputs + biases), it has good results even in the case of a very small dataset. In some cases, however, the evaluation metrics drop to zero when there is not a good quality of the input from one of the binary networks (T3 trachea). As also mentioned in Section 4.2.2, the outputs of the 1x1 convolution have the additional advantage of being easily interpretable: in Appendix A we show the weights associated with the different inputs and how they affect the final segmentation decision.

The architecture that on average performs better than the others is the feature fusion. It also performs well in the case of an input network trained on different data than the ones in test dataset, where it has much better results than the rest of the networks. In case of a small dataset it is always a good choice, but its performances are however slightly worse even than the simple argmax.

Finally, in Table 6.11 we see a comparison between the Dice Score of the best networks of our experiments (t and e are for trachea and esophagus) and the state of the art models presented by the winner of the MICCAI StructSeg2019 challenge [19], the source of the main dataset of this work. The networks to be compared are therefore the main network called Cascaded-SE-ResUnet (C-SE) and the SE-ResUnet (SE) present as baseline in [19].

	DLV3	**Amax**	**T1**	**T2-e**	**T2-t**	**T2-et**	**T4**	**SE**	**C-SE**
Left Lung	96.88	97.10	97.05	97.66	96.72	96.98	97.00	96.70	97.01
Right Lung	96.52	96.69	96.65	96.70	97.00	96.67	96.80	96.28	96.03
Heart	91.74	93.25	93.27	92.91	92.77	92.89	92.71	93.97	94.49
Trachea	61.70	71.83	71.78	76.46	71.73	74.87	71.06	77.02	80.69
Esophagus	77.98	78.22	79.69	79.44	81.99	81.90	81.40	84.04	84.98
Spinal Cord	85.89	89.42	89.92	88.32	88.21	87.68	89.04	89.18	91.00
Mean	85.12	87.75	87.94	88.07	88.48	88.50	88.15	89.31	91.56

Table 6.11: Comparison between our best models vs [19] (C-SE). DLV3 and Amax are DeepLabV3 and Argmax, while the best nets by experiments are: 1x1 convolution (T1, T4), Feature Fusion (T2-e, T2-t, T2-et).

Although the results on the largest organs are very similar to the best network, on the trachea and esophagus the performances are visibly different. Furthermore, even the network taken as baseline by the paper, SE-ResUnet, performs much better than our architectures in some organs: we can assume that, in addition to having room for improvement, the hyperparameters of the training phase and the different hardware performances have influenced the final results.

