

École polytechnique de Louvain

Real-time tumor tracking for proton therapy using Deep Learning and X-Ray fluoroscopy

Author: **Edouard CHATZOPOULOS**
Supervisors: **Benoît MACQ, Estelle LOÏEN**
Reader: **John LEE**
Academic year 2021–2022
Master [120] in Computer Science and Engineering

Abstract

Cancer is a leading cause of death worldwide, accounting for nearly 10 million deaths in 2020, or nearly one in six deaths. Among those, lung cancer is the biggest killer, causing more than 1.8 million deaths in the same year [1]. Radiotherapy is one of the most common treatment methods for tumors, it uses ionizing radiation to destroy cancerous cells. Radiotherapy can take a multitude of forms, the most frequent one being external radiotherapy using photons as the radiation delivering mechanism. In recent years, proton therapy, another form of external radiotherapy has been steadily gaining traction. It uses protons as the radiation delivery medium, and as a result, can be more precise in its treatment thanks to a particularity of its radiation dose deposition curve named the Bragg peak. Because of this peak, the radiation can be deposited locally at the position of the tumor. However, for tumors located on moving organs such as the lungs, the position of the tumor can shift during the treatment, reducing the treatment's accuracy and requiring large security margins to cover the range of movement. In this work, we develop and analyze the performance of a real-time tumor segmentation model based on deep learning. This new approach consists of detecting the exact position of the tumor with a convolutional neural network to adjust the target of the treatment based on X-ray fluoroscopic imaging. If this approach proves successful, it would remove the need for large security margins and lead to a more accurate treatment with less compromise.

Acknowledgements

The completion of this thesis would not have been possible without the advice and support of many people, I would like to express my sincere gratitude to all of them.

Foremost, I would like to thank my supervisor, Professor Benoît Macq, for agreeing to assign me this topic, and for overseeing my work through the different stages of this thesis. Thanks to him, I was able to contribute to a field of research that inspires me and work with technologies that have long fascinated me.

I would also like to thank profusely Estelle Loÿen for her priceless advice, her amazing help, and her availability and responsiveness throughout the year. This work would not have been possible without her support.

Then I would like to thank Damien Dasnoy-Sumell, for his help and support for this work, as well as for his technical help with the servers.

I thank as well Professor John Lee for agreeing to read this work.

I would also like to thank Dr. Pierre Castadot, of the radiology and oncology department of the Chirec hospital group, for taking the time to answer my questions about radiotherapy in the early stages of this work.

Then I would like to acknowledge with gratitude the support of my family. In particular my parents, for their advice on the medical aspect of this work. Finally, I would like to thank Audrey for her constant support and encouragement.

Contents

Abstract	i
Acknowledgments	ii
Acronyms	vi
List of Figures	vii
List of Tables	xiii
1 Introduction	1
1.1 Basic principles of radiotherapy	1
1.1.1 Physic and biological principles of radiotherapy	1
1.1.2 The biological effects of radiation therapy	1
1.1.3 Delivering Methods	2
1.2 Proton therapy	2
1.2.1 Motivations for Proton therapy	2
1.2.2 Proton therapy on moving tumors	4
1.2.3 Other challenges and limitations associated with Proton therapy	6
1.3 Objectives of this work	7
1.4 Structure of this document	7
1.5 Summary	7
2 Deep Learning in medical imaging	8
2.1 The breakthrough of deep learning in radiology	8
2.2 A brief introduction to machine learning	8
2.3 The multilayer perceptron	10
2.3.1 The perceptron	10
2.3.2 Building an MLP from perceptrons	11
2.4 Convolutional neural networks	12
2.4.1 Biological inspiration of CNNs	12
2.4.2 Building blocks of convolutional neural networks	13

2.4.3	Convolution	13
2.4.4	Pooling Layer	15
2.4.5	Non-linear activation function	17
2.4.6	Fully Connected layers	18
2.4.7	Final Layer activation function	19
2.5	Training a neural network	20
2.5.1	Loss function	20
2.5.2	Gradient descent	21
2.5.3	Overfitting	22
2.5.4	Evaluation	22
2.6	U-net	25
2.6.1	Encoder-Decoder inspirations	26
2.6.2	The skip connections	26
2.7	Deep learning in radiology and medical images segmentation	27
2.7.1	Motivations	27
2.7.2	Challenges	28
2.7.3	Pathology detection	30
2.7.4	Segmentation of medical images	31
2.8	Summary	35
3	Generation of the training data	36
3.1	Synthetic data	36
3.2	Simulating fluoroscopy from 3D Computed Tomography scans	38
3.3	Data augmentation tools	40
3.3.1	OpenTPS data augmentation	41
3.3.2	Random noise	42
3.3.3	Random crop and rotate	43
3.4	Composition of the dataset	43
3.5	Summary	44
4	Real-time tumor detection	45
4.1	The Neural Network	45
4.1.1	Topology of the network	45
4.2	Loss function	49
4.2.1	Binary Cross-Entropy loss	50
4.2.2	Binary Cross-Entropy loss with cropped output	50
4.2.3	Binary Dice Loss	51
4.2.4	Distance Penalized Dice-Loss	51
4.2.5	Focal Loss	53
4.2.6	Binary Hausdorff loss	54
4.3	Methodology	54

4.3.1	Derivation of the optimal parameters	54
4.3.2	Loss function comparison	55
4.3.3	Robustness evaluation	56
4.3.4	Comparison against static treatment planning	59
4.3.5	Inference time evaluation	60
4.4	Summary	60
5	Results and discussion	61
5.1	Synthetic data in 128x128 resolution	61
5.2	Loss functions comparison	62
5.3	Robustness evaluation	67
5.3.1	Robustness to noise	67
5.3.2	Robustness to deformations	70
5.3.3	Robustness to rotations	71
5.4	Comparison against a static treatment planning	72
5.5	Inference time evaluation	74
5.6	Summary	74
6	Conclusion	76
A	Additional evaluation metrics	90
A.1	Robustness to noise	90

Acronyms

ANN Artificial Neural Network

BCE Binary Cross-entropy

BN Batch Normalization

DNA DeoxyriboNucleic Acid

CNN Convolutional Neural Network

CT Computerized Tomography

GS Grid Search

HD Hausdorff Distance

MidP MID Position

MR Magnetic Resonance

MRI Magnetic Resonance Imaging

MRI-Linac Magnetic Resonance Imaging Guided Linear Accelerator

OpenTPS OPEN Treatment Planning System

ReLU REctified Linear Unit

ROI Region Of Interest

List of Figures

1.1	Percentage depth-dose distribution curves of proton beam vs photon beam [8]. The dose deposited by the photon beam increases rapidly before a slow decrease across the depth, whereas the dose deposited by the proton beam is small in the beginning, peaks and then falls abruptly to a negligible amount after a specific depth in the tissue.	3
1.2	Bragg peak for different energies. The deposition of the dose in depth and the maximum of the Bragg peak vary according to the energy of the proton. [9].	4
2.1	Example of a simple perceptron with 3 input values. The weighted sum of the input values is passed through the threshold function to produce a single binary output value.	10
2.2	Illustration of a simple multilayer perceptron with 3 hidden layers. This type of network is fully connected, every output of each layer is connected to every input of the next layer. The weights "w" of the connections in a hidden layer are independent trainable parameters learned by the network during the training phase.	11
2.3	Example of a simple CNN (LeNet5 digit recognition network [28]). The channels of each layer are represented by stacked rectangles, each rectangle in the stack represents one channel.	13
2.4	Illustration of a small convolutional layer. The kernel is superposed with a part of the input layer and each element of the input is multiplied with the corresponding element of the kernel, the result of these products is then summed to produce one output element. The Kernel is attractively shifted and the process is repeated to compute the whole output array. Illustrations inspired by [29]	14
2.5	Illustration of a convolutional layer with zero-padding. The input layer is extend with a border consisting of zeros in order to produce an output of equal size as the input.	15

2.6	Illustration of a max pooling layer with a 2x2 pooling size. The input is divided into patches of equal size and the greatest value in each patch is selected as the output value representing the whole patch.	16
2.7	Illustration of a average pooling layer with a 2x2 pooling size. The input is divided into patches of equal size and the average of the values of each patch produces the output value representing the whole patch.	16
2.8	Illustration of a global average pooling layer. For each channel, the average of the values in the input produces a single output value representing the whole channel.	17
2.9	Response curve of the hyperbolic tangeant and Sigmoid activation functions.	18
2.10	Illustration of the response curve of the ReLU activation function [37]	19
2.11	Intuitive illustration of the effect of different learning rates on the loss for gradient descent with a single weight w	21
2.12	Illustrations of the two most common types of cross-validation. Kfold cross-validation divides the dataset in K subsets while leave-one-out divides the dataset in N subsets (with N being the number of samples in the dataset). At each iteration, one subset is used as validation data while the rest is used as training data.	25
2.13	Original U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations [55].	27
2.14	Number of publications for the search phrases with at least the terms radiotherapy or radiation therapy or radiation oncology and at least the terms deep learning or deep network or convolutional network [56].	28
2.15	One example of the visualization technique proposed by Zeiler and Fergus. Starting from the left, the first image shows how different portions of the scene are systematically covered up with a gray square. The second image shows the activation of the feature map of a top layer. The third image shows the visualization of the features of the same layer. The fourth image represents the localization map of the spatial zones of the images used by the classifier to identify the dog. The last image shows a map of the correct class as a function of the position of the grey square [63].	30

2.16	Overall workflow of the proposed tumor-contouring method of Toshiyuki Terunuma et al. [94]	34
3.1	Different steps of the generation of a simulated fluoroscopic image and its associated tumor segmentation mask.	37
3.2	Simple projection of the 3D CT onto a 2D plane with the python library Tomopy. The resulting images are stretched because of the the uneven shape of the 3D CT images.	39
3.3	Simulated fluoroscopic images of two patients, reshaped to a standardized 256x256 pixels resolution ((a), (b)). Figure (c) and Figure (d) present the same images with the contour of the tumor highlighted in red. The resizing is done with a bi-linear interpolation for the images and nearest-neighbor interpolation for the segmentation masks.	40
3.4	Generation of a new phase of the respiration cycle with amplitude modulation in OpenTPS. The blue circles represent the existing phases of the 4D CT, the red circle represents the new phase before amplitude modulation and the green circle represents the new phase after amplitude modulation.	42
4.1	Visual representation of the topology of the U-net used in this work. This network is composed of 4 contract blocks, 4 expand blocks and 3 skip connections.	46
4.2	Illustration of the first two steps of a transposed convolution with a stride of one and a padding of two.	47
4.3	Alternative illustration of the first step of a transposed convolution with a stride of one and a zero-padding of two.	48
4.4	Example of a transposed convolution with a stride of 2, a padding of 1 and an output padding of 1. The output array is twice the size of the input array in each dimension.	48
4.5	Response curve of the sigmoid function [98].	49
4.6	Comparison between the segmentation target and the predicted mask with a network trained with BCE loss. Artifacts appears on the bottom edge of the prediction.	51
4.7	Binary segmentation mask of the tumor of Patient 1 (a) and the associated error-penalizing distance map (b). The darkest parts of the distance map have a pixel value of 1 and the lightest parts have a pixel value of 2.	52

4.8	A Fluoroscopic image at the different levels Gaussian noise tested. Each level of Gaussian noise has a standard deviation which corresponds to a specified percentage of the maximum intensity of the image.	57
4.9	Example of a fluoroscopic image before and after rotation. The rotated image has been re-framed and resized with a bi-linear interpolation in order to fit the initial resolution without black borders caused by the rotation.	59
5.1	Distribution of the Jaccard loss of each fold for every loss function. The yellow bars represent the median values and the green triangles correspond to the average values	62
5.2	Visual representation of the precision and recall of a segmentation mask generated with the network trained with focal loss on patient 1. The intersection between the real mask and the predicted mask is displayed in blue, the pixels of the predicted mask not present on the real mask are shown in green and the pixels present on the real mask not predicted by the network are shown in red. As with the majority of the samples, there are no red pixels as the network has an almost perfect recall.	65
5.3	Evolution of the average Jaccard index of the segmentation for different levels of Gaussian noise in the validation dataset. The network has not been trained on noisy data.	68
5.4	Evolution of the average Jaccard index of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.	69
5.5	Increase in percent of the proposed method in terms Dice index, precision, recall and variance compared to a static mask inspired by static treatment planning (plotted with a symmetric logarithmic vertical scale).	73

List of Tables

4.1	Hyper-parameters of the network used for each loss function.	55
5.1	Dice index, Jaccard index, precision and recall of the proposed network trained on 128x128 simulated fluoroscopic images on a 5-fold cross validation.	61
5.2	Patient 1 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average precision and recall metric of every loss.	65
5.3	Patient 2 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.	66
5.4	Patient 3 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.	66
5.5	Patient 4 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.	66
5.6	Patient 5 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.	66
5.7	Patient 1 : Average Dice index, average Jaccard index, average Precision and Recall metric on evaluation sets following a different deformation amplitude distribution range.	71
5.8	Patient 4 : Average Dice index, average Jaccard index, average Precision and Recall metric on evaluation sets following a different deformation amplitude distribution range.	71
5.9	Patient 1 : Average Dice index, average Jaccard index, variance of the Jaccard index, average Precision and Recall metric of the same network evaluated on one set without rotations and one set augmented with random image rotations uniformly distributed between -5 and 5 degrees.	72

5.10	Patient 4: Average Dice index, average Jaccard index, variance of the Jaccard index, average Precision and Recall metric of the same network evaluated on one set without rotations and one set augmented with random image rotations uniformly distributed between -5 and 5 degrees.	72
5.11	Patient 1 : Comparison between the segmentation of the proposed method with a static mask inspired by treatment planning	73
5.12	Patient 4: Comparison between the segmentation of the proposed method with a static mask inspired by treatment planning	73
5.13	Comparison between the segmentation of the proposed method with a static mask inspired by treatment planning (Patient 5)	73
5.14	Metrics and quantiles of the distribution of the end-to-end latency of the network for a single 256x256 image over 100 repetitions.	74
A.1	Patient 1 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.	90
A.2	Patient 2 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.	90
A.3	Patient 3 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.	91
A.4	Patient 4 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.	91
A.5	Patient 5 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.	91
A.6	Patient 1 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.	91

A.7	Patient 2 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.	92
A.8	Patient 3 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.	92
A.9	Patient 4 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.	92
A.10	Patient 5 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.	93

Chapter 1

Introduction

1.1 Basic principles of radiotherapy

Radiotherapy is one of the most common treatment methods for tumors. It consists in targeting cancerous cells with ionizing radiations to destroy them while minimizing damage to the surrounding tissues. Radiotherapy can be used on its own or in combination with other treatments in order to eradicate a tumor.

1.1.1 Physic and biological principles of radiotherapy

Radiation therapy is a treatment of cancer that uses ionizing radiation. The goal of radiation therapy is to destroy or inactivate the cancerous cells while maintaining the integrity of adjoining normal cells.

Irradiating cells with high-energy particles or waves causes ionization within them. Ionization is the process by which a neutral atom acquires a positive or negative charge. Some charged particles such as electrons, protons, and alpha particles have sufficient kinetic energy to produce ionization by collision as they penetrate matter. Uncharged particles such as neutrons also produce ionization when they interact with matter. It is also possible to use high-energy photons, such as x-rays or gamma rays, this is known as electromagnetic radiation. These photons can either be generated electrically by machines such as linear accelerators or produced through spontaneous emission by radioactive materials such as radioactive cobalt or cesium. [2].

1.1.2 The biological effects of radiation therapy

Exposure of cells to ionizing radiation results in immediate and widespread oxidative damage, especially to the Deoxyribonucleic acid (DNA) [3]. The damage to the

DNA results from the direct effect of the energy from the radiation, but also results from the interactions of the radiation with the cellular water, forming ion radicals which then attack it. The lesions produced to the DNA include single-strand breaks, base alterations, and double-strand breaks [4].

Cellular lethality seems to be related to the number of double-stranded breaks produced in the cell [5]. The sensitivity of tissues to radiotherapy seems to depend on their ability to repair radiation damage, repopulate and re-oxygenate. The capacity to repair double and single-stranded breaks varies between normal tissue and different tumors. The ability to repopulate is important, as cells can rapidly divide to replace cells killed by radiation.

1.1.3 Delivering Methods

Radiotherapy is divided into three main delivering methods; external radiotherapy, curietherapy, and therapy by radio-isotopes [6]. In external radiotherapy, a machine is used to aim beams of radiation at the tumor. Curietherapy, or brachytherapy, involves implanting small pieces of radioactive material either inside or near the tumor. Finally, in therapy by radio-isotopes, the patient either swallows a radioactive liquid or has it injected into his blood. A patient can be treated with one of these forms of radiotherapy or a combination of multiple delivery methods.