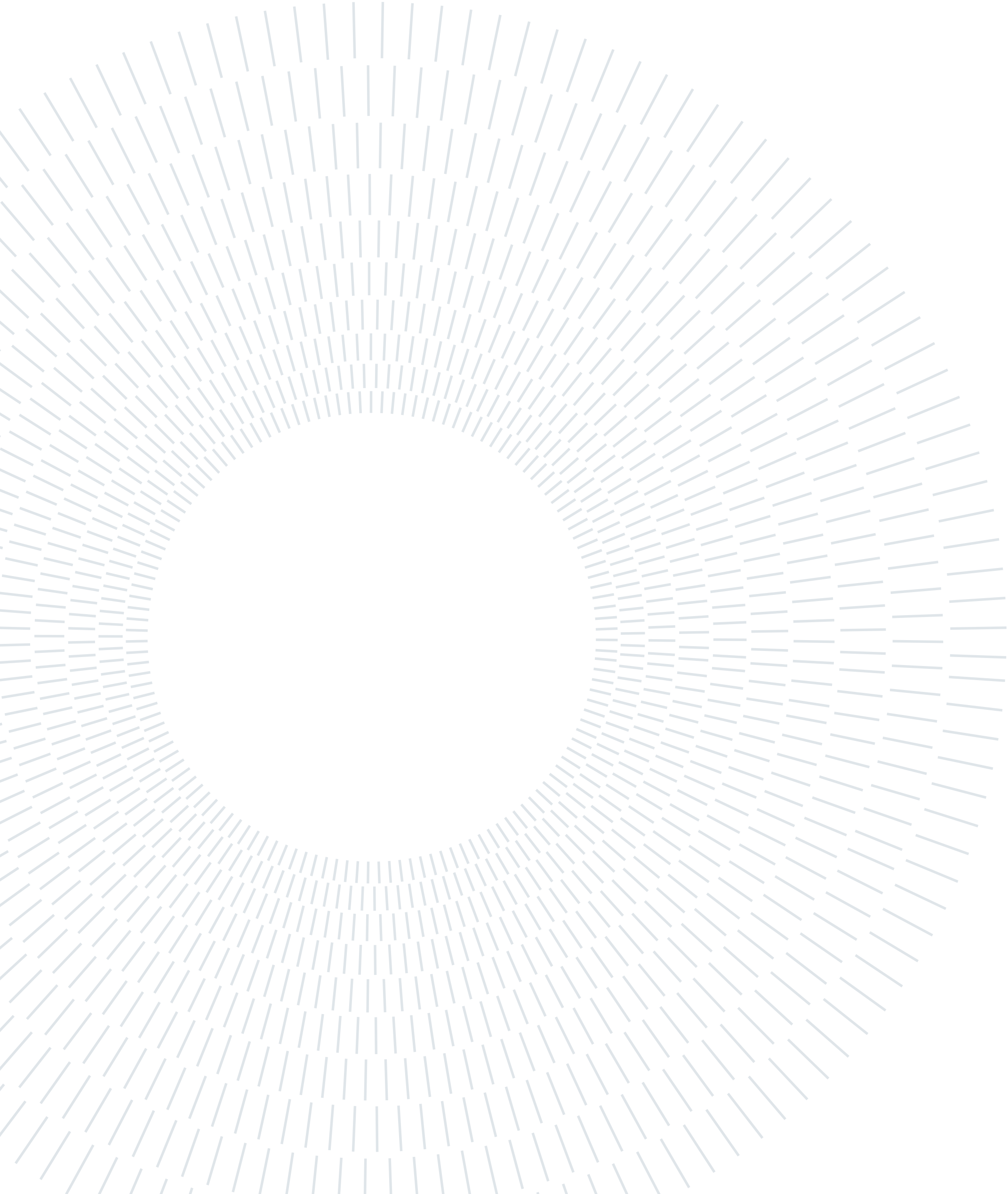
6.4. Summary

In this chapter we have presented all the values of the metrics obtained by evaluating our models on the test set. We have shown the performance of all models according to the experiments described in chapter 5.

Subsequently we showed a comparison through images of all our models always divided according to the experiment in question.

At the end of the chapter we discussed these results in detail, in both qualitative and quantitative ways. We have made some considerations regarding the different use of our different ensemble architectures, both talking about advantages and disadvantages.

In the next chapter we give general considerations on our work and present the open problems we have identified, as well as possible future works.



7 | Conclusions and future works

In Chapter 6 we have shown the results obtained in segmentation performances of our ensemble architecture networks: generated masks are accurate and are able to capture well the structure of the target organs, with a few exceptions involving esophagus and trachea (known to be the most difficult in literature). In some cases the performances show values equal to or slightly lower than [19], the winning method of the StructSeg2019 challenge.

Some types of tests with different settings were performed to demonstrate the applicability of these architectures in realistic situations. The results obtained are encouraging for a future development in the direction of specialized ensemble methods, particularly in absence of large, complete datasets with multi-organ labels. With this type of dataset we have shown how these methods are able to perform better than a simple combination of single single-organ networks despite the time and computational power limitations imposed by costs and logistics.

7.1. Open problems

In this section we present several problems we encountered on this work.

Training performances. Multiple binary networks, when designed from scratch, requires huge amount of time to be trained. This time can be significantly reduced using pre-trained networks already achieving good performances on single or multiple organs. In this case, especially for simpler ensemble networks, the training time is greatly reduced as there are fewer learnable parameters. This is possible also thanks to the exploitation of transfer learning.

Baseline comparisons. This study was carried out using FCN multiclass architecture as a baseline to compare the performances on multiclass datasets. In order to have a broader overview and a better comparison between models, networks of different genres could also be included (e.g. AE, R-CNN, GAN) which for reasons of time and effort have not been included in this work.

Data availability. The amount of data is always a crucial point in machine learning problems. The reasons for this are mainly the difficulty in obtaining labeled medical images due to the tedious work of the labeling process by experts, and the problems of confidentiality and privacy with these data. For these causes, the available public datasets for the purpose of organ segmentation are not many. In addition these datasets usually include scans of a few patients (in our case 50), for a total of only a few thousand images (about 4000 for our dataset): in a complex problem such as organ segmentation, especially abdominal organs, these numbers are not adequate. Both the binary segmentation and the training of the ensemble models could be affected.

Another type of problem is that for this type of work several training datasets would be needed to avoid hauling our models onto correlated structures. In particular, it would be useful to have different datasets available for each organ and a dataset with multi-organ masks, even from the same source. This could lead to independent data model results and improved generalization.

7.2. Possible applications and Future developments

In this section we describe some of what we think are possible future research directions for this work, in order to improve the design and performance of the system.

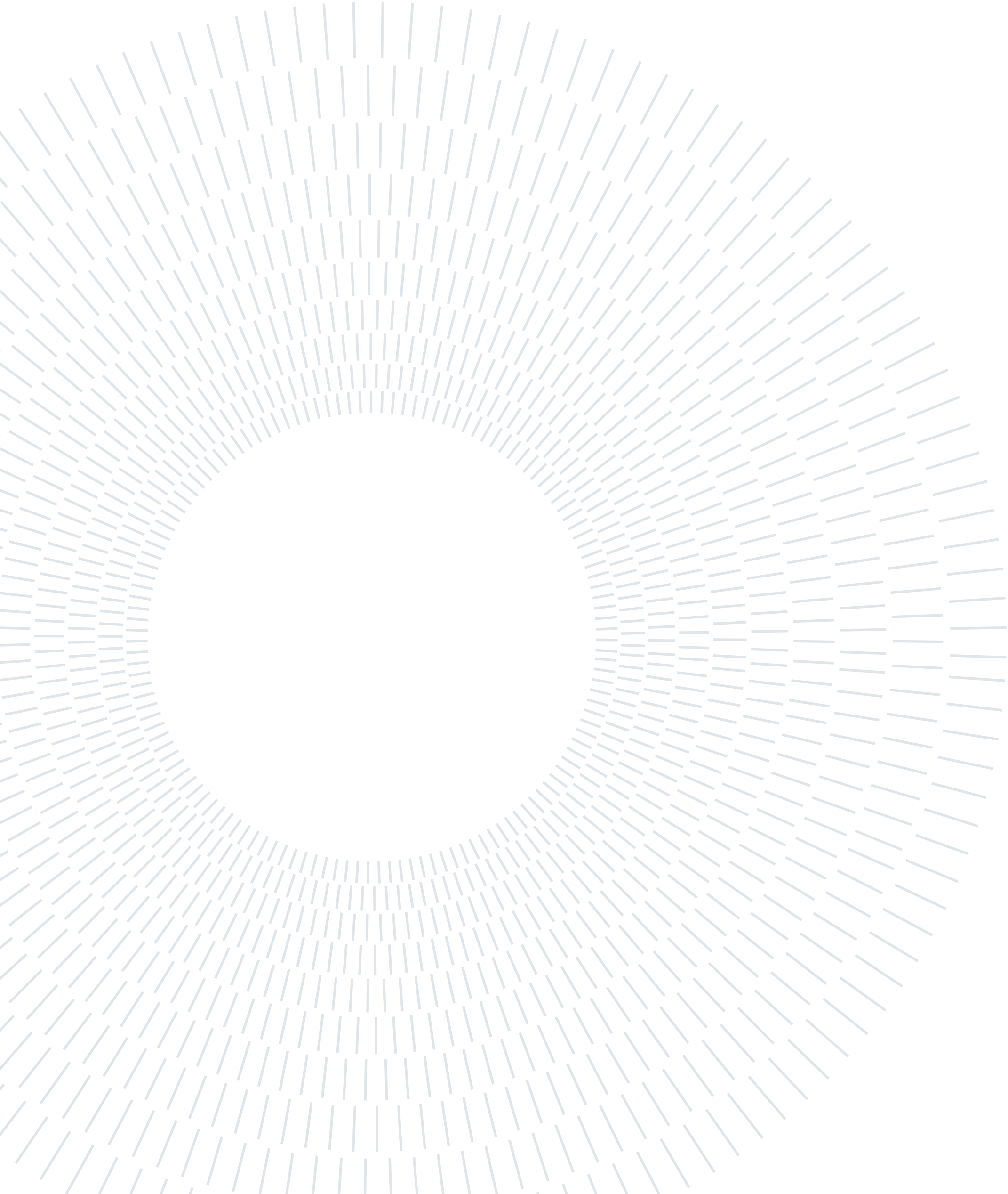
Single binary nets performances. The ensemble networks in this work combine the result of networks that have specific targets, therefore the result is strongly dependent (and limited) by the quality of the networks on the single organ. We can therefore act individually on a binary network to improve the metrics of the complete ensemble network of all the organs, for example by trying to combine architectures or different training methods.

Better Loss functions. A fundamental problem of semantic segmentation is the choice of the loss function for the training phase. The problem exists mainly because, in unbalanced datasets, a wrong loss function can lead to optimizing the network in order to penalize a small and not very present organ compared to larger ones with a greater volume. For the ensemble models in this work CrossEntropy Loss was used, that is a loss widely seen in literature. Despite the study of the weights for loss done in Appendix B, we could not find suitable weights. We therefore think that finding the right loss, or even the right organ weights, is a good way to further improve the performance of the proposed models.

Hyperparameters tuning. We mostly used parameters widely used in the literature and online challenges. We believe that the capabilities of these networks can be increased with a better parameters tuning.

Different fusion points. Regards the network with Features Fusion, it could be interesting to study the different performances and behaviors on ensemble networks that use features taken from different layers.

Multiple datasets comparison In this work we have based ourselves on the study of the behavior of architectures with respect to a single dataset (StructSeg), and with support of a different dataset (SegTHOR). In the future it might be useful to be able to extend the study with respect to other datasets, even from different source, to generalize the conclusions. It would also be important to study the performance with respect to organs other than those of the chest.



Bibliography

[1] 1x1 convolution: Demystified. URL `https://towardsdatascience.com/1x1-convolution-5219bbc09027`.

[2] Deeplabv3 implementation. URL `https://pypi.org/project/segmentation-models-pytorch/`.

[3] Computed tomography overview. *Winship Cancer Institute of Emory University*, . URL `https://www.cancerquest.org/patients/detection-and-diagnosis/computed-axial-tomography-ct-or-cat-scan`.

[4] Computed tomography. *National Institute of Biomedical Imaging and Bioengineering*, . URL `https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct`.

[5] nnunet implementation. URL `https://github.com/MIC-DKFZ/nnunet`.

[6] Se-resunet implementation. URL `https://github.com/zjuybh/StructSeg2019`.

[7] Review: Spatial pyramid pooling. URL `https://medium.com/analytics-vidhya/review-spatial-pyramid-pooling-1406-4729-bfc142988dd2`.

[8] Deep learning: Image segmentation and localization — u-net architecture. . URL `Https://medium.com/analytics-vidhya/deep-learning-image-segmentation-and-localization-u-net-architecture-ea4cff5595`

[9] unet implementation. . URL `https://github.com/milesial/Pytorch-UNet`.

[10] Artificial intelligence. URL `https://en.wikipedia.org/wiki/Artificial_intelligence_in_healthcare`.

[11] Window width and window level. URL `https://myctregistryreview.com/courses/my-ct-registry-review-demo/lessons/ct-physics/topic/window-width-and-window-level/`.