External radiotherapy is the main focus of this work, and among the various forms of sources of ionizing radiation used in this method, X-ray-based radiotherapy is the most widespread. However, another form of external radiotherapy based on charged particles has begun to be implemented in the field and promises numerous advantages compared to conventional photon-based radiotherapy. This form of radiotherapy is proton therapy, and the objective of this work is to provide software tools destined to further improve this type of treatment.

1.2 Proton therapy

1.2.1 Motivations for Proton therapy

Most of the current practice of clinical radiotherapy utilizes photon beams, ranging from 4 to 18 mega-volts [7]. As shown in Figure 1.1 the radiation dose delivered by X-ray radiotherapy as a function of the depth in the patient tissue quickly rises, before stagnating at a maximum and then slowly decreasing as photons are absorbed. Thus, a photon beam delivers the dose all the way across its path

through the patient's body. In order to deliver a high dose of radiation to the tumor while maintaining the radiations administered to the surrounding tissues below a relatively safe threshold, a crossfire of beams at multiple angles are therefore needed.

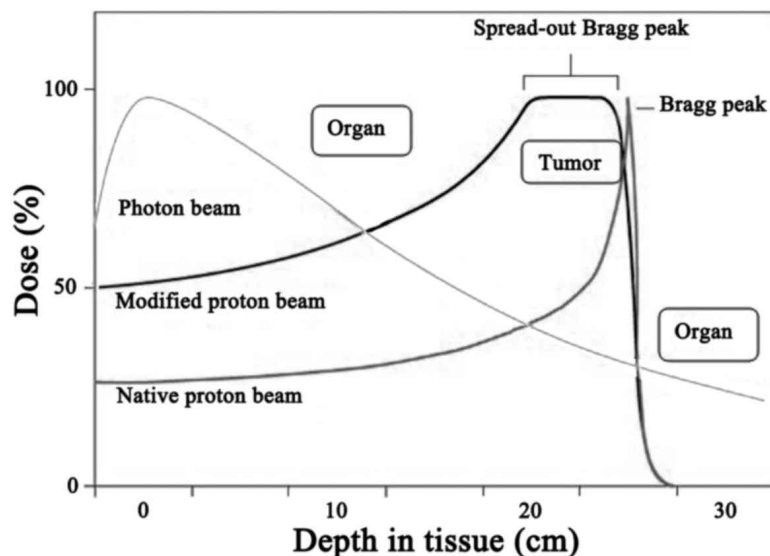


Figure 1.1: Percentage depth-dose distribution curves of proton beam vs photon beam [8]. The dose deposited by the photon beam increases rapidly before a slow decrease across the depth, whereas the dose deposited by the proton beam is small in the beginning, peaks and then falls abruptly to a negligible amount after a specific depth in the tissue.

In contrast with photons, protons slow down as they penetrate matter. Furthermore, the rate of their energy transfer increases as their velocity decreases, before coming to an abrupt stop when their entire energy is depleted. As a result of these phenomenons, the dose deposition as a function of the depth in the patient presents a characteristic shape referred to as the "Bragg Curve" or "Bragg peak" as shown in Figure 1.1. The depth of the peak is a function of the initial energy of the emitted protons as shown in Figure 1.2.

Thanks to this particularity, by simply modulating the energy of the protons, it is possible to focus most of the dose delivery to a specific depth. By measuring the precise location of the tumor it is possible to compute the optimal energy of the photon beam in order to deliver the majority of radiation to the tumor. The tissue situated before this targeted depth will receive a moderate dose of radiation, the tumor will receive a maximum amount of radiation, and the dose deposited beyond it will be negligible. As protons traverse the body, they also scatter laterally, but

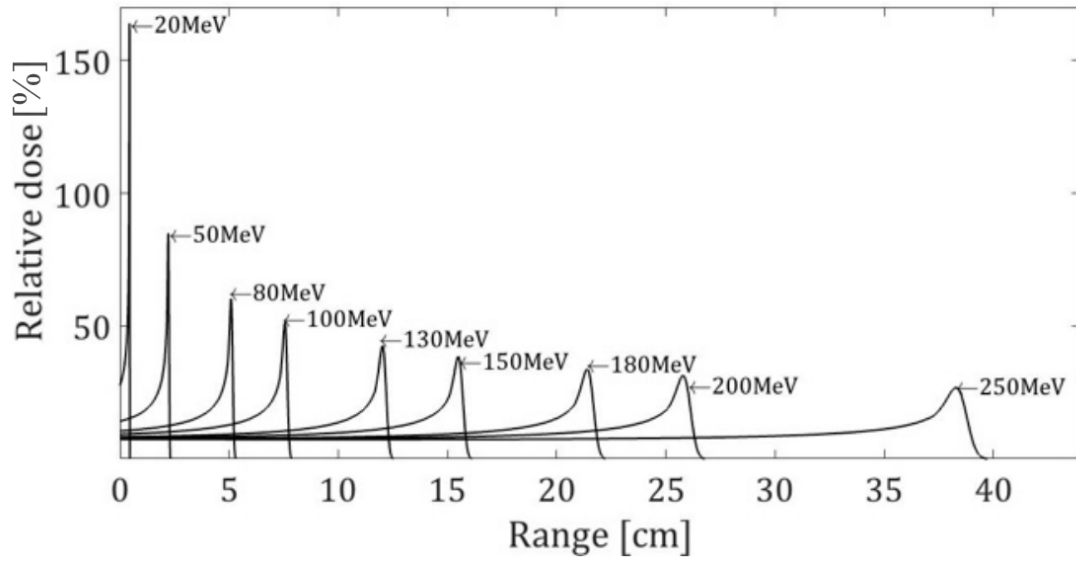


Figure 1.2: Bragg peak for different energies. The deposition of the dose in depth and the maximum of the Bragg peak vary according to the energy of the proton. [9].

the dose outside of the beam falls rapidly [7].

1.2.2 Proton therapy on moving tumors

Certain types of tumors, especially those located on organs affected by the respiratory cycle such as lung tumors or liver tumors, can shift in location during the treatment of the patient. This displacement can be particularly challenging but several strategies to mitigate motion-induced uncertainties or at least reduce their effect on treatment effectiveness and safety are currently being applied or researched [10]. Applicable strategies involve three main components: planning, patient movement, and timing-based beam delivery.

Robust planning optimization

The basic idea of robust planning optimization is that the optimizer takes into consideration the expected uncertainty in the planning stage by considering the worst-case scenario or all possible scenarios [11]. Based on these worst-case scenarios, security margins are computed to limit the dose to critical organs while

allowing a uniform dose distribution to the target. The radiation dose will therefore be delivered to the whole range of motion of the tumor.

Patient motion management

The patient's motion management based approach consists in either asking the patient to hold their breath, by blocking their airways or compressing their abdomen to reduce motion amplitude. In addition to reducing the range of motion of the tumor, deep inspiration breath-hold can also reduce the lung density and therefore reduce the normal lung volume in the high dose region of the treatment [12]. The biggest drawback of the patient's motion management based approach is that either the compliance of the patient is needed, or their respiration has to be forcefully restricted.

Timing based beam delivery strategies

Timing-based beam delivery strategies are an assortment of dynamic approaches to dose delivery. This approach can be divided into 3 main categories: Beam gating, rescanning, and tumor tracking [10].

Beam gating consists in monitoring the patient breathing and timing the dose delivery in order to only irradiate the tumor in a predetermined phase of the respiration cycle.

Rescanning consists in delivering one fraction of the dose in multiple repeated scans. Each scan of the rescanning procedure will either result in an underdose or an overdose, with the goal of obtaining a statistical averaging of the dose delivery.

The tumor tracking strategy, as its name implies, involves tracking the position of the tumor in real-time in order to dynamically adjust the target during the treatment. For proton therapy, this approach is still only at the stage of research and is not yet used clinically. However, if done correctly this approach would be the most advanced method for the mitigation of the effects of tumor motion. The reasons why this method is not currently implemented are the many challenges it represents. The first challenge is the low quality of the imaging available during proton therapy, which usually consists of low-power X-ray fluoroscopic images, where the tumor is often not visible. The second challenge is the timing constraint, as the latency of the system needs to be short enough to adjust the aiming of the proton beam before the tumor moves too much. The final challenge is to get the system to be reliable enough

to be trusted, as tracking errors could result in terrible consequences for the patient.

Despite these challenges, the fast development of the field of deep learning in computer vision, as well as the rapid gain in performance of neural networks due to GPU-accelerated computing are bringing real-time tumor tracking into the realm of possibilities. This method of mitigating motion-induced uncertainties for lung tumor treatment in proton therapy will be the main focus of this work.

1.2.3 Other challenges and limitations associated with Proton therapy

Despite the dosimetric advantages of proton therapy, the advantages still have to be translated to proven clinical benefits [13]. This discrepancy might be attributable to inadequate dose prescription, or current technical limitations that may lead to differences between the planned dose and the actual delivered radiation dose.

Moreover, because the technology of X-ray based radiotherapy is more mature, systems with advanced imaging capabilities such as MRI-Guided Linear Accelerators (MR-Linac) are starting to be implemented. The hybrid machines allow real-time MRI imaging during the treatment. Instead, proton therapy is currently limited to low-power fluoroscopic imaging during the treatment.

The biggest obstacle to the universal use of the proton in cancer treatment is the size and cost of the cyclotron or synchrotron equipment necessary. However, this obstacle is slowly reducing thanks to technological advancement and the maturation of the technology. While proton therapy facilities are still more expensive and larger than X-ray based radiotherapy equipment, this difference is getting smaller [14] and proton therapy is becoming a growing alternative to conventional radiotherapy [15]

Because of these factors, there are still only 103 proton therapy facilities worldwide as of May 2022 [16]. However, this number is steadily growing every year and as the technology becomes more mature, we could envision proton care as an equal partner to photon care for the next generation of radiation oncologists [17].

1.3 Objectives of this work

The main objective of this work is to develop a method for real-time tracking of tumors during proton therapy. This tracking uses a deep learning based segmentation of the 2D X-Ray fluoroscopic images available during the treatment.

Such tracking would allow to adjust the radiation beam in real-time in order to target the tumor more efficiently during the whole breathing cycle. This technique could therefore augment the efficiency of the treatment while minimizing collateral damages to adjacent organs and tissues, without the need to restrict the respiration of the patients. Moreover, this technique could be implemented with very little new equipment, as it only requires ordinary fluoroscopic imaging and a computer equipped with a recent GPU in order to compute the tumor detection.

1.4 Structure of this document

The next chapter of this document will be dedicated to describing an introduction to machine learning and an overview of its applications in radiology. The main focus of this chapter will be put on real-time tumor tracking as it is the theme of the method proposed in this work. The third chapter will be dedicated to the generation of the dataset used to train the proposed neural network. It will explain in detail how the data was chosen as well as the data augmentation methods used in the proposed method. The fourth chapter will describe the proposed network topology, its different characteristics as well as the evaluation methodology. Finally, the fifth chapter will assess the performance of the proposed algorithm, as well as propose several paths for further improvement.

1.5 Summary

In this chapter, we have defined the principle of radiotherapy and the specificities of proton therapy. Furthermore, we have described the main advantages and challenges of proton therapy with regard to treating tumors located on moving organs.

Chapter 2

Deep Learning in medical imaging

2.1 The breakthrough of deep learning in radiology

In the last decade, there has been a colossal interest in deep learning, which has brought about breakthroughs in many applications such as image, audio or video processing and many others [18]. The field of medical imaging is no exception to this revolution. Impressive improvements by deep learning, over other machine learning methods, have been demonstrated in the literature [19]. Those successes are now large enough to draw the attention of researchers in the field of computational medical imaging to investigate the potential of deep learning in medical imaging. The method developed in this work to implement real-time tumor tracking is therefore based on deep learning.

2.2 A brief introduction to machine learning

Deep learning is a subset of a broader class of models called machine learning. Machine learning is defined as a field of computer algorithms that can improve automatically through experience and by the use of data[20]. It is more generally a part of artificial intelligence. The core particularity of machine learning compared to more classical approaches to artificial intelligence is the fact that specific rules are not programmed into the model in order to make predictions, but instead learned through a statistical approach.

Most machine learning models can be sorted into three main categories: supervised learning, unsupervised learning, and reinforcement learning [21].

In supervised learning, the data used to train the algorithm is composed of pairs of an input and the associated desired output. The model then constructs a function that tries to predict the output for each training input. It then iteratively compares these predictions to the actual desired outputs and corrects its objective function accordingly in order to get closer to get the predictions as close as possible to the targets. The objective of this training phase is not only to get a function that can predict the correct output for each input provided in the training dataset but most importantly to also predict the correct output for input that was never observed by the model in the training phase.

In unsupervised learning, on the other hand, the data used to train the algorithm is only composed of input data, without an associated output. The goal of unsupervised learning algorithms is therefore to find an underlying structure in the data. This type of learning is useful to find unknown patterns in the data and is often used for clustering. Clustering is the process of dividing a dataset into several groups, or clusters, that share similarities. Clustering is used in many applications such as recommendation engines, unsupervised image segmentation, search engines, or even image compression.

Reinforcement learning is similar to unsupervised learning in the way that it does not require a labeled training set of input and output pairs. It consists in training an agent to evolve and interact in a specific environment based on its previous experience, to optimize a cumulative reward. This type of model is often applied to Markov decision processes, discrete-time stochastic control processes based on probabilistic automata that model decision making. A reinforcement learning model is particularly useful when exact models are unfeasible, for example in a task where we have to replace a human behavior such as autonomous driving, chat-bots, image generation or playing a game against a human. Unlike supervised networks which need the expected targets to improve their predictions, reinforcement learning answers only need basic feedback.

These subdivisions are not completely mutually exclusive and a multitude of other techniques also exist. This work will be mainly focused on supervised machine learning techniques.

The main particularity of deep learning networks compared to other machine learning algorithms is their structure. The term "deep" refers to the use of several successive layers in the network, that allow it to use a high level of abstraction and find intricate structures in the data to generate its predictions. These networks are often referred to as artificial neural networks (ANN). The most basic form of an

ANN is the multilayer perceptron.

2.3 The multilayer perceptron

In order to understand the type of deep learning networks used in this work, the Convolutional Neural Networks (CNN), we first need to understand the more general network they are based on, the MLP. MLPs are a class of ANN and are sometimes colloquially referred to as the classic neural networks, especially when they have a single hidden layer as they are among the simplest forms of ANN.

2.3.1 The perceptron

The building block of MLPs is the perceptron. The perceptron is a simple supervised binary classifier inspired by the topology of a neuron [22]. It takes a vector of elements as an input and generates a single output element as a binary threshold function of a linear combination of the input 2.1. A graphic representation of a simple perceptron is shown in Figure 2.1

$$f(x) = \begin{cases} 1 & \text{if } w * x + b > 0 \\ 0 & \text{otherwise} \end{cases} . \quad (2.1)$$

Where w is the vector composed of the weights associated to each input, x is the input vector and b is the bias.

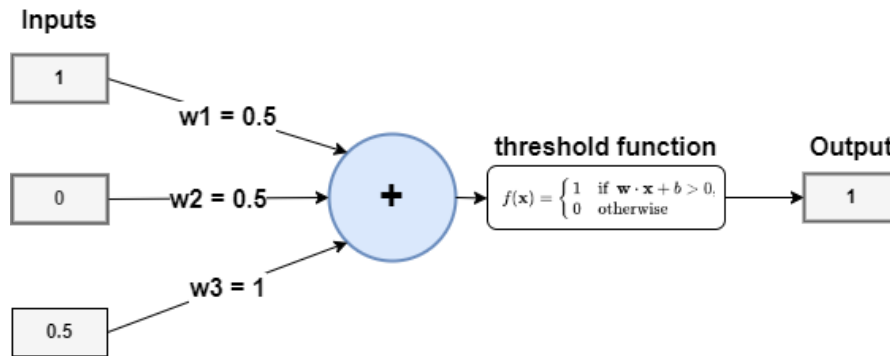


Figure 2.1: Example of a simple perceptron with 3 input values. The weighted sum of the input values is passed through the threshold function to produce a single binary output value.

2.3.2 Building an MLP from perceptrons

A MLP consists of one input layer, one output layer, and at least one layer of artificial neurons. A neuron is the unit composed of a perceptron, followed by a non-linear activation function. The non-linear activation function is used to either activate or inhibit the output of the perceptron depending on its input value. An example of a simple MLP is shown in Figure 2.2. The first layer is referred to as the input layer, the last layer as the output layer, and the intermediary layers are known as the hidden layers. MLPs are fully connected, this means that the output of every neuron of each layer is fed to the input of every neuron of the next layer. Thus, the input of each neuron is the weighted sum of the output of every neuron of the previous layer. The weights of this sum are called the hidden weights and are specific to each neuron. These hidden weights are derived during the training phase of the algorithm [23].