[12] Segthor: Segmentation of thoracic organs at risk in ct images. 2019. URL `https://competitions.codalab.org/competitions/21145#learn_the_details`.

[13] Automatic structure segmentation for radiotherapy planning challenge 2019. 2019. URL `https://structseg2019.grand-challenge.org/Home/`.

[14] 5oss B. Girshick. Fast r-cnn. *IEEE International Conference on Computer Vision*, 2013.

[15] R. V. Amisha PM, Pathania M. Overview of artificial intelligence in medicine. 2019.

[16] J. M. A.N. Ramesh, C. Kambhampati. Artificial intelligence in medicine. *Ann R Coll Surg Engl*, 2004. URL `https://www.scopus.com/record/display.uri?eid=2-s2.0-85104680660&origin=inward&txGid=379ac8ddc13402961c479f923ea77352`.

[17] Y. L. B. Wang. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. 2016.

[18] C. B. Bernhard Preim. Acquisition of medical image data. *Visual Computing for Medicine (Second Edition)*, 2014. URL `https://doi.org/10.1016/B978-0-12-415873-3.00002-X`.

[19] Z. Cao, B. Yu, B. Lei, H. Yinga, X. Zhang, D. Z. Chen, and J. Wu. Cascaded se-resunet for segmentation of thoracic organs at risk. 2020.

[20] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. 2017.

[21] A. G. Dan C. Cireşan. Deep neural networks segment neuronal membranes in electron microscopy images. 2012.

[22] X. Dong, Y. Lei, T. Wang, M. Thomas, and et al. Automatic multiorgan segmentation in thorax ct images using u-net-gan. 2019.

[23] A. B. et. al. Albumentations: fast and flexible image augmentations. 2018.

[24] M. G. et al. Ensemble deep learning: A review. 2021.

[25] N. N. Fausto Milletari. V-net: Fully convolutional neural networks for volumetric medical image segmentation. 2016.

[26] M. Ganaie, M. Hu, M. Tanveer, and P. Suganthan. Ensemble deep learning: A review. 2021.

[27] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2013.

[28] M. R. O. H. C. Shin. Stacked autoencoders for unsupervised feature learning

and multiple organ detection in a pilot study using 4d patient data. 2013. URL `<GotoISI>://WOS:000320381400009`.

[29] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. 2018.

[30] M. M. Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial nets.

[31] M. C. Jamie Shotton, Andrew Fitzgibbon. Real-time human pose recognition in parts from single depth images. 2011.

[32] R. C. Jamie Shotton, Matthew Johnson. Semantic texton forests for image categorization and segmentation. 2008.

[33] T. D. Jonathan Long, Evan Shelhamer. Fully convolutional networks for semantic segmentation. 2015.

[34] Lei, J. Harms, T. Wang, Y. Liu, H. K. Shu, A. B. Jani, W. J. Curran, H. Mao, T. Liu, and X. Yang. Mri-only based synthetic ct generation using dense cycle consistent generative adversarial networks. *Med Phys*, 2019.

[35] C.-J. Lin, C.-H. Lin, and S.-Y. Jeng. Using feature fusion and parameter optimization of dual-input convolutional neural network for face gender recognition. 2020.

[36] 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.

[37] J. McCarthy. What is artificial intelligence? 1998. URL `http://jmc.stanford.edu/articles/whatisai/whatisai.pdf`.

[38] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. 1943.

[39] P. F. O. Ronneberger. U-net: Convolutional networks for biomedical image segmentation. 2015.

[40] R. H. P. Aljabara. Multi-atlas based segmentation of brain images: Atlas selection and its effect on accuracy. 2009.

[41] M. E. A. E. Patrick Ferdinand Christ. Automatic liver and lesion segmentation in ct using cascaded fully convolutional neural networks and 3d conditional random fields.

[42] S. Ren, K. He, R. B. G. adn, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 2015.

[43] L. N. Smith. Cyclical learning rates for training neural networks. 2017.

[44] F. Steimann. On the use and usefulness of fuzzy sets in medical ai. 2001.

[45] P. VL. The coming of age of artificial intelligence in medicine. 2009.

[46] D. H. Wolpert. Stacked generalization. 1992.

[47] L. B. Y. Lecun. Gradient-based learning applied to document recognition. 1998.

[48] Y. F. Yang Lei. Deep learning in multi-organ segmentation. 2020.

[49] T. Zhou, S. Ruan, and S. Canu. A review: Deep learning for medical image segmentation using multi-modality fusion. 2019.

[50] Y. Y. Zhou, Y. Wang, S. B. P. Tang, W. Shen, E. K. Fishman, and A. Yuille. Semi-supervised 3d abdominal multi-organ segmentation via deep multi-planar co-training. *Conference on Applications of Computer Vision (Wacv)*, 2019.

A | CrossEntropy Loss Tuning

In this appendix we show the tests performed on the loss of the ensemble models in order to find the right loss for a good balance of the organs. In very unbalanced datasets, such as the case of multi-organ medical images, we often have to deal with targets of different sizes and dimensions. When there are small organs (such as trachea or esophagus) and large organs (such as lungs and heart) in the same image there is a risk that small objects are penalized during the train phase.

To improve this situation, an effective method is to work on the loss: a different function, or even the same loss but with different weights, can help a lot [36].

The experiments and results related to different weights per organ used in the CrossEntropy Loss function are now shown in Tables A.2, A.1, A.3, A.4. The networks used were standard Unet and Feature Fusion ensemble (FF).

	Weights	$Dice_{Unet}$	$Dice_{FF}$
RightLung	1	0.9597 ± 0.012	0.9681 ± 0.005
LeftLung	1	0.9656 ± 0.009	0.9724 ± 0.004
Heart	1	0.8937 ± 0.064	0.9216 ± 0.05
Trachea	1	0.5379 ± 0.113	0.7436 ± 0.082
Esophagus	1	0.7812 ± 0.056	0.8098 ± 0.037
SpinalCord	1	0.8054 ± 0.064	0.8914 ± 0.019

Table A.1: Standard CE - Performances on test set

	Weights	$Dice_{Unet}$	$Dice_{FF}$
RightLung	1	0.965 ± 0.005	0.968 ± 0.005
LeftLung	1	0.9674 ± 0.007	0.9718 ± 0.005
Heart	1	0.8642 ± 0.07	0.9222 ± 0.049
Trachea	2	0.467 ± 0.086	0.7427 ± 0.077
Esophagus	2	0.6733 ± 0.076	0.802 ± 0.047
SpinalCord	2	0.8443 ± 0.038	0.8885 ± 0.029

Table A.2: Double the weights on small organs - Performances on test set

	Weights	$Dice_{Unet}$	$Dice_{FF}$
RightLung	1	0.9596 ± 0.008	0.9628 ± 0.007
LeftLung	1	0.9613 ± 0.008	0.9653 ± 0.009
Heart	1	0.7646 ± 0.092	0.9232 ± 0.054
Trachea	15	0.4264 ± 0.095	0.6143 ± 0.065
Esophagus	5	0.6092 ± 0.099	0.7477 ± 0.07
SpinalCord	10	0.8079 ± 0.054	0.8599 ± 0.046

Table A.3: CE weights from [19] - Performances on test set

	Weights	$Dice_{Unet}$	$Dice_{FF}$
RightLung	1.2	0.9296 ± 0.021	0.935 ± 0.018
LeftLung	0.92	0.9369 ± 0.016	0.9528 ± 0.013
Heart	2.51	0.7915 ± 0.081	0.8477 ± 0.07
Trachea	30	0.24 ± 0.052	0.459 ± 0.058
Esophagus	48	0.5525 ± 0.084	0.4244 ± 0.077
SpinalCord	21	0.5512 ± 0.093	0.6514 ± 0.06

Table A.4: Inverse of the percentage of pixels - Performances on test set

B | 1x1 convolution weights

In this appendix we put the additional results of networks using a 1x1 convolution of the output of binary networks. As mentioned in Section 4.2.2 is possible to attribute the importance given to the single networks in the final segmentation by taking the weights of the convolution.

Each network is made up of a number of learnable parameters equal to:

$$N_{parameters} = N_{target} * N_{input} + N_{bias}$$

which represent the number of target organs, the number of input networks and a number of biases equal to the number of targets.

The different results are shown in tables B.1, B.3, B.2, B.4, B.6, B.5, B.8, B.9 according to the experiment shown in Section 5.

Every organ is represented with its ID number: [0, 1, 2, 3, 4, 5, 6] for [BackGround, RighLung, LeftLung, Heart, Trachea, Esophagus, SpinalCord].

T1

The weights are evenly distributed, as expected: no important networks for the background and strictly positive for each network associated with the target.

	1	2	3	4	5	6	bias
0	-5.16	-5.33	-5.14	-4.62	-4.4	-4.86	3.71
1	4.96	-2.36	-2.46	-2.73	-2.09	-2.87	-2.91
2	-2.62	4.87	-2.67	-2.77	-1.37	-3.02	-2.94
3	-2.37	-2.65	4.89	-2.65	-2.81	-2.91	-3.25
4	-2.44	-2.33	-1.99	4.0	-3.13	-2.99	-3.29
5	-1.68	-1.86	-2.24	-2.83	4.57	-2.56	-3.79
6	-2.33	-2.29	-2.1	-2.88	-2.52	4.56	-3.21

Table B.1: T1 - 1x1 convolution ensemble weights

T2

The most interesting thing to note is the almost equal distribution of the values of the input networks with respect to the corresponding targets: intuitively, for the esophagus and trachea the importance is divided between the two networks.

	1	2	3	4	5	6	4_a	bias
0	-5.57	-5.71	-5.52	-3.41	-4.45	-5.17	-2.28	3.61
1	4.67	-2.53	-2.38	-2.23	-1.83	-3.12	-1.8	-3.16
2	-2.73	4.45	-2.39	-1.95	-1.27	-3.4	-1.62	-2.87
3	-2.91	-2.23	4.29	-2.28	-2.45	-2.91	-1.83	-3.12
4	-2.33	-2.4	-2.11	2.75	-2.96	-2.95	2.1	-3.23
5	-2.41	-2.06	-2.78	-2.76	4.32	-3.08	-1.25	-3.67
6	-2.8	-2.72	-2.47	-2.35	-3.04	4.7	-1.84	-3.77