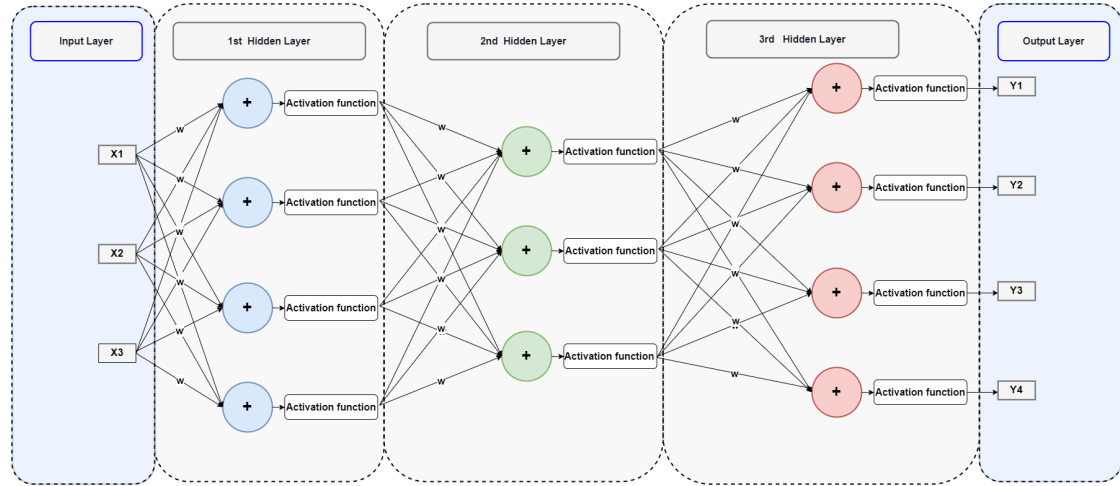


Figure 2.2: Illustration of a simple multilayer perceptron with 3 hidden layers. This type of network is fully connected, every output of each layer is connected to every input of the next layer. The weights "w" of the connections in a hidden layer are independent trainable parameters learned by the network during the training phase.

The training process of a MLP starts by making random predictions, and iteratively adjusting the weights of each connection in order to minimize the loss function (see section 2.5.1) over the training dataset. The most commonly used algorithm for this task is referred to as gradient descent and is explained in further detail in section 2.5.2 for the case of CNNs as this is the main focus of this work.

While the basic form of the MLP is rarely used on its own nowadays, it is the base on which the majority of modern ANNs are built. This is the case of CNNs, the type of network used in this work to implement the tumor detection.

2.4 Convolutional neural networks

A convolutional neural network is a type of deep learning model designed to process data shaped in a matrix form and exploit the spatial features of this data. This capacity to extract spatial structures makes CNNs particularly good at processing images and explains how they became a dominant method in computer vision tasks.

2.4.1 Biological inspiration of CNNs

Convolutional neural networks were inspired by early findings in the study of biological vision [24]. Their creation was heavily influenced by the work of two winners of the 1981 Nobel prize in medicine on the information processing in the visual system, David H. Hubel and Torsten N. Wiesel.

In their work, these physicians uncovered the layered structure of the visual system inside the brains of mammals, and how the cells of one layer interact with neighboring cells at the subsequent levels [25]. Essentially, exiting an optic nerve by shining a light in its direction, might inhibit or enhance the response to the stimulus of neighboring optic nerves. This spatial organization indicates that the cells are making spatial comparisons between the amount of light in one region and its surrounding. Because of this structure, the visual system is not best at observing absolute values but is instead specialized in making comparisons between different areas of interest. Hubel and Wiesel concluded that this specific structure allows the eye to analyze a scene in terms of dark-light contours, and detect the shape of objects. They recognized that they did not yet completely understand how the more complex shapes are recognized by the brain but guessed that the subsequent stages made sense of this information by combining these spatial features.

These discoveries lead to the conception of the neocognitron, a hierarchical and multilayered neural network proposed by Kuniyiko Fukushima in 1979 [26]. This network is used primarily for handwritten character recognition [27] and is the inspiration for the current convolutional neural networks.

2.4.2 Building blocks of convolutional neural networks

In the same way as the visual system brain explained in the previous section, convolutional neural networks use a layered structure and make use of the spatial locality of pictures. Compared to more conventional neural networks such as the MLPs which use the intensity of every value of one layer to produce each element of the next layer, CNNs use small groups of neighboring values in order to generate the next elements thanks to an operation known as convolution (see section 2.4.3). This extraction of spatial features is done by using multiple successive building blocks, composed of different layers. The most common layers include convolutional layers, pooling layers, activation layers as well as fully connected layers. Each of these blocks can operate at different resolutions in order to extract structures of different scales out of the image. An example of a simple CNN is shown in Figure 2.3.

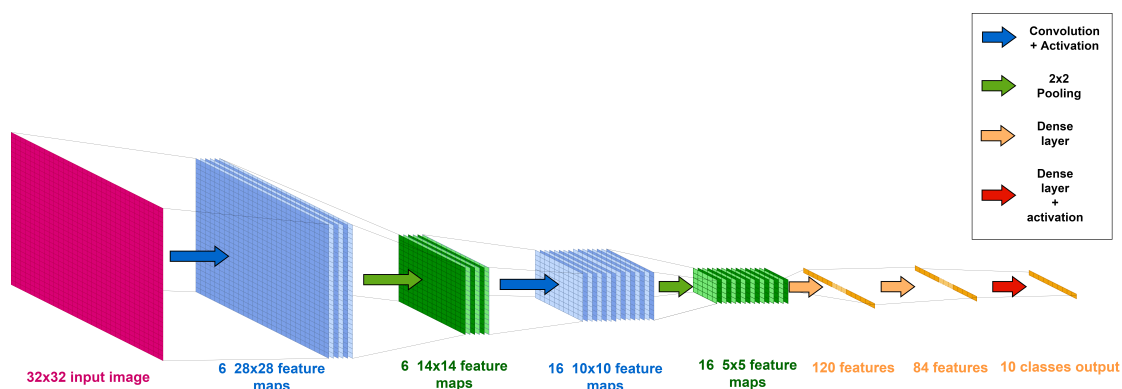


Figure 2.3: Example of a simple CNN (LeNet5 digit recognition network [28]). The channels of each layer are represented by stacked rectangles, each rectangle in the stack represents one channel.

2.4.3 Convolution

The convolutional layer is probably the most important component of convolutional neural networks, because it is this layer that exploits the spatial locality of pictures and really differentiates CNNs from MLPs. This layer prevents the network from using pixels too far apart on the image to extract spatial features, and at the same time reduces the number of trainable weights needed by the network. A convolutional layer generates a feature map based on a weighted sum of adjacent elements of the input array. In order to construct each element of its output array, it uses a small windows-like array, called the kernel. As shown in Figure 2.4, the kernel is overlaid across a part of the input, and the sum of the element-wise

product of the overlapping part of the input and the kernel generates one output element. Each element of the kernel can therefore be seen as the weight of its corresponding input pixel in the sum, the kernel elements are thus often referred to as the weights of the convolution. The kernel is then shifted by a fixed number of elements called the stride to produce the next output value. Therefore, if the stride is smaller than the size of the kernel, neighboring output values will overlap in terms of corresponding input elements. A typical value of the stride is 1, but a larger stride might also be used and results in a down-sampling of the feature maps [29].

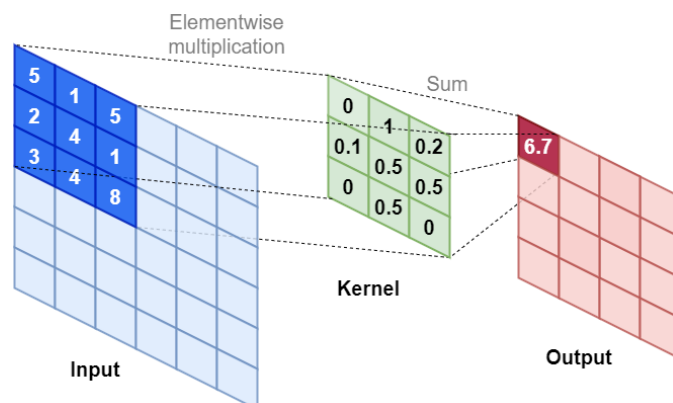


Figure 2.4: Illustration of a small convolutional layer. The kernel is superposed with a part of the input layer and each element of the input is multiplied with the corresponding element of the kernel, the result of these products is then summed to produce one output element. The Kernel is attractively shifted and the process is repeated to compute the whole output array. Illustrations inspired by [29]

In this kind of convolutional layer, the output array will inexorably be smaller than the input array, as the center of the kernel will never be allowed to overlap the edges of the input. This is not always a desirable priority, as it may not allow chaining multiple convolutional layers at will. In order to prevent this reduction in resolution, it is common to apply zero-padding along the edges of the input in order to match the input array size as shown in Figure 2.5.

The kernel used in order to produce one output layer is the same across the whole input. However, in order to produce a multitude of output arrays, each corresponding to a different feature map, this operation is repeated several times with different kernels. Each kernel is therefore a specific feature extractor: one kernel close to the input might extract the edges of an input image, and another might be better at extracting corners, while another one closer to the output might

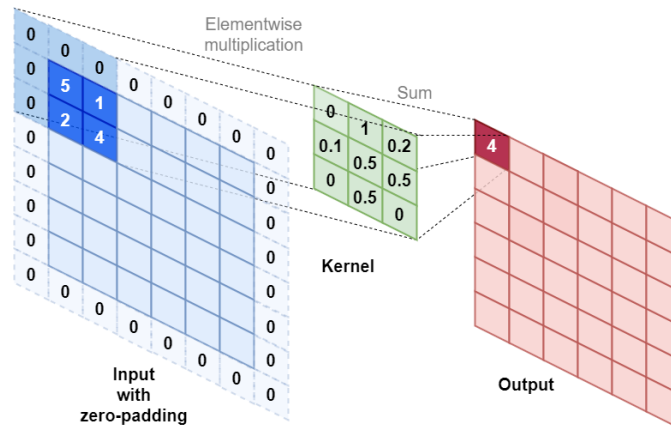


Figure 2.5: Illustration of a convolutional layer with zero-padding. The input layer is extended with a border consisting of zeros in order to produce an output of equal size as the input.

extract higher-level features such as complex shapes or even specific objects. Unlike the stride, the kernel size, the padding, or the number of kernels, the values of the weights are not assigned and fixed in advance, they are instead learned by the network during the training phase (see section 2.5).

2.4.4 Pooling Layer

When using convolutional layers on their own, the resulting network might be highly susceptible to slight shifts in the input, as the extracted feature will depend on the exact positions on the input grid. Because of this effect, minor changes to the input image result in a different feature map. One solution to alleviate this effect is the use of a pooling layer after the convolution. Pooling layers are essentially down-samplers, they divide their input into square groups of equal sizes, and output one single element for each group of input elements which is supposed to be representative of the whole patch. This operation results in a loss of precise localization of the features, but this summary of larger patches of the input gives more context to the feature detection as well as more shift-invariance [30].

There are three main types of pooling layers: max pooling, average pooling, and global pooling [31].

Max pooling is the most common type of pooling, its output is the largest element of the patch of the feature map covered by the filter and it simply discards the other elements as shown in Figure 2.6. As a result, its output contains the

most prominent features of the input.

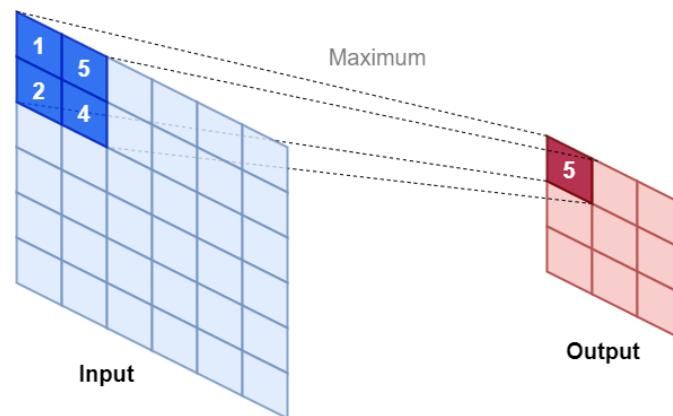


Figure 2.6: Illustration of a max pooling layer with a 2x2 pooling size. The input is divided into patches of equal size and the greatest value in each patch is selected as the output value representing the whole patch.

Average pooling, on the other hand, takes the average of the elements present in the patch covered by the filter. Instead of outputting the most prominent feature, it creates an average of the features present in each patch as shown in Figure 2.7.

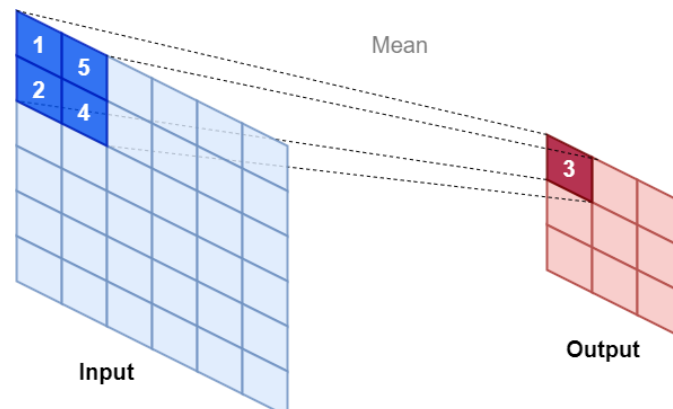
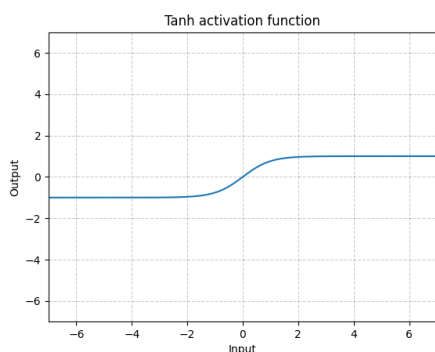


Figure 2.7: Illustration of a average pooling layer with a 2x2 pooling size. The input is divided into patches of equal size and the average of the values of each patch produces the output value representing the whole patch.

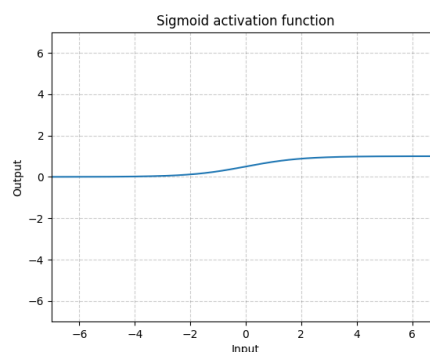
Finally, global pooling flattens all the channels of multidimensional arrays into one single 1D array. A global layer is a normal pooling layer applied with a single patch of the same size as the input array as shown in Figure 2.8. Most global

Some of the first networks use saturating non-linear activation functions, such as hyperbolic tangent or sigmoid functions, because their output range is comprised between 0 and 1 or -1 and 1 and is therefore easily interpretable. The response curve of the activation function is shown in Figure 2.9a and 2.9b respectively. However, activation functions presenting such a small range are subject to a phenomenon called vanishing gradient which prevented the weights of the kernels to get updated in the training phase of deep networks. Because of this problem, most of the recent networks use a Rectified Linear Unit (ReLU) as shown in Figure 2.10. The ReLU activation function is defined by the equation 2.2.

$$f(x) = \max(0, x) \quad (2.2)$$



(a) Hyperbolic tangent [35]



(b) Sigmoid [36]

Figure 2.9: Response curve of the hyperbolic tangeant and Sigmoid activation functions.

Applying the ReLU is computationally easy as it only consists in keeping the positive values, and discarding the negative ones.

Using non-linear activation functions allows to convert the linear input signals into non-linear output signals, this is essential as it allows the network to perform non-linear regression and solve classification problems that are not linearly separable.

2.4.6 Fully Connected layers

Fully connected layers are similar to the layers of an MLP as described in section 2.3. In these layers, each element of the output array is a weighted sum of every

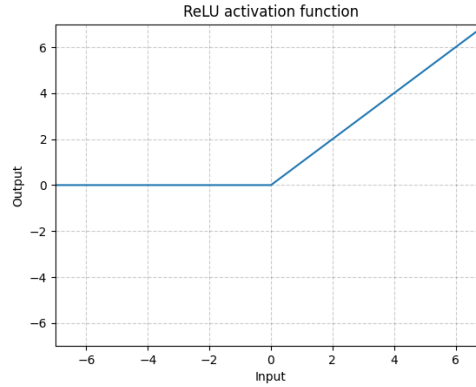


Figure 2.10: Illustration of the response curve of the ReLU activation function [37]

element of the input array. These layers are sometimes also referred to as dense layers. Dense layers are often used as the last layers of a classification network, with each output element corresponding to the probability of belonging to a specific class. This type is less prevalent inside the network because the decrease in the number of trainable weights and the focus on spatial locality is the essence of CNNs. A CNN with only dense layers instead of convolutional layers would essentially be an MLP.