Table B.2: T2 (trachea) - 1x1 convolution ensemble weights

	1	2	3	4	5	6	5_a	bias
0	-6.22	-5.87	-5.25	-4.1	-2.03	-5.45	-2.4	3.48
1	4.24	-2.68	-2.39	-2.82	-0.26	-3.17	-1.6	-3.42
2	-2.96	4.34	-2.05	-2.31	0.55	-3.81	-2.07	-3.11
3	-2.64	-2.55	4.54	-2.34	-1.62	-3.07	-1.66	-3.23
4	-3.03	-2.56	-2.33	4.33	-2.61	-3.13	-1.8	-3.32
5	-2.83	-2.61	-2.52	-2.55	1.98	-2.97	2.68	-3.28
6	-2.74	-2.2	-2.23	-2.32	-1.37	4.41	-1.32	-3.85

Table B.3: T2 (esophagus) - 1x1 convolution ensemble weights

	1	2	3	4	5	6	4_a	5_a	bias
0	-5.38	-5.84	-5.44	-3.56	-2.23	-5.34	-2.37	-2.84	3.67
1	4.74	-2.84	-2.25	-2.42	-0.6	-3.08	-1.96	-2.31	-2.96
2	-2.49	4.33	-2.68	-2.27	-0.05	-3.51	-1.63	-2.13	-2.91
3	-2.33	-2.76	4.32	-2.09	-1.51	-3.5	-1.58	-1.47	-2.99
4	-2.55	-2.22	-1.99	3.06	-2.02	-2.86	2.07	-1.98	-3.64
5	-2.05	-2.52	-2.32	-2.28	2.17	-2.86	-1.46	2.38	-3.56
6	-2.6	-2.4	-2.6	-2.51	-2.13	4.54	-2.08	-1.65	-3.69

Table B.4: T2 (esophagus + trachea) - 1x1 convolution ensemble weights

T3

Looking at the trained network on the trachea, we see that there is no positive weight corresponding to the input. Unexpectedly it is the esophagus (trained on the same test database) that has positive values on both outputs. The situation is reversed for the esophagus.

	1	2	3	4	5	6	bias
0	-5.53	-6.05	-5.48	-1.47	-4.28	-5.16	3.57
1	4.51	-2.87	-2.67	-1.77	-2.13	-3.23	-3.03
2	-2.82	4.0	-2.36	-1.28	-1.53	-3.26	-2.92
3	-2.72	-2.92	4.25	-1.21	-2.15	-3.45	-3.13
4	-2.67	-2.83	-2.13	-0.52	-1.66	-2.86	-3.2
5	-1.96	-2.57	-2.66	1.21	3.5	-3.09	-3.59
6	-2.4	-2.2	-2.09	-1.02	-2.08	4.3	-3.41

Table B.5: T3 (trachea) - 1x1 convolution ensemble weights

	1	2	3	4	5	6	bias
0	-5.61	-5.93	-5.51	-4.09	-0.95	-5.09	3.24
1	4.35	-2.45	-2.22	-2.66	-1.35	-3.19	-3.25
2	-2.41	4.38	-3.01	-2.1	-1.17	-3.6	-3.44
3	-2.74	-2.83	4.24	-2.24	-0.27	-2.78	-3.38
4	-2.54	-2.32	-2.13	3.53	1.15	-3.15	-3.64
5	-2.21	-2.39	-2.18	-1.19	1.23	-3.14	-3.25
6	-2.84	-2.92	-2.11	-2.25	-1.24	4.25	-3.6

Table B.6: T3 (esophagus) - 1x1 convolution ensemble weights

T4

The positive weights are equally distributed among the right input/target pairs, except for the smaller organs for which it is evidently more useful to use the input from the single-organ network corresponding to the target.

	1	2	3	4	5	6
0	-2.55	-2.86	-2.85	-2.24	-1.84	-3.2
1	2.15	-1.48	-1.44	-1.31	-1.19	-0.86
2	-1.61	1.83	-1.2	-1.03	-0.55	-1.05
3	-1.66	-1.67	2.35	-2.09	-1.3	-1.16
4	-1.28	-1.7	-1.48	2.36	-2.05	-1.18
5	-1.33	-1.56	-1.42	-2.0	1.84	-0.82
6	-1.44	-1.56	-1.03	-1.08	-1.44	2.54

Table B.7: T4 - 1x1 convolution ensemble weights from single nets

	0	1	2	3	4	5	6	bias
0	1.8	-2.66	-2.95	-2.7	-0.43	-1.65	0.34	2.1
1	-1.68	2.84	-1.84	-1.64	-1.08	-1.35	-1.3	-1.91
2	-2.05	-1.68	2.51	-1.41	-1.03	-1.04	-1.15	-1.56
3	-1.76	-1.87	-1.58	2.7	0.13	-1.36	-1.74	-2.04
4	-1.79	-1.42	-1.82	-1.43	4.2	-1.57	-1.35	-1.93
5	-2.17	-1.46	-1.39	-1.73	-0.26	3.71	-1.5	-1.76
6	-1.71	-1.77	-1.71	-1.64	-1.67	-1.54	4.47	-1.9

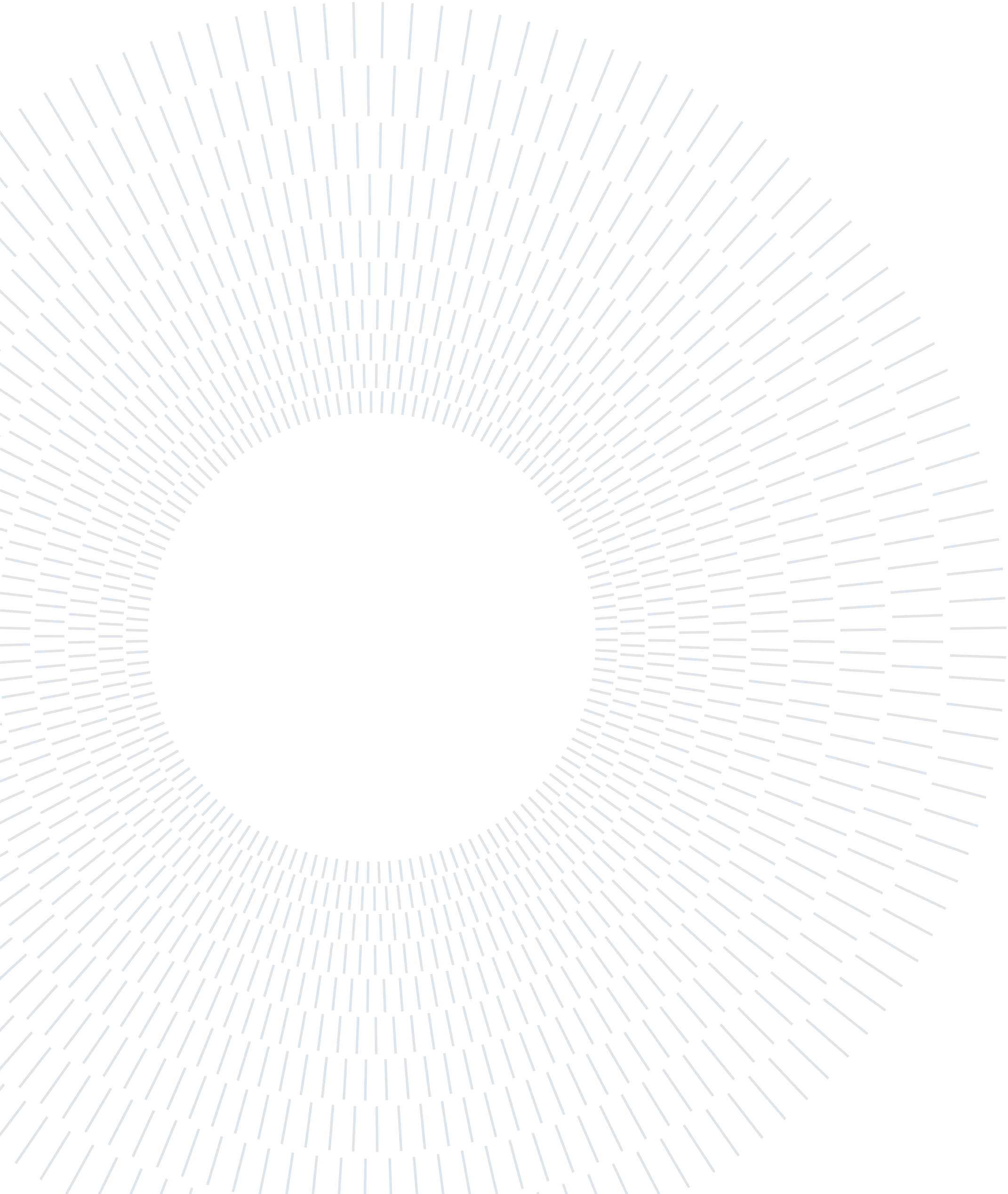
Table B.8: T4 - 1x1 convolution ensemble weights from multiclass

T5

The situation is very similar to the T1 experiment, only with a small reduction of all positive values.

	1	2	3	4	5	6	bias
0	-5.31	-5.68	-4.72	-3.86	-3.54	-4.81	3.34
1	4.06	-2.48	-2.42	-2.86	-1.34	-2.7	-2.92
2	-2.65	3.59	-2.52	-2.11	-0.8	-3.12	-2.81
3	-2.47	-3.05	3.92	-2.11	-1.92	-2.61	-2.97
4	-2.17	-2.57	-1.84	3.69	-2.23	-2.58	-2.66
5	-2.0	-2.26	-2.42	-2.07	3.89	-2.25	-2.67
6	-2.42	-2.74	-2.39	-2.06	-2.13	3.84	-2.65