2.4.7 Final Layer activation function

In order to make sense of the raw value outputted by the network, it is needed to pass it through an appropriate activation function. This function will normalize the output and convert it to something akin to class probabilities. The specific activation function used will depend on the task, but the most commons are by far the sigmoid and the softmax functions.

A sigmoid function is usually used in binary classification, when the network has a single output layer. For each element, this function converts its value ranging from $-\infty$ to $+\infty$ into a value ranging from 0 to 1. These values can be used as an estimation of the probability of each pixel to belong to the binary class.

The softmax activation function is a generalization of the sigmoid function to multi-class classification. For a classification problem with N classes, it takes N input values per pixel and outputs N strictly positive values of sum 1 for each pixel. These values can then be used as an estimation of the probability of each pixel to belong to each class and the class with the largest value is selected as the predicted one.

2.5 Training a neural network

The training phase of a neural network is the phase where the network learns the value of its hidden weights. For a convolutional layer, these weights correspond to the elements of the convolutional kernel, and for a fully connected layer to the weight of each input element in the sum.

During the training phase of the network, a subset of the available data referred to as the training data is given to the network. In the case of classification CNN the training set is constituted of pairs of an input image and its corresponding class, whereas in the case of a segmentation CNN, the training data consists of pairs of an input image and its associated segmentation mask.

The goal of the training phase is to optimize the value of the hidden weights in order to get the predictions of the network as close as the provided targets for the training input.

2.5.1 Loss function

The loss function is a cost function, designed to measure the dissimilarity between a prediction of the network and the actual target. A large loss indicates a large divergence between the target and a prediction, while a loss of 0 indicates a perfect prediction. The goal of the training phase is to minimize the value of the loss function over the examples of the training dataset. There is an infinite number of ways to evaluate the similarity measure between two samples, and there is therefore an infinite number of ways to measure the loss.

Most loss functions are derived from common evaluation metrics such as those described in section 2.5.4 or from a linear combination of several of them. The ideal loss function depends on the specific problem the model is trying to solve, depending on the intended characteristics of the predictions. The loss function is the medium used by the programmer to help the network to derive the characteristics of a good prediction based only on the examples seen in training. Some loss functions will be designed to maximize recall, others to maximize precision, and others might try to maximize any other arbitrary characteristics of the predictions. For a system designed to detect life-threatening pathologies for example, one might want to increase the rate of detection at the expense of false positives, in order to minimize false negatives. The loss functions used in this work are described in detail in Section 4.2.

2.5.2 Gradient descent

Gradient descent is a method designed to train a neural network. It is an iterative process in which the partial derivative of the loss function over each trainable weight of the network is computed for the examples of the training set. This partial derivative is referred to as the gradient. The gradient provides the network with a direction and an amplitude to adjust the weight. This amplitude is scaled by a parameter called the learning rate. The goal of the gradient descent is to make the loss function converge to a minimum.

A learning rate too slow will take many iterations to converge to the minimum as shown in Figure 2.11(a). A high learning rate usually leads to a faster-converging network, but a learning rate set too high can lead to an erratic behavior where the updates never lead to the minimum as shown in Figure 2.11(b). A good learning rate should be balanced in order to converge fast enough without this erratic behavior, this is illustrated in Figure 2.11(c). For this reason, adaptive learning rates are sometimes used. Adaptive learning rate algorithms dynamically adjust the learning rate during the learning process, in order to combine swift convergence to the minimum at the beginning of the gradient descent with a more stable behavior when approaching the minimum as illustrated in Figure 2.11(d).

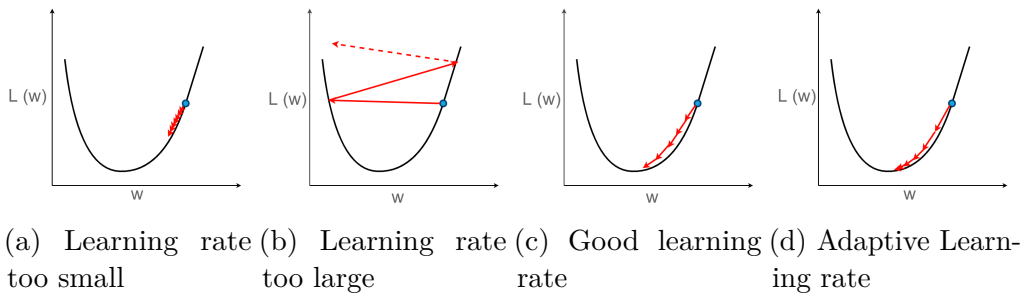


Figure 2.11: Intuitive illustration of the effect of different learning rates on the loss for gradient descent with a single weight w .

Nowadays, in order to reduce the computational cost of computing the gradient, gradient descent is usually replaced with stochastic gradient descent (SGD). In stochastic gradient descent, the actual gradient is replaced with an estimation of the gradient computed from a randomly selected subset of the training data. A common extension of SGD is Adaptive Moment Estimation (Adam), which consists in combining the estimation of the gradient and of the lower moments of the gradient [38]. Other notable and widely used variations of SGD include SGD with momentum [39], RMSprop [40] and AdaGrad [41].

2.5.3 Overfitting

Overfitting occurs when the network performs well on a training set, but its performances drop drastically over data unseen during the training phase. Overfitting often appears when training a network with a large number of trainable parameters and small training data set. The network is therefore able to learn to recognize the examples of the dataset, by learning the whole image or irrelevant patterns in the noise, but is unable to generalize to new examples.

2.5.4 Evaluation

In order to analyze and measure the performance of a network in an unbiased way, to compare it with other methods, or to assess the quality of its predictions we need to define objective evaluation metrics.

Evaluation metrics

The performances of a neural network can be evaluated in a variety of ways. The two main basic evaluation metrics are precision and recall.

Precision, on the one hand, measures the relevancy of the positive results. It is defined as the number of true positives (Tp) over the number of true positives plus the number of false positives (Fp). A high precision reflects that the samples attributed to the class in question have a high probability of actually belonging to this class.

$$P = \frac{Tp}{Tp + Fp} \quad (2.3)$$

Recall, on the other hand, measures the ratio of positives that were detected. It is defined as the number of true positives (Tp) over the number of true positives plus the number of false negatives (Fn). A high recall reflects that the samples actually belonging to the class in question have a high probability of being attributed to this class by the model.

$$R = \frac{Tp}{Tp + Fn} \quad (2.4)$$

On their own, each of these measures only paints a partial picture of the performance of the network. In the case of a network designed to predict a binary classification for each pixel of an image, if the network only predicts "1"s, the recall

will be maximized, but the accuracy will often be very poor. In the case of a network only predicting positive occurrences for the very high confidence samples, the precision would probably be high, but the recall very low.

However, even combined, these two metrics are not always sufficient. This is particularly true in the case of heavy class imbalance. For example, in a binary classification on images containing mostly positive occurrences, a network only predicting "1"s would get both high precision and high recall, while being practically useless.

The Sørensen–Dice coefficient, also referred to as the F1 score or simply the Dice coefficient is another very common similarity measure. It was independently developed by the botanists Thorvald Sørensen [42] and Lee Raymond Dice [43]. One big advantage of the Dice coefficient is the fact that it takes into account class imbalance and reflects both the accuracy and the precision of a classification. In binary classification, the dice coefficient is defined as

$$DSC = \frac{2Tp}{2Tp + Fp + Fn} \quad (2.5)$$

However, this coefficient does not respect the triangle inequality. Another very similar similarity coefficient that respects the triangle inequality is the Jaccard index which is defined as

$$J = \frac{Tp}{Tp + Fp + Fn} \quad (2.6)$$

The Jaccard index can also be calculated as a function of the Dice coefficient (and vice versa) they are therefore equivalent.

$$J = \frac{DSC}{2 - DSC} \quad (2.7)$$

Data set splits

Because of the phenomenon of overfitting described in section 2.5.3, evaluating the performances of a neural network on the training examples only is not a sufficient and reliable method to predict its real-world performances. In order to get a more accurate approximation of the quality of the network's predictions, the most common technique consists in dividing the dataset into three mutually exclusive subsets : The training set, the validation set, and the test set.

The training set is usually the biggest set, this data is the one used by the model during the training phase. Gradient descent will use the quality of the prediction of the network over these examples in order to set the weights of the network.

The validation set is not seen by the model during training and is therefore not used by the model to refine its trainable parameters. Instead, the performance of the network over the validation set is usually used by the programmer to fine-tune the hyper-parameters of the network such as the learning rate, the loss function, or the number of training iterations. Evaluating the performances over this set is therefore a bit more impervious to overfitting than an evaluation over the training set. However, because this dataset is used to decide the meta-parameters of the network, they can still be subject to a fair bit of overfitting [44].

Therefore, in order to get an accurate and unbiased approximation of the quality of the network's prediction over unseen data, we need yet another subset : the test set. This subset of the available data is used neither by the network for training, nor by the programmer to determine the optimal hyper-parameters of the network. This data set is only supposed to be used on the final model after the training phase.

Cross-validation

In cross-validation, instead of using a fixed training and validation set split, the part of the dataset that is used for the training and the validation is divided into a chosen number of parts. At each iteration, one subset is designed as the validation set and the remaining subsets constitute the training data. Once the model has been trained on the training data, it is evaluated on the validation set. This process is repeated until each subset has been used as the validation set, and the final score given to the model consists of the mean of the scores evaluated on each validation subset [45].

Cross-validation can be exhaustive, and use a single observation at the time as the validation set, this is known as "leave-one-out" cross validation (see Figure 2.12b). However, the most used version of cross-validation, known as k-fold cross-validation, consists in dividing the dataset into k subsets of equal length. An illustration of K-fold cross-validation is shown in Figure 2.12a. Other variations of cross-validation exist but are less prevalent and will not be discussed further in this work.

When cross-validation is used in order to derive the optimal hyper-parameters of the network, it can still be preferable to use a separate subset of the data as a test set, totally isolated from the optimization of the hyper-parameters. This test

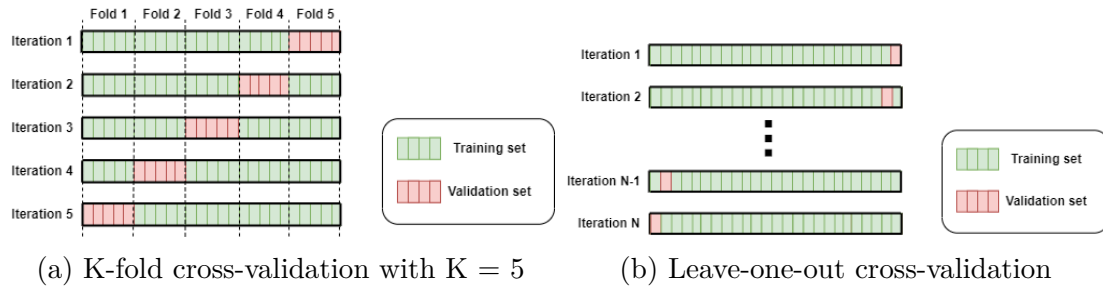


Figure 2.12: Illustrations of the two most common types of cross-validation. Kfold cross-validation divides the dataset in K subsets while leave-one-out divides the dataset in N subsets (with N being the number of samples in the dataset). At each iteration, one subset is used as validation data while the rest is used as training data.

set can simply be chosen statically like in the previous approach. However, it is also possible to use nested cross-validation for this purpose. One possibility is to use at each iteration one of the subsets as a validation set, one as a test set, and the rest as the training set. In this case, the choice of the hyper-parameters is based on the results on the validation set alone, and the scores over the test sets is only used to evaluate the final performance of the network. Another possibility would be to do a first K-fold, where each subset is iteratively used as the test set and cross-validation is applied to the rest of the data in order to determine both the hyper-parameters and the trainable weights. In this scenario however, the hyper-parameters of the network could differ across each fold and the final hyper-parameters need to be chosen among the different iterations of the K-fold of the highest order.

2.6 U-net

U-net is a particular implementation of CNNs developed in 2015 for medical image segmentation by Olaf Ronneberger, Philipp Fischer, and Thomas Brox at the german university of Freiburg [46]. This type of CNN takes a full resolution image as its input and outputs another full resolution image. This network is now used in a variety of other domains, such as image super-sampling [47], object detection [48], image forgery detection [49], or as a component of a generative adversarial network [50], but its most common usage is image segmentation and pixel-based classification. Over the year, several variations of the original U-net have been proposed and optimized for specific purposes such as variations with additional skip connections [51], networks optimized for nuclei segmentation in histology [52], 3D tumors and liver segmentation [53], nerve segmentation [54] and many

others. Nowadays, the name "U-net" is often used to describe not only the specific implementation of Olaf Ronneberger et al. but also the numerous variations and extensions derived from it.

2.6.1 Encoder-Decoder inspirations

U-net is based on an encoder-decoder architecture. A traditional encoder-decoder network takes a high number of parameters as an input, extracts its features in order to represent it in a compact state vector representation, and uses this abstract representation to construct a high dimension output.

The first part of the U-net is the encoder part of the network. It resembles a traditional contracting CNN in the way that the successive blocks use an incremental number of layers, with a decreasing resolution.

The second part of the U-net is the decoder part of the network. It is similar to a traditional CNN, but the pooling layers are replaced by up-sampling layers. Because of this difference, the output of the encoder is progressively scaled back to the full resolution in the decoder part.

2.6.2 The skip connections

One big difference between a traditional encoder-decoder network and the U-net is the use of skip-connections, connections that bypass parts of the network and skip feature maps at the bottom of the "u" structure. While an encoder constructs its output exclusively from its compact state vector representation, the U-net makes direct use of the output of every block in the contracting phase in order to generate its output. The input of each expanding block is therefore the concatenation of the output of the previous block with the output of the contracting block of the same level as shown in Figure 2.13. This direct connection between the output of a contracting block and the input of an expanding block is known as a "skip connection", because it bypasses parts of the network to connect the output of some blocks directly to the input of blocks further in the network. A skip-connection allows the network to construct its predictions based on a combination of precisely located low-level features of the first levels of the network, and the more abstract features that necessitate a wider spatial context but are not as precisely located on the image because of the smaller resolution of the deep levels feature maps.

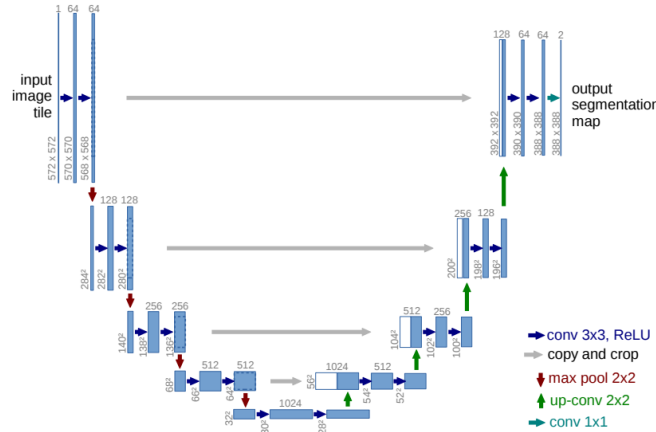


Figure 2.13: Original U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations [55].

2.7 Deep learning in radiology and medical images segmentation

During the last decade, with the development of artificial intelligence in a large number of domains, there has been a growing interest in research for applications of deep learning in radiology as one can observe in Figure 2.14. However, these techniques are still rarely applied in practice, and traditional methods relying on manual inspection of the images by professionals are still prevalent. This section explores the motivations for machine learning in radiology, its challenges, and its current applications, particularly in the context of tumor segmentation on X-ray fluoroscopic images.

2.7.1 Motivations

The motivations for the usage of deep learning in radiology are numerous.

The first advantage of deep learning compared to manual inspection is the acceleration of the diagnostic. Indeed, with sufficient computational power, an automated system is capable of analyzing a very large number of images at the same time, at any time of the day or the night. A computer does not need to sleep and is able to process data at inhumane speed. This increase in diagnosis speed is

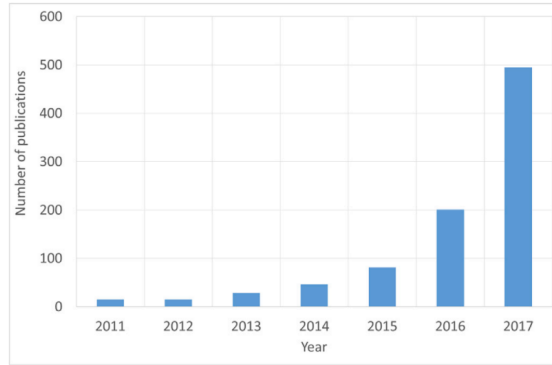


Figure 2.14: Number of publications for the search phrases with at least the terms radiotherapy or radiation therapy or radiation oncology and at least the terms deep learning or deep network or convolutional network [56].

even bigger nowadays thanks to the advancement of GPU-based machine learning [57, 58]. This technical leap makes it possible to use real-time algorithms, to either assist or even automate some medical procedures.

A second motivation for the usage of deep learning in radiology is to eliminate human perception bias from diagnosis. Indeed, radiologists are human and could therefore be subject to biases that may lead to medical errors such as cognitive bias, anchoring bias, or confirmation bias [59]. Furthermore, even an experienced radiologist is only exposed in his work to a sample of the possible pathologies. An autonomous system based on deep learning could process data from numerous sources and therefore have multiple examples of rare diseases that may be frequently overlooked. Even used as a secondary tool for the radiologist, the system might raise a flag in order to notify the practitioner of the need for a thorough inspection or additional testing in order to rule out a specific pathology.

Another important advantage of deep learning based image processing is the ability to detect objects invisible to the eye on images, based on correlation with the surrounding structures or complex structural patterns, thanks to the numerous filters used in CNNs.

2.7.2 Challenges

Despite promising results, AI-based solutions are still rarely applied in the clinical workflow.

One big challenge for AI-driven radiology would be the legal liability in case of misdiagnosis. If a medical error was to result from an erroneous prediction made by an algorithm, it would be difficult to attribute the fault to someone. Is the mistake the fault of the radiologist, the software engineer, or the manufacturer of the imaging device?

Another challenge is the access to sufficient data. Indeed, most deep learning systems need a lot of data to train and to be able to generalize based on previous observations. However, medical data is private and companies often want to protect their intellectual property and keep the datasets proprietary. Moreover, there may not be a consensus on the format of annotated medical images. Additionally, annotated examples for medical images are inherently difficult to produce, as they necessitate the intervention of specially trained radiologists. Conversely, classifiers for everyday images could be trained by ordinary people, and labeled datasets can be easily crowd-sourced. Google, for example, uses the results of its "reCaptcha" anti-botting system to continuously adapt its classification network based on human interaction [60].

Furthermore, neural networks can be statistically impressive, but individually unreliable, as they could potentially make mistakes that humans would not [61]. Some AI-based classification networks perform better than human observers most of the time, but sometimes make enormous mistakes like confusing a tree with a cat because of a bias in the dataset where all the cats are represented over a green background. This type of mistake is particularly prevalent in the case of overfitting (see section 2.5.3), when the network makes its predictions based on non-general criteria. Some researchers are also able to alter the data in subtle ways in order to mislead the classifications of neural networks on purpose. Using very small perturbation vectors on natural images, in a way that is invisible to the human eye, they are able to misclassify the images with a very high confidence probability [62]. This shows that neural networks do not detect features in the exact same way as humans, and we must therefore be careful with their use. In order to better understand the features used by neural networks, Zeiler and Fergus [63] developed a way to visualize the feature maps at the different levels of CNNs and a localization map of the spatial zones of the images used. This technique is shown in Figure 2.15. Zhou and al. also proposed another technique to produce localization maps of the spatial zones of the images used by CNNs to produce their prediction. These types of tools could lead to a better certification of neural networks, and a way to verify that they base their prediction on meaningful patterns of the data.

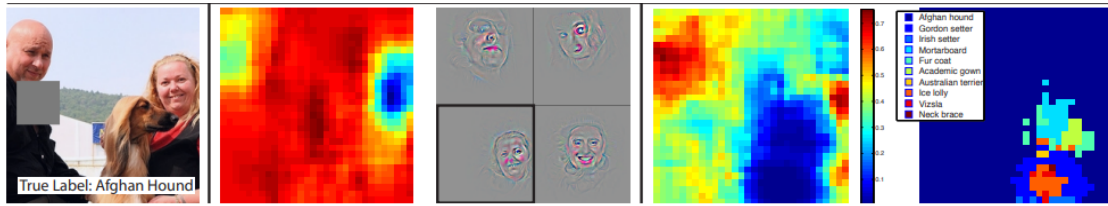


Figure 2.15: One example of the visualization technique proposed by Zeiler and Fergus. Starting from the left, the first image shows how different portions of the scene are systematically covered up with a gray square. The second image shows the activation of the feature map of a top layer. The third image shows the visualization of the features of the same layer. The fourth image represents the localization map of the spatial zones of the images used by the classifier to identify the dog. The last image shows a map of the correct class as a function of the position of the grey square [63].

2.7.3 Pathology detection

Deep learning is a powerful tool in pathology detection, numerous algorithms and models work remarkably well for detecting pathologies in medical data. The method used for pathology detection in the medical field can be based on various sources of data, such as the analysis of medical markers [64], or even voice recording analysis [65]. However, in the field of radiology, the most promising systems are based on computer vision and often involve some forms of CNNs.

The CNNs used for pathology detection of medical images are based on the classical classification CNNs, which take an image as its input, and return the probability for the image of belonging to a set of classes. These approaches have shown very promising results at detecting pathologies, such as diagnosis of prostate cancer, which would usually necessitate an invasive biopsy, achieved area under curve (AUC) of 0.89 and 0.78 to classify cancer vs. benign and high- vs. low-risk of prostate cancer, based on MR images only [66]. Other examples include the numerous CNN models developed for the detection of breast cancer on thermograms [67, 68, 69, 70, 71, 72] with achieved accuracy oscillating between 92 for the least performant methods and 100 percent for the best ones.

The previous subsection 2.7.2, explained the difficulties of obtaining sufficient training data. However, research has shown that CNNs trained on non-medical dataset can also be very powerful for pathology detection. Indeed, Yaniv Bar et al. [73], were able to obtain an AUC of 0.87-0.94 for the different pathologies using CNNs trained with non-medical images. Furthermore, a lot of research is focused

on using data augmentation techniques on medical images in order to improve the small datasets [74, 75, 76, 77]. Data augmentation aims to generate additional data which is used to train the model and has been shown to improve performance when validated on a separate unseen dataset.

2.7.4 Segmentation of medical images

Autonomous organs and regions of interest (ROI) segmentation is a task where deep learning approaches are shown to be very effective. Because of the complex morphology of the human body, and the high variations between patients, non-machine learning based approaches can be unreliable or hard to conceive. Thanks to the U-net and other similar full resolution classification networks, deep learning algorithms are able to learn complex relationships between spatial structures, and segment tumors, organs or other ROIs with very high accuracy. Notable examples include segmentation of the brain anatomy [78, 79], liver tumor segmentation [53], brain tumor segmentation [80, 81] and prostate tumor segmentation [82].

Real-time tumor segmentation for proton therapy

One advantage of using CNN-based segmentation compared to other approaches is the relative speed at which segmentation can be made. In such a manner, a CNN takes a long time to train, but once trained has a very short inference time, particularly when the operations are performed with a GPU. It is therefore possible, depending on the exact GPU speed and the resolutions of the images, to segment images in a few milliseconds [83, 84, 84]. This incredible speed could allow using these segmentations during real-time applications, in order to automate procedures or enhance surgeon's visions with augmented reality.

One particularly promising application of real-time image segmentation is the possibility to use real-time tumor tracking in proton therapy, in order to better target moving organs. As stated in section 1.2.3, one big challenge of proton therapy and radiotherapy in general is to target the tumorous cells, while minimizing damage to the surrounding tissues and vital organs. However, tumors located on organs such as the lungs or the liver move a lot during the different phases of the respiration cycle. The current solution to alleviate this problem is to aim the charged particles at the whole range of motion of the tumor, in order to avoid missing tumor cells and ensure the destruction of the tumor. While this may be effective to target the tumor volume, it results in collateral damages to the adjacent tissue.

Having a real-time prediction of the precise location of the tumor allows to adjust the target of the charged particles over time, and increase the effectiveness of the treatment while minimizing the negative side effects of proton therapy. Because of the incredible advancement such a technology would provide, numerous researchers are currently attempting to develop this kind of tool.

One technology that could greatly improve real-time tumor tracking on moving organs such as the lungs is **MRI-guided proton therapy**. MRI-guided radiotherapy is a technique that currently only exists in photon based radiotherapy. MRI-Linac machines offer superior high-definition image quality, especially for some soft tissue, as compared to traditional linear accelerators which use x-ray-based fluoroscopic imaging to visualize the target area and adjacent anatomy for treatment setup and delivery [85]. With these high-resolution 3D or slice images at our disposal, the tumor is often clearly visible, and therefore easier to track than on fluoroscopic images which are 2D projections and have a weaker contrast. However, MRI-Linac machines are still very rare and only currently exist for photon based radiotherapy. It will be complex to implement this kind of technology for proton-therapy because of the interaction between the charged protons and the magnetic fields of the MRI and research for this particular subject is still in its infancy [86, 87]. Providing real-time magnetic resonance imaging to proton therapy seems possible and might in the future combine the advantages of MRI-Linac with the advantages of proton-therapy [88], but they will probably not be available before a few years [89], if ever. It is thus crucial to develop algorithms able to perform high quality tumor tracking on low-power x-ray fluoroscopic images in order to bring the advantages of real-time guided radiotherapy to proton therapy with its current hardware limitations.

Another solution that can be used in order to track a moving lung tumor precisely without disposing of real-time MRI imaging is to use **fiducial markers**. This consists in implanting small markers that are highly visible on fluoroscopic imaging around the tumor volume in order to be able to follow its displacement. However, this technique is very invasive and the markers are only effective for 1 to 2 weeks [90] and is therefore not always an option. Fiducial markers placed in the lung may result in a complication called pneumothorax or collapsed lung, in which air becomes trapped in the space between the lung and the chest wall. This condition may require the insertion of a chest tube to remove the air pocket and re-expand the lung. Furthermore, any procedure that penetrates the skin carries a risk of infection, even if the chance of infection requiring antibiotic treatment appears to be less than one in 1,000 [91]. Having a non-invasive tracking technique at our disposal would involve fewer risks and be more comfortable for the patient.

A proposed solution that is not CNN based but is worth mentioning is the one proposed by Xu et al. [92] based on optical flow. In this research paper, they describe a **markerless lung tumor tracking in fluoroscopic video, based on the motion vectors**. Their algorithm starts with an initial tumor contouring and adjusts this contouring on the subsequent frames based on the optical flow in the fluoroscopic video. This solution showed good results, as the resulting position difference in 95% of the frames was found to be less than 1.4 pixels (~ 0.07 mm) in the best case and 2.8 pixels (~ 0.14 mm) in the worst case for the five patients studied. However, this method requires a precise manual contouring of the tumor in the first frame of the video to work.

Another non-CNN based solution would be to use a **template based method**. Cui Y. et al. [93] used a method that consisted of constructing several templates at different phases of the respiration cycle based on orthogonal fluoroscopic images. The templates are then matched with the fluoroscopic images available during the procedure. This approach also provided impressive results with average localization errors of less than 2 pixels (1 mm) and the error at a 95% confidence level of about 2–4 pixels ($\sim 1 - 2$ mm)

Toshiyuki Terunuma et al. [94] described another **markerless tumor-contouring method that uses a U-net**. In this research paper, they used a novel method in order to decorrelate the position of the tumor with the skeleton on the image and to make the system consider the position of the soft tissues as a better indicator. The way they implement it, is to first segment the skeleton from the soft tissues on the 3D Computerized Tomography (CT) images, and then generate simulated X-ray fluoroscopy, where the bones are randomly shifted from the rest of the image. This allows for remarkable performance with patient specific-training, even when the tumor is partially hidden, as the network is able to learn the shape of the tumor. On their testing equipment, they achieved a 25 ms inference time per image for 256x256 pixels images. On their data-set, they were able to obtain an average Jacquard index of 0.966, with a confidence interval ranging from 0.906 to 0.994.

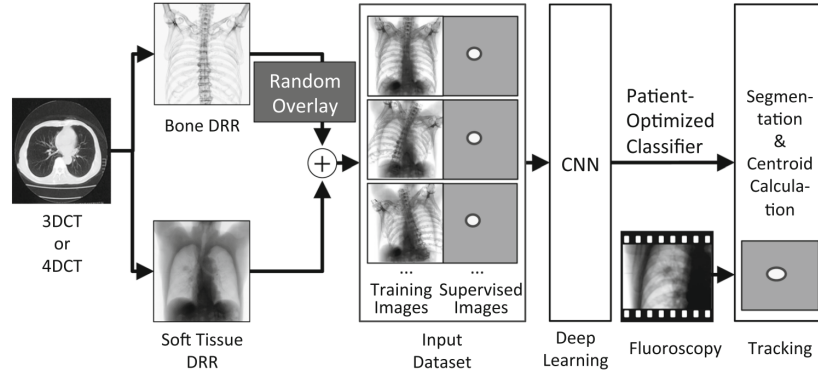


Figure 2.16: Overall workflow of the proposed tumor-contouring method of Toshiyuki Terunuma et al. [94]

This last research paper was a great source of inspiration for the workflow used in this work which tries to further improve this real-time tumor segmentation algorithm.

Novel elements of this work

The objective of this work is to further explore the applications of U-net based tumor segmentation on simulated fluoroscopic images derived from 4D CT scans. The goal of the proposed algorithm will be to provide real-time tumor segmentation based on patient-specific training. Patient-specific training means that the data used to train the model used in the treatment of one patient is derived from this patient alone. Because of the patient-specific training, the model is able to learn prior knowledge of the patient's anatomy and tumor shape and will be able to deduce obstructed information. In contrast with what has been done in the work of Toshiyuki Terunuma et al., the dataset is not augmented by applying random shifts of the skeleton. Instead, it will be generated by simulating realistic respiratory movements on the 3D CT scans before simulating the 2D fluoroscopic images as well as other image deformations.

By simulating enough fluoroscopic images at different parts of the respiratory cycle, and feeding this dataset to the network during training, the U-net should be able to learn the shape and position of the tumor depending on the position and deformations of the surrounding tissues. The network should therefore be able to predict the position of the tumor based only on the low dose X-ray available during the procedure, with a high degree of accuracy even when the tumor is not

visible to the naked eye.

The U-net used in this work will be specifically tuned and adjusted based on a modified version of the original U-net model and will use custom loss functions and data processing in order to best fit the application and constraints of real-time tumor tracking for proton therapy.

2.8 Summary

In this chapter, we have described the principles, and fundamental mechanisms of machine learning as well as its applications to radiology and radiotherapy. We began by defining the concept of machine learning and explaining the different breakthroughs leading to CNNs, a type of neural network which dominates the field of computer vision in recent years. We then explained the different components of CNNs and the particularities of the U-net. In the following part of the chapter, we explained the motivations and challenges associated with the use of deep learning in radiology and radiotherapy and summarized its different applications. Finally, we explained how this work fits in the state of the art of deep learning in radiology and how it differs from previous works.

Chapter 3

Generation of the training data

3.1 Synthetic data

In order to test the feasibility and the basic mechanics of such a system, I decided to first generate basic synthetic data. This data represents simpler, cleaner X-ray fluoroscopic images composed of basic geometric shapes, two examples of this synthetic data are shown in Figure 3.1. This approach is inspired by the work of Terunuma et al. on real-time tumor-contouring [94]. The images are grayscale and have a resolution of 128x128 pixels. In those images, the background is composed of a gray uniform color, the tumors are represented by darker ellipsoids with a major axis of 18 pixels and a minor axis of 15 pixels. This tumor can move randomly with a uniform distribution on both axes of the image. The range on the vertical axis varies from $1/8$ to $7/8$ of the image height and the range on the horizontal axis from $4/10$ to $7/10$ of the image width. The range is larger on the vertical axis in order to mimic the respiratory movements. This ellipse can also rotate with an angle uniformly distributed between 0 and 30 degrees. The simulated tumor is shown in Figure 3.1(a) and the associated mask in Figure 3.1(g).

In order to make the detection task on these images a bit more challenging, a few modifications were added to the input data such as visual obstructions, noise, and other shapes similar to the tumor. The overall goal of this first approach is to show the feasibility of a U-net based detector for tumors partially hidden in fluoroscopic images.

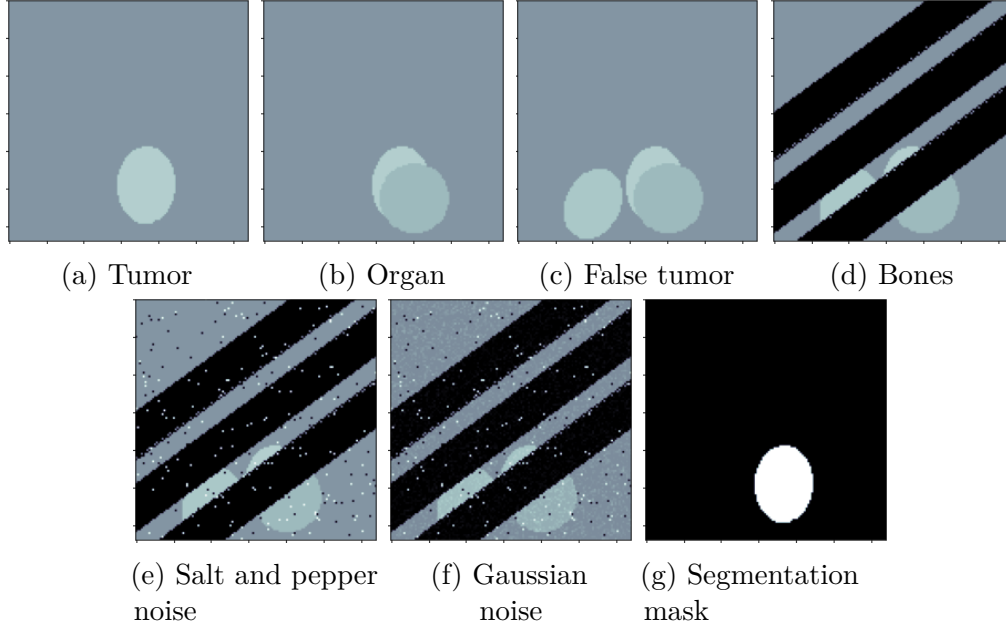


Figure 3.1: Different steps of the generation of a simulated fluoroscopic image and its associated tumor segmentation mask.

The first modification is the addition of a round shape with a radius of 18 pixels filled with a fixed color close to the color of the symbolic tumor. This shape represents an organ of which the position is highly correlated with the position of the tumor. This modification is shown in Figure 3.1(b). Its position is determined by equation 3.1.

$$p_{organ} = p_{tumor} + (random(5, 10), (random(6, 9))) \quad (3.1)$$

The second modification is the addition of another ellipsoidal shape filled with a random color close to the color of the symbolic tumor. This shape is added in order to verify that the system does not have false positives for every organ or shape similar to the tumor that the system might encounter in real X-ray fluoroscopy. If the system targets every round shape as a tumor, it could erroneously detect tumors everywhere in the body which could lead to disastrous consequences. Its position follows an uniform distribution across the whole image. This modification is shown in Figure 3.1(c).

The third modification is the addition of opaque parallel rectangles, representing the bones of the patient, masking parts of the simulated tumor. Indeed, in the real fluoroscopic images, the tumor is rarely clearly visible, because of organs and bones

masking it, we therefore need to verify that the system is able to detect hidden parts of the tumors based on contextual information and its prior knowledge of the basic shape of the tumor. The center rectangle is centered in the middle of the image and the two others are spaced 1/4 of the size of the image apart from it. This is shown in Figure 3.1(d). The angle of these simulated parallel bones is randomly determined by the equation 3.2.

$$angle_{bones} = random(-45, 45) \quad (3.2)$$

Where $random(x, y)$ is a function which returns a random integer uniformly distributed between x and y .

The last modifications are the addition of salt and pepper noise and a Gaussian noise. These modifications are shown in Figure 3.1(e) and Figure 3.1(f) respectively. This prevents the network from basing its decision on a simple thresholding. This choice was made because organs and tissues do not appear with uniform colors on X-ray images, there are always subtle variations and the presence of a fair bit of noise.

3.2 Simulating fluoroscopy from 3D Computed Tomography scans

Our training data consists of 4D CT scans, sequences of 3D CT volumes from 5 patients with lung tumors at 10 different phases of the respiratory cycle. However, during the treatment phase, we only have real-time 2D X-Ray fluoroscopic images at our disposal. In order to train the system to detect tumors on these 2D projected images, we have to simulate them from the 3D CT scans. This roughly corresponds to projecting simulated X-rays through the 3D tomographic image onto a 2D plane. This operation is done via the TomoPy python library, which is a tomographic reconstruction library.

The 3D CTs are initially composed of around 150 512*512 images and this number of slices is not consistent, as it varies from 142 to 173 slices. Furthermore, the initial 3D CTs include information that is not needed for network training, e.g. table, belt, ... In order to eliminate this unnecessary information, the images are cropped around the ROI corresponding to the patient's body. This means that the resolutions are no longer identical for the different patients. these slices have been cropped to contain only the region of interest, and their resolution therefore also varies. Because of those two characteristics of the dataset, a straightforward

projection of the 3D images results in stretched images of various resolutions as shown in Figure 3.2.

For every image projected in this way, the corresponding tumor mask also has to be projected in the exact same way in order to still represent the right position for the tumor. One slight difference is that, since we want to keep a binary mask, this projection has to be thresholded and converted into a binary image. Otherwise, the resulting mask is not binary but is instead greater at the center of the tumor, where the depth through the tumor is greater, and smaller near the edges of the tumor.

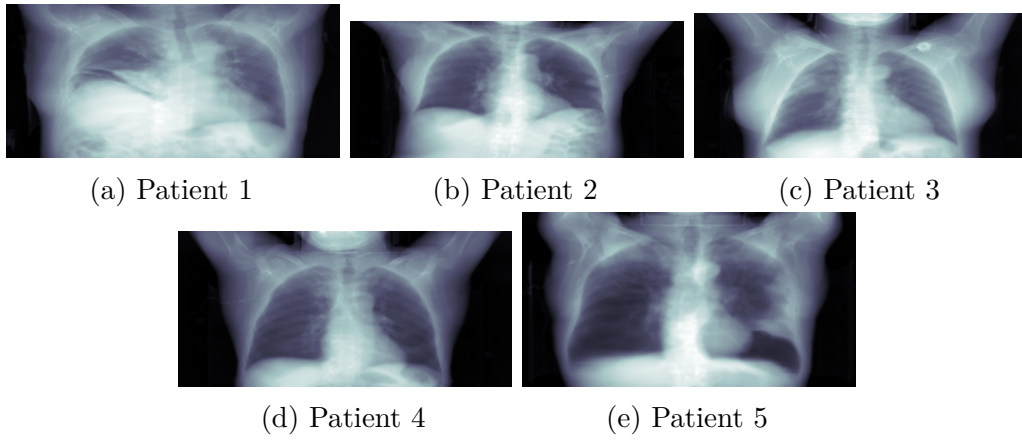


Figure 3.2: Simple projection of the 3D CT onto a 2D plane with the python library Tomopy. The resulting images are stretched because of the uneven shape of the 3D CT images.

In order to remove the deformation resulting from the uneven number and dimension of slices as well as make the resolution more consistent for every patient, it is preferable to resize these images. A resizing of every simulated fluoroscopic image to a 256x256 resolution is therefore applied. For this implementation, a bilinear interpolation was chosen. This type of interpolation results in more realistic images than a simple nearest-neighbor interpolation, by not duplicating pixel values, but is still simple to implement and computationally light. However, because of the bilinear interpolation, once stretched, the masks would once again become non-binary and present slight alterations in intensity on the edges of the tumors. In order to avoid this issue, nearest-neighbor interpolation is used for the resizing of the masks. The result of this resizing is shown in Figure 3.3 with the contour of the segmentation mask highlighted in red. The shape of the tumor is very difficult to discern on the fluoroscopic image with the naked eye.

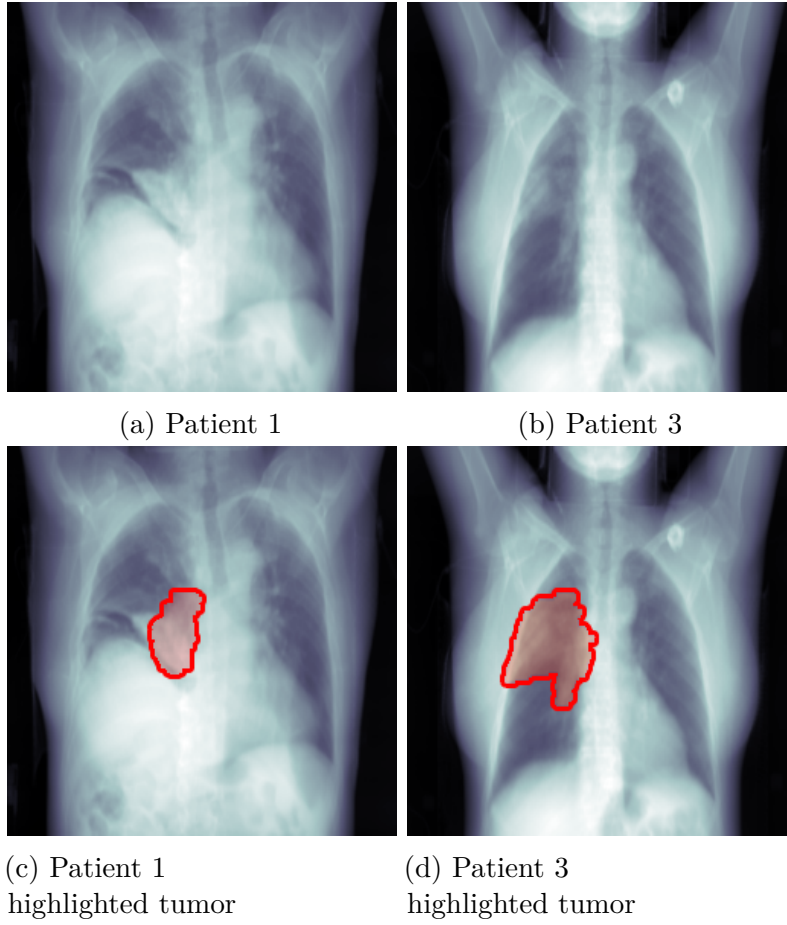


Figure 3.3: Simulated fluoroscopic images of two patients, reshaped to a standardized 256x256 pixels resolution ((a), (b)). Figure (c) and Figure (d) present the same images with the contour of the tumor highlighted in red. The resizing is done with a bi-linear interpolation for the images and nearest-neighbor interpolation for the segmentation masks.

3.3 Data augmentation tools

During the treatment planning data-acquisition procedure, only a few 3D CTs of the patient are acquired. This number of CT scans is limited to ten, that are supposed to be spaced in a way to represent every key phase of the respiration cycle. If we only convert those scans into simulated fluoroscopic images, we would have a very limited number of training data, and it would be impossible for the network to derive any useful data and make good predictions on unseen data, the

network would probably overfit. In order to mitigate this problem, some data augmentation techniques were applied to the data.

3.3.1 OpenTPS data augmentation

The open treatment planning system (OpenTPS) is a software developed by the MIRO laboratory from UCLouvain, with the help of piLAB. This toolbox allows manipulations of 3D CT images. One of its tools allows to generate a mid position (MidP) image from a sequence of 10 3D CT images. The midP represents the average position of the patient's anatomy over the respiratory cycle. Another tool provided by OpenTPS allows the generation of new phases of the respiration cycle from this midP image.

As shown in Figure 3.4, we assign a discrete phase number to each image of the original sequence. For each of the existing phase, the associated deformation field starting from the midP is generated during the computation of the midP. It is possible to simulate an almost infinite number of additional respiration phases by computing the weighted sum of two existing neighboring phases as described in equation 3.3a. The resulting 3D CT and segmentation mask of each of those new phases can then be obtained by applying the new deformation fields to the midP image. Therefore, in order to generate a new phase, we generate a decimal number between 0 and 1 and multiply it by the total number of phases in the original 4D CT (10 in this case). The two closest integer values determine the existing phase used, and the ratio of the sum is determined by equation 3.3b.

As described in equation 3.3c, we can even further modulate these new deformation fields. Indeed, by modulating the intensity of this deformation field, we can get the resulting image to be closer to the midP image with a ratio smaller than 1, or accentuate the deformation with a ratio bigger than 1. This intensity is referred to as the amplitude of the deformation field. Each of the existing phases of the 4D CT is defined as having an amplitude of 1, and the midP corresponds to an amplitude of 0.

$$D_{INTER} = D_N + (D_{N+1} - D_N) * ratio \quad (3.3a)$$

$$ratio = \frac{P_N - P_{NEW}}{P_N - P_{N+1}} \quad (3.3b)$$

$$D_{NEW} = D_{INTER} * amplitude \quad (3.3c)$$

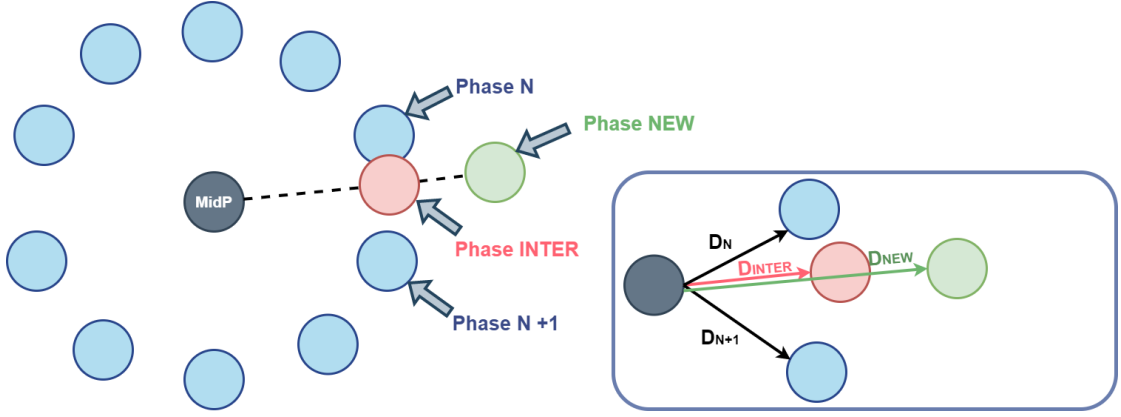


Figure 3.4: Generation of a new phase of the respiration cycle with amplitude modulation in OpenTPS. The blue circles represent the existing phases of the 4D CT, the red circle represents the new phase before amplitude modulation and the green circle represents the new phase after amplitude modulation.

Therefore, using this method with randomly generated phases and amplitudes allows the creation of an almost infinite number of new realistic 3D CT images. One limitation to the number of generated images is the resolutions of the images, which would render invisible transformations too small to change pixel values. Furthermore, it could be possible for the patient to breathe in an unpredictable way outside of the acquisition of the 4D CT, in a way that does not correspond to any linear combination of the images of the 4D CT. This system is therefore not exhaustive. Nevertheless, it should generate images that are way more realistic than the methods proposed before, and therefore results in a better and more generalizable training of the network.

3.3.2 Random noise

As all of the images included in a training set come from the same scanning session, and are the product of applying deformation fields to the same image, there is a high correlation between all of the images, and this could make the model very vulnerable to noise. In order to evaluate this effect, and make the model more robust to noise, random Gaussian noise is added to its training input. This should help to decorrelate the small-scale intensity variations due to noise that might occur either during the validation phase or on real fluoroscopic images.

3.3.3 Random crop and rotate

When we generate our training data, we try to get a projection angle as close as possible to the angle of the X-ray fluoroscopy used during the treatment phase. However, it is really difficult to get the exact same framing and small shifts during treatment might result in a drastically diminishing accuracy, as the images will be different from any other image seen by the network during treatment. In order to measure this effect as well as to try to reduce it, small rotations with a random uniform distribution of the angle between -5 and +5 degrees and random cropping are applied to the dataset, and to the validation set.

3.4 Composition of the dataset

In order to evaluate the capabilities of this system in regard to patient-specific training, 5 different datasets were derived from the 4D-CT of 5 lung tumor patients. Each of those datasets is divided into 5 different subsets of 300 images each. All of these sets are generated with a uniformly distributed phase given to the OpenTPS tool.

The first set is constructed with a uniform amplitude distribution, ranging from 0.7 to 1.2. This set is therefore supposed to represent equally each possible state of the respiration cycle, with a normal range of motion.

The second set is constructed with a uniform amplitude distribution, too, but ranging from 0.8 to 1.4. This set is also supposed to represent equally each possible state of the respiration cycle, but with generally deeper breathing.

The third set is constructed with a Gaussian amplitude distribution, ranging from 0.8 to 1.4. This set is therefore supposed to represent the most common states of the respiration cycles, with less really deep breathing states.

The fourth set is very similar to the third one, with a Gaussian amplitude distribution, ranging from 0.6 to 1.3. It puts more emphasis on shallow breaths.

And the fifth and final set is the one with the widest range, with a Gaussian amplitude distribution ranging from 0.5 to 1.5.

These datasets can be combined in order to form a larger dataset of 1500 images, or we can also train on some sets and use the others for evaluation in order to

measure the effects of the training range on evaluation performance depending on the real range of the respiration.

3.5 Summary

In this chapter, we detailed how the different datasets used in this work were constructed. We began by explaining the synthetic dataset used to train the early versions of the neural network. We then explain how X-ray fluoroscopic images are simulated from 4D CT images. Next, we explained how data augmentations were used to improve and extend the original dataset composed of only ten respiratory phases into larger datasets. Finally, we explained the composition of the different datasets used in the training and validation of the network.

Chapter 4

Real-time tumor detection

4.1 The Neural Network

4.1.1 Topology of the network

The network conceived and used in this work is based on a slightly modified U-net architecture. As shown in Figure 4.1 it has an encoder part composed of 4 contract blocks, and a decoder part composed of 4 expand blocks. The first major difference between this network and the classical U-net is the variation in size of the convolutional kernel, which goes from 3x3 on the original U-net to 5x5 on this one. The second major difference is the removal of the cropping operation in the skip connections, since this model works with an output size equal to the input size. The third difference, is the removal of one stage of the network, as this one only has 4 contract blocks and 4 expand blocks as stated earlier, and only three skip connections as a result. Finally, batch normalization has been added after each convolution.

Each contract block of this network is composed of two sequences of a 2D convolution, a 2D batch normalization, and one ReLu. This duplicated sequence is then followed by a 2D Max pooling layer.

The expand blocks are identical to the contract blocks, except for the fact that the Max pooling layer is replaced by a transposed convolution layer. This layer acts as an up-sampling layer, as it up-scales the resolution of the images by a factor of two (see below).

This model can be used at many different resolutions, with square images with a resolution that is a power of two, but is mainly used with a 256x256 input and output in this work. The input and output resolutions of the model are always perfectly symmetrical.

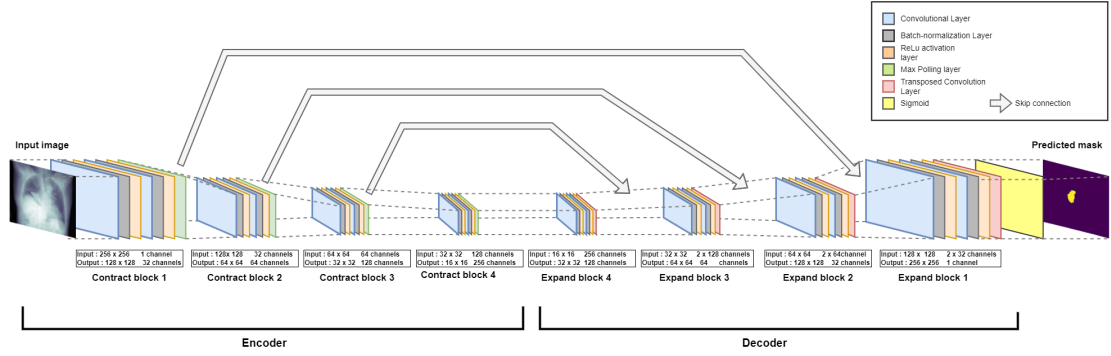


Figure 4.1: Visual representation of the topology of the U-net used in this work. This network is composed of 4 contract blocks, 4 expand blocks and 3 skip connections.

Convolutional layers

The convolutional layers of this network use zero-padding in order to output images of the same resolution as their input. Moreover, these convolutions use a kernel of size 5x5 instead of the 3x3 kernel of the classical U-net. The goal of this change is to slightly increase the scale and complexity of the features with a smaller increase in trainable weights than a deeper network would imply. Reducing the number of trainable weights would in theory reduce training and inference time which is a desirable propriety for this real-time application, as it results in a smaller delay between the acquisition of the images and the segmentation.

Batch-normalization layers

This network uses batch-normalization (BN) after each convolutional layer. Using BN in convolutional neural networks has been shown to often improve or at least not degrade the performance of the neural network, while at the same time reducing the standard deviation of the prediction scores [95]. This diminution in the variation of the prediction accuracy is particularly important in the case of a critical application such as medical image segmentation. Other normalization methods are also possible and can in some cases perform even better [95]. However, those methods are often slower than batch normalization due to their added complexity, which is an undesirable priority in the case of a real-time application such as this work.

Max pooling layers

This network uses a 2x2 max pooling layer at the end of each contract block. This layer reduces the resolution of the images by a factor of two in each dimension. This allows the following layers to aggregate multiple low-level features in order to construct more abstract feature maps and work at different scales.

Transposed convolution layers

The up-sampling of the decoder part of this network is operated by a transposed convolution. The concept of a transposed convolution is to do the opposite of a convolutional layer (but it is not a reverse convolution mathematically speaking) [96]. While the objective of a convolution is to aggregate the values of one patch of the image array, weighted by a kernel, the goal of a transposed convolution is to distribute the value of one element of the image array, to a patch of the output image, by weighting this distribution of a kernel. The transposed convolution operation is shown in Figure 4.2. This operation is in fact mathematically equivalent to a padded convolution with a transposed kernel as shown in Figure 4.3, hence the name transposed convolution [97].

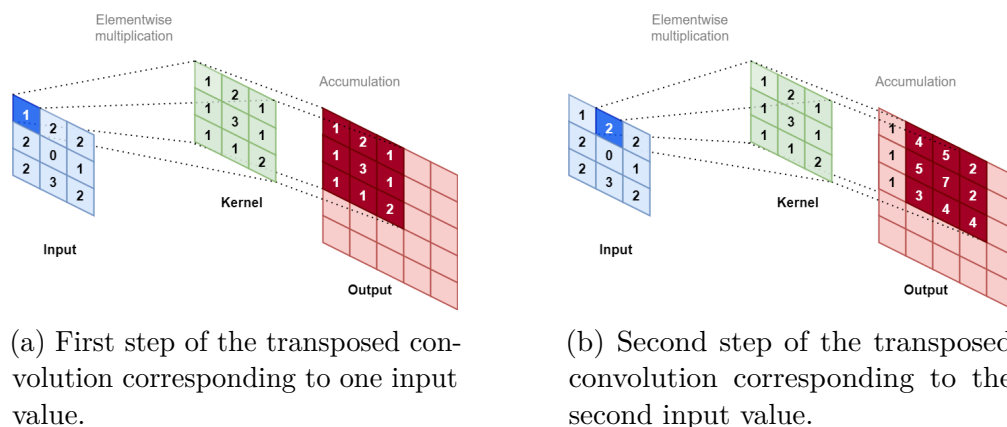


Figure 4.2: Illustration of the first two steps of a transposed convolution with a stride of one and a padding of two.

Moreover, the strides are applied in a different way than with a normal convolution. Strides are applied as zero-padding between the values of the original input array as shown in Figure 4.4. As a result, a greater stride results in a larger size of the output array, in opposition to the stride in convolutions which results in smaller arrays. In order to make the size of the output dimensions even, it is

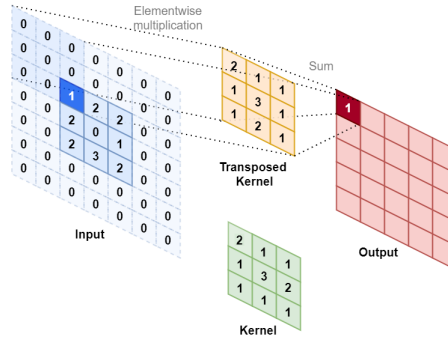


Figure 4.3: Alternative illustration of the first step of a transposed convolution with a stride of one and a zero-padding of two.

common to add zero-padding on one side of each dimension.

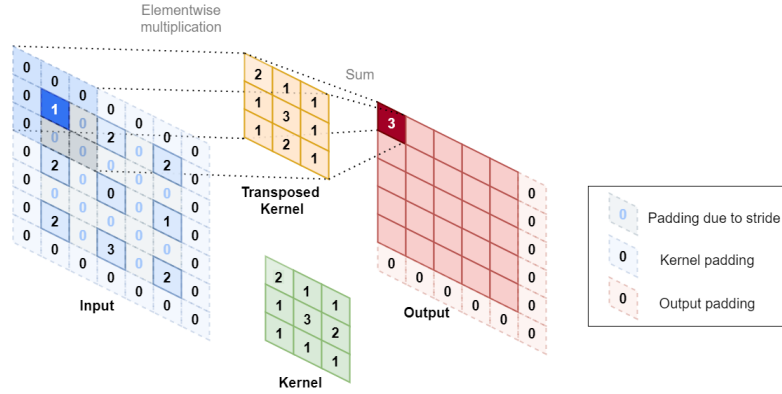


Figure 4.4: Example of a transposed convolution with a stride of 2, a padding of 1 and an output padding of 1. The output array is twice the size of the input array in each dimension.

The transposed convolutions used in this network use 3x3 kernels, with a stride of 2 and a kernel padding of 1, and an output padding of 1. As a result, these layers result in an up-sampling of 2 in each direction. As the number of expand blocks is equal to the number of contract blocks, and because the down-sampling of the max pooling layers is of equal magnitude as this up-sampling, the network reconstructs segmentation masks of the same resolution as the original image, allowing for easy matching of the two.

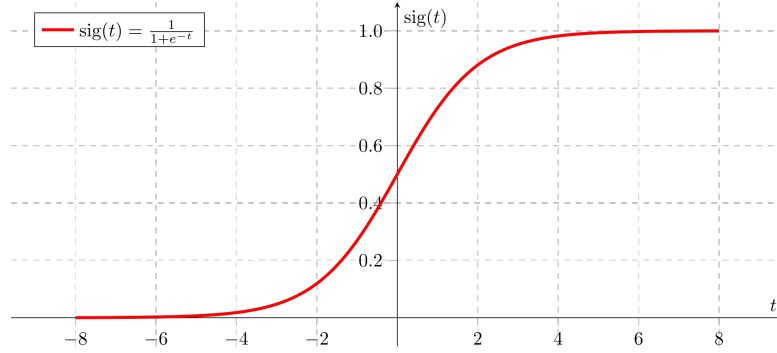


Figure 4.5: Response curve of the sigmoid function [98].

Sigmoid function

In order to transpose the output layer raw output into an actual prediction, we need an activation function. As this problem only includes one class, the "tumor" class, we can simply use a sigmoid. The sigmoid function takes an input ranging from $-\infty$ to $+\infty$ and converts it into an output with a range between 0 and 1. We can then use the sigmoid function as a measure of confidence that each specific pixel is a tumor or not. The response curve of the sigmoid function is shown in Figure 4.5.

4.2 Loss function

Choosing the right loss function is a very important part of designing a CNN for a specific problem (see section 2.5.1). In order to determine the one that corresponded the most to the problem of tumor segmentation, several different functions were implemented and compared.

For this problem, the main objective is to obtain a very high recall, as we want to make sure to irradiate every part of the tumor during the treatment.

In regards to the precision, it needs to be at least as high as the precision obtained in conventional methods, and should be maximized as much as possible without degrading the recall. A higher precision would result in smaller collateral damages for the proton therapy treatment.

4.2.1 Binary Cross-Entropy loss

Cross entropy loss, also known as logistic loss or log-loss, is defined as the negative log-likelihood of a logistic model that returns y_pred probabilities for its training data y_true [99]. The log loss is only defined for two or more labels. Binary cross entropy (BCE) loss is cross entropy loss applied to a binary classification problem. Its mathematical definition is given by the equation 4.1. BCE is one of the most commonly used loss functions in binary classification problems.

$$BCE_{loss}(p_t) = -\sum_{n=0}^N \log(p_{t,n}). \quad (4.1a)$$

$$p_{t,n} = \begin{cases} p_n, & \text{if } y_n = 1 \\ 1 - p_n, & \text{otherwise} \end{cases} \quad (4.1b)$$

Where y_n is the binary indicator (0 or 1) of the class of observation n , $p_n = Pr(y_n = 1)$ is a probabilistic estimate of the class of observation n .

4.2.2 Binary Cross-Entropy loss with cropped output

When applying BCE to our dataset, we noticed that in a few rare cases training the network with BCE can result in segmentation artifacts on the borders of the image as shown in Figure 4.6. As none of the images in the dataset include tumors on the borders of the images, because tumors can only be located inside the body of the patient which is centered on the image, we can simply crop a margin of 10 pixels along the border of the output mask and replace it with a zero-padding in order to avoid solve this problem. This extension is therefore not a different loss function but an adaptation of the network designed to make the network more stable with a BCE loss function.

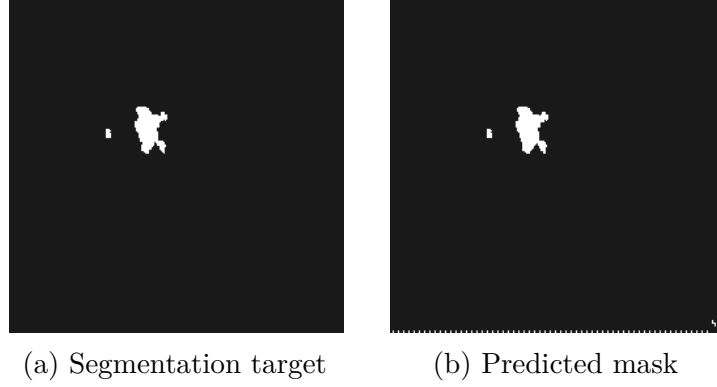


Figure 4.6: Comparison between the segmentation target and the predicted mask with a network trained with BCE loss. Artifacts appears on the bottom edge of the prediction.

4.2.3 Binary Dice Loss

Dice loss is a loss derived from the Sørensen-Dice coefficient (see section 2.5.4). It is defined as $1 - DICE$, where $DICE$ is a smooth extension of the Dice coefficient. Dice loss is defined in equation 4.2

$$DICE_{LOSS} = 1 - DICE \quad (4.2a)$$

$$DICE = \frac{2 * \sum (Mask_{predicted} * Mask_{target}) + smooth}{\sum Mask_{predicted} + \sum Mask_{target} + smooth} \quad (4.2b)$$

4.2.4 Distance Penalized Dice-Loss

Distance penalized dice loss is an extension of the conventional Dice loss inspired by Caliva F, Iriondo C. et al [100] distance penalized cross-entropy loss initially created for bone segmentation on 3D MRI. Their loss function results from their observation that most segmentation errors were located along the edges of the segmented masks, and therefore adds a penalty term to binary cross-entropy which guides the focus of the network's focus toward the edge of the region to segment. In the same way, distance penalized dice loss adds a penalty weighting to Dice loss based on a distance map from the segmentation mask boundaries. This modification is supposed to make the segmentation mask more robust along the edges of complex shapes, and therefore extract more precise boundaries in the resulting detection.

The procedure computes an error-penalizing distance map, Φ , by computing the distance transform on the segmentation maps and reversing it. A similar distance

map is also computed for the opposite binary mask, in order to compute the overall distance from the segmentation edge, on the inside of the mask as well as inside it. The resulting error-penalizing distance map is shown in Figure 4.7, it assigns to each pixel of the image a value that tends toward 2 on the edge of the segmentation mask and tends toward 1 in the center of the mask on the edges of the image. This distance map is then used to compute a weighted Dice coefficient as defined in equation 4.3.

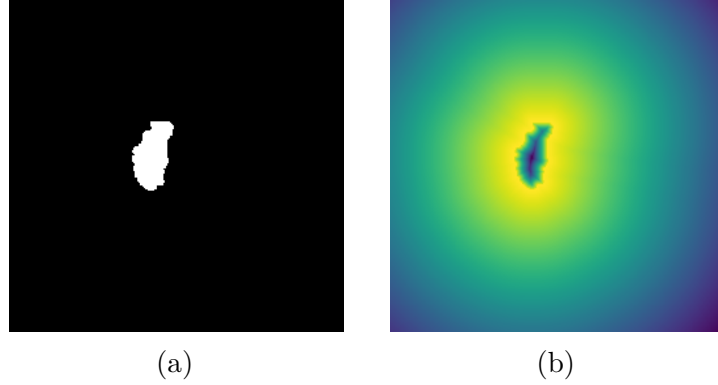


Figure 4.7: Binary segmentation mask of the tumor of Patient 1 (a) and the associated error-penalizing distance map (b). The darkest parts of the distance map have a pixel value of 1 and the lightest parts have a pixel value of 2.

$$Dice_{DP_LOSS} = 1 - Dice_{DP} \quad (4.3a)$$

$$Dice_{DP} = \frac{2 * \sum (Mask_{predicted} * \Phi * Mask_{target} * \Phi) + smooth}{\sum Mask_{predicted} * \Phi + \sum Mask_{target} * \Phi + smooth} \quad (4.3b)$$

$$\Phi = (max(d_{out}) - d_{out}) * \frac{Mask_{target}}{max(d_{out})} + (max(d_{in}) - d_{in}) * \frac{1 - Mask_{target}}{max(d_{in})} + 1 \quad (4.3c)$$

$$d_{out} = edt(Mask_{target}) \quad (4.3d)$$

$$d_{in} = edt(1 - Mask_{target}) \quad (4.3e)$$

Where *edt* is the exact euclidean distance transform and *max* returns the largest element of a two dimensional array.

4.2.5 Focal Loss

Focal loss is an extension of cross-entropy loss created by Lin et al.[101] to manage heavy class imbalance better and focus the learning process on hard examples. This loss function was initially developed for dense networks and is applied to a CNN in this work.

Cross entropy loss is very common and can be very powerful, however, it is not always ideal when dealing with large class imbalance, which is usually the case in tumor segmentation. Indeed, tumors only represent a small part of the output, and networks trained with BCE are sometimes susceptible to prediction collapse and only predict the class corresponding to the background. A common method for reducing class imbalance is to add a weighting factor $\alpha \in [0, 1]$ for class 1 and $1 - \alpha$ for class 0. This class weighting balances the importance of positive and negative examples, however, it is not sufficient. Indeed, this weighting does not differentiate between hard and easy examples, and the data is mostly composed of easily classified negative examples (everything that is not a tumor) which dominate the loss and the gradient. Instead, focal loss down-weight easy examples to focus on hard negative examples.

This focal loss uses both the weighting factor α and a tunable focusing parameter $\gamma \geq 0$. The γ parameter smoothly adjusts the rate at which easy examples are down-weighted. Focal loss with $\gamma = 0$ is equivalent to BCE. A greater γ reduces the contribution of examples classified with a high probability. With a γ of 2, an example classified with a probability $p_t = 0.9$ would have $100\times$ lower loss compared with BCE and with $p_t \approx 0.968$ it would have $1000\times$ lower loss. The mathematical definition of class balanced focal loss is described in equation 4.4.

$$Focal_{loss}(p_t) = -\sum_{n=0}^N \alpha_t (1 - p_{t,n})^\gamma \log(p_{t,n}). \quad (4.4a)$$

$$p_{t,n} = \begin{cases} p_n, & \text{if } y_n = 1 \\ 1 - p_n, & \text{otherwise} \end{cases} \quad (4.4b)$$

$$\alpha_t = \begin{cases} \alpha, & \text{if } y_n = 1 \\ 1 - \alpha, & \text{otherwise} \end{cases} \quad (4.4c)$$

Where y_n is the binary indicator (0 or 1) of the class of observation n , $p_n = Pr(y_n = 1)$ is a probabilistic estimate of the class of observation n , α is the class weighting factor and γ is a tunable focusing parameter.

4.2.6 Binary Hausdorff loss

The Hausdorff Distance (HD) is an evaluation metric used in evaluating medical image segmentation methods. Karimi et al. [102] developed HD based loss functions in order to train CNNs directly with this metric. They proposed three different methods to estimate HD from the segmentation probability map produced by a CNN. The first method uses distance transform of the segmentation boundary, the second uses morphological erosion, and the third uses circular/spherical convolution kernels. The method used in this work is the first one based on distance transform. The mathematical definition of HD loss is defined in equation 4.5.

$$HD_{LOSS} = mean(error_{pred} * distance) \quad (4.5a)$$

$$distance = d_{pred}^\alpha + d_{target}^\alpha \quad (4.5b)$$

$$error_{pred} = (Mask_{pred} - Mask_{target})^2 \quad (4.5c)$$

$$d_{target} = edt(Mask_{target}) + edt(1 - Mask_{target}) \quad (4.5d)$$

$$d_{pred} = edt(Mask_{pred}) + edt(1 - Mask_{pred}) \quad (4.5e)$$

Where $\alpha > 0$ is a distance penalizing factor, *edt* is the exact euclidean distance transform and *mean* returns the average value of an array. In this work α is fixed to 2 as advised in the original article of Karimi et al.

4.3 Methodology

4.3.1 Derivation of the optimal parameters

In order to create the topology of the network, the primitive versions of the network are first evaluated with a 5-fold cross-validation over the synthetic dataset, with a smaller resolution of 128x128 in order to reduce the training time. Once a version of the network is deemed performant enough and provides reliable predictions for the synthetic dataset, the evaluation is switched to the datasets derived from real 4D CT of lung cancer patients consisting of 256x256 images (see section 3.4).

To determine the optimal hyper-parameters for the network corresponding to each loss function, training is performed in a Grid Search (GS) manner for each loss function. This means that each network is trained with every possible combination of hyper-parameters in the range provided to the GS. This search is not extensive for time-constraints reasons but provides an overview of the possible capabilities of the network with different meta-parameters. This GS is effectuated on a single

train-test split of 80% for training and 20% for test, of dataset 2 described in section 3.4 and on Patient 1 exclusively. This is done in order to avoid overfitting of these hyper-parameters to the validation data. The resulting hyper-parameters for each loss function described in Table 4.1.

Table 4.1: Hyper-parameters of the network used for each loss function.

Loss function	Learning rate	Epochs	Batch size
BCE loss	0.0005	200	1
BCE loss with cropping	0.0002	150	1
Dice loss	0.0005	200	1
Distance penalized Dice loss	0.0005	200	1
Focal loss	0.0005	120	1
Binary Hausdorff loss	0.002	150	1

Furthermore, the tunable parameters used for each adjustable loss function are the following:

- **Dice loss** : $smooth = 1$
- **Distance penalized Dice loss** : $smooth = 10^{-5}$
- **Focal loss** : $\gamma = 2$, $smooth = 10^{-5}$, $\alpha = 20$
- **Binary Hausdorff loss** : $\alpha = 2$

4.3.2 Loss function comparison

Once the optimal hyper-parameters of the network are determined for each loss function, we need to determine which loss function performs the best for this network on the specific task of segmentation in the context of proton therapy guidance.

Determining the optimal loss function is a critical part of the design of a neural network. The loss function is the part of the networks which guides the training in the right direction. Therefore, we need to determine which loss function leads to the most desirable network for our application. In the case of this work, the resulting neural network should be reliable and provide consistent results, and it should have an almost perfect recall as well as a high precision.

The different evaluation functions were compared on dataset 1, composed of 300 images with a phase uniformly distributed and an amplitude uniformly distributed

between 0.7 and 1.2 to cover the whole range of the respiration cycle in normal circumstances. This evaluation is repeated independently for each of the 5 patients to evaluate the consistency of the segmentation.

This evaluation is done with a 5-fold cross-validation and the validation set is used as the test set in order to compare the different loss functions. A separate test set is not used as it would be too time-consuming and the objective of this evaluation is not to derive the most precise approximation of the real performances of the network, but only to compare the relative performance of each loss function. Furthermore, the optimal parameters have been derived by a GS on dataset 2 of patient 1, a completely independent dataset that is never used in the actual evaluation of the network, and therefore the chosen meta-parameters have no bias related to this cross-validation.

The metrics used in this test are the average Dice index, the average Jaccard index, the worst and best Jaccard index, the intra-fold Jaccard index variance, the average precision, and the average recall.

4.3.3 Robustness evaluation

Robustness to noise

To evaluate the robustness of the network to noise, this test constructs multiple variations of the same dataset, each augmented with different levels of Gaussian noise, and analyzes its effects on both training and evaluation. The base dataset used for training is dataset 1 (300 images, uniform amplitude distribution, ranging from 0.7 to 1.2) and the base dataset used for evaluation is dataset 4 (300 images, Gaussian amplitude distribution, ranging from 0.8 to 1.4).

First, different levels of Gaussian noise are added to the evaluation set and the network is only trained on the training set without noise. The performance of this network is then evaluated on the evaluation set modified by this noise in order to determine the robustness of the network to different levels of noise, without any specific precautions. The noise added to the images follows a Gaussian distribution with zero mean and standard deviations of respectively 0.5%, 1%, 2%, 5%, and 10% of the maximum intensity of the image. The effects of these levels of noise on a fluoroscopic image are shown in Figure 4.8

Secondly, the network is trained on the same dataset but augmented with random Gaussian noise of zero mean and standard deviation of 1% the maximum

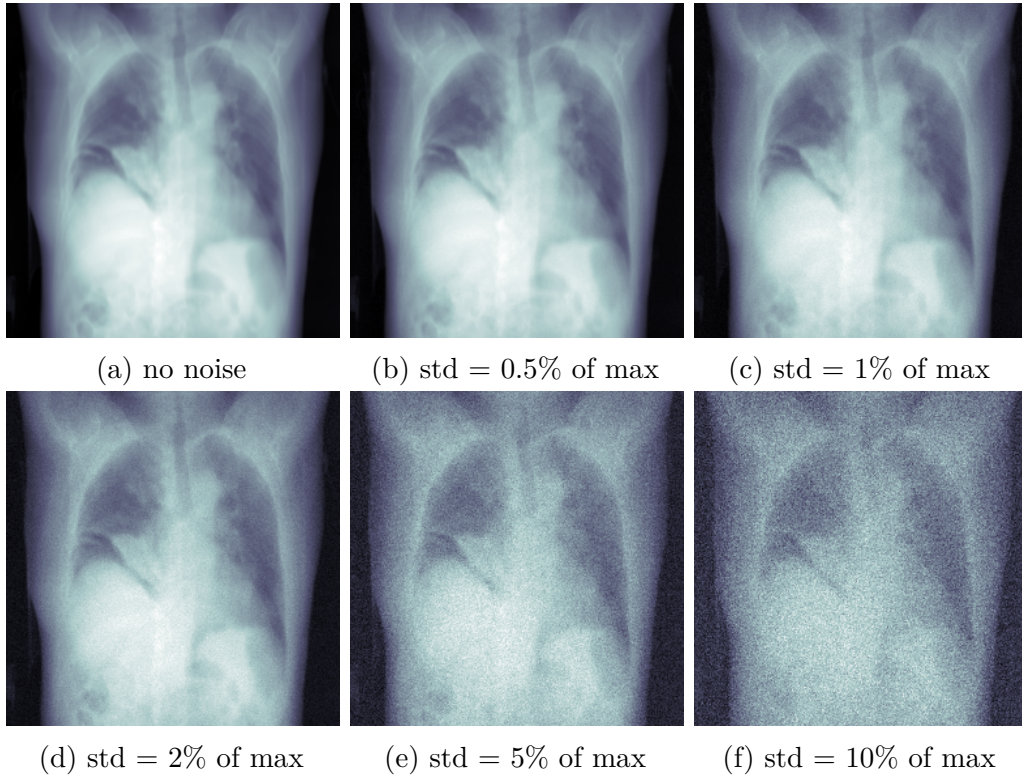


Figure 4.8: A Fluoroscopic image at the different levels Gaussian noise tested. Each level of Gaussian noise has a standard deviation which corresponds to a specified percentage of the maximum intensity of the image.

intensity on every image. The performance of the network is again evaluated on the same noisy dataset.

The first test is designed to evaluate the effect of noise on the segmentation performance of the network, and the acceptable levels of noise on the input images. The second test is designed to evaluate whether augmenting the training set with random noise is beneficial or detrimental to the segmentation performance and the robustness to noise. These tests are repeated on all five patients.

The metrics used in this test are the Dice index, the Jaccard index, the Jaccard index variance, the precision, and the recall.

Robustness to deformations

During the treatment phase, a patient could potentially have a respiration cycle that differs from his respiration cycle during the data acquisition phase, with a respiration amplitude larger than the observed ones. This could be caused by a variety of factors such as stress, coughing or hiccups for example. In order to test the network’s robustness to unexpectedly deep breaths, two different cases are analyzed.

First, a network is trained on ordinary respiratory movements, with dataset 1, composed of 300 images with a phase uniformly distributed and a Gaussian amplitude distribution, ranging from 0.7 to 1.2. This network is then evaluated on dataset 4, composed of 300 images with a phase uniformly distributed and an amplitude generated with a Gaussian distribution between 0.6 and 1.3. The evaluation on this dataset replicates testing on a typical radiotherapy session, with a distribution of amplitude close to the one expected during treatment.

Secondly, the same network, trained on the same dataset, is evaluated on dataset 5, composed of 300 images with a phase uniformly distributed and an amplitude generated with a Gaussian distribution between 0.5 and 1.5. Evaluation of the performances on this dataset replicates testing on a radiotherapy session where the patient has slightly abnormal respiration. The variation in amplitude of the respiratory movements is slightly larger than the previous evaluation and, most importantly, from the training data.

Comparing the different metrics of the segmentation performance in these two cases should provide a good overview of the capabilities of the network to work with a respiratory amplitude slightly out of the range of the training data. These tests are repeated on patients 1 and 4.

The metrics used in this test are the Dice index, the Jaccard index, the precision, and the recall.

.

Robustness to rotations and scaling

To evaluate the robustness of the network in regard to slight changes in alignment and scaling, the network is trained on an unmodified dataset 1 (300 images, uniform amplitude distribution, ranging from 0.7 to 1.2) and then evaluated on both an unmodified dataset 4 (300 images, Gaussian amplitude distribution, ranging from 0.6 to 1.3)) and a modified version of dataset 4 composed of rotated and re-scaled images.

The network is trained on dataset 1 (uniform amplitude distribution, ranging from 0.7 to 1.2) and then evaluated on two versions of the dataset 4 (Gaussian amplitude distribution, ranging from 0.6 to 1.3.). The first version of the evaluation dataset is not modified, and the second version is augmented with random rotations of each image, with a uniform distribution of the angle between -5° and $+5^\circ$. These images are cropped in order to avoid empty pixels along the edge of the images and reshaped to the initial resolution using bilinear interpolation. An example of this transformation is shown in Figure 4.9. This test is repeated for patients 1 and 4.

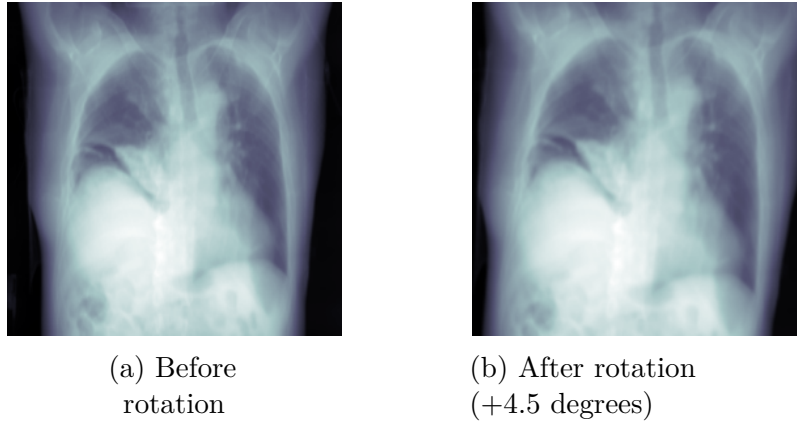


Figure 4.9: Example of a fluoroscopic image before and after rotation. The rotated image has been re-framed and resized with a bi-linear interpolation in order to fit the initial resolution without black borders caused by the rotation.

The metrics used in this test are the Dice index, the Jaccard index, the variance of the Jaccard index, the precision, and the recall.

4.3.4 Comparison against static treatment planning

This evaluation compares the proposed algorithm with the most common method of treatment planning, which targets the whole range of motion of the tumor to maximize recall (see section 1.2.2). This test compares the two methods on one projection angle of the robust planning phase which would typically involve multiple different projection angles.

In this test, the two methods are compared on the Dice index, The variance of the Dice index, the precision, and the recall in order to determine if the proposed

method offers advantages compared to the most commonly applied method. These tests are repeated for patients 1, 4 and 5 because these patients presented the widest tumor movements.

4.3.5 Inference time evaluation

This test evaluates the end-to-end segmentation latency for one single image with the proposed network. The latency measured includes copying the data to the GPU, the inference time through the network, the activation, and copying the data into the main memory. The test is repeated 100 times in order to determine the average latency as well as its distribution and variations. The GPU is forced to synchronize with the CPU at every evaluation as calls to the GPU library (CUDA) are asynchronous. The goal of this test is to evaluate whether the network can realistically be used for real-time segmentation or if its delay is too high for such an application with current hardware. The test is done on a Linux server with a Intel i9-10900X processor, 64go of memory, and an Nvidia RTX 6000 GPU. The time is evaluated based on its average, its distribution in quantiles, and its maximum and minimum value.

4.4 Summary

In this chapter, we have seen the detailed topology of the network used in this work. Then, the different candidate loss functions proposed for the training of this network are detailed and their particularities are explained. Finally, the evaluation methodology of this work is explained in detail.

Chapter 5

Results and discussion

5.1 Synthetic data in 128x128 resolution

Table 5.1 shows the Dice index, Jaccard index, precision, and recall of the network trained on 128x128 simulated fluoroscopic images. The system evaluated in this first section has not been completely refined as fine-tuning the network on an artificial dataset would not necessarily result in better performances on real data. These results are therefore presented as a first proof of concept with sub-optimal meta-parameters before moving on to the real datasets. This test shows that the network is able to predict the segmentation masks even with obstructed low-resolution images. Some refinements need to be done mostly on the recall aspect of the performance. At this stage, the value of the recall falls as low as 0.63988 in the fourth fold. Maximizing the recall is one of the main focuses of the following section that compares different loss functions. Nevertheless, we can observe that the precision of the network is already good, and ranges between 0.88 and 0.97. This discrepancy between precision and recall could be explained by the large class imbalance of the dataset, as the tumors represent only a small fraction of the segmentation masks.

Table 5.1: Dice index, Jaccard index, precision and recall of the proposed network trained