Table B.9: T5 - 1x1 convolution ensemble weights

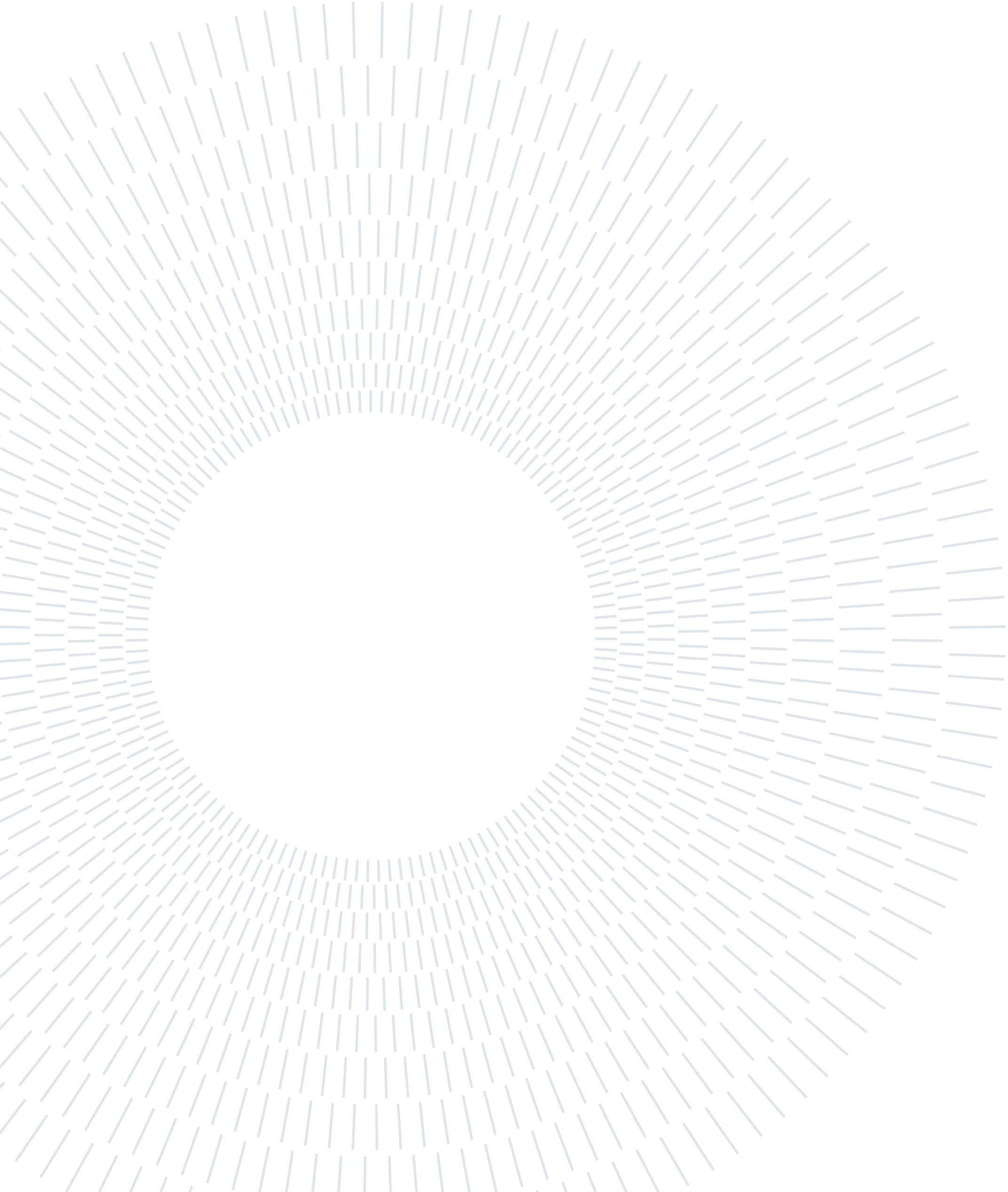


List of Figures

List of Tables

Acronyms

ANN - **A**rtificial **N**eural **N**etwork
AE - **A**uto **E**ncoder
CNN - **C**onvolutional **N**eural **N**etwork
FCNN - **F**ully **C**onvolutional **N**eural **N**etwork
CT - **C**omputer **T**omography
MRI - **M**agnetic **R**esonance **I**maging
R-CNN - **R**egional - **C**onvolutional **N**eural **N**etwork
ROI - **R**egion **O**f **I**nterest
OaR - **O**rgan **a**t **R**isk
DICOM - **D**icom **Imaging** and **C**ommunications in **M**edicine
HU - **H**ounsfield **U**nits



Ringraziamenti

Vorrei ringraziare prima di tutto il Prof. Daniele Loiacono, mio relatore in questo percorso, e il Dott. Leonardo Crespi per la loro grande disponibilitá ma anche per la pazienza mostrata, per i preziosi consigli e per i confronti costruttivi che mi hanno permesso di scrivere questo lavoro.

Grazie a mia mamma e mio papà, che non hanno mai smesso di incoraggiarmi e mi hanno supportato in ogni modo possibile durante questi anni permettendomi di concentrarmi sullo studio a tempo pieno. Tutto questo, senza di loro, non sarebbe stato possibile. Ringrazio mio fratello Gio, su cui ho sempre potuto fare affidamento. Un grazie speciale ai miei nonni Aldina, Peo e Anna, le persone che piú mi hanno ispirato a diventare quello che sono e che piú di chiunque altro aspettavano questo momento.

Ringrazio Alessia, che nonostante la distanza é sempre stata la prima a sostenermi e ad ascoltarmi in tutti i momenti, specialmente in quelli di sconforto. Spero di riuscire ad essere per lei quello che lei é stato per me.

Una menzione speciale per Tasca, con cui condivido infiniti progetti e pomeriggi di studio. Insieme allo storico gruppo Ste, Sonzo, Puti e Sacco avete reso molto più leggeri questi anni. Grazie a tutti i colleghi con cui ho condiviso un pezzo di questo percorso.

Ringrazio inoltre il gruppo del Rugby Monza, la mia squadra da una vita e valvola di sfogo nei momenti di maggiore stress. Lí ho trovato un gruppo che ritengo speciale e persone stupende che reputo tra i migliori amici.

Ringrazio infine amici e parenti che non ho menzionato per motivi di spazio, ugualmente importanti per me nel raggiungimento di questo risultato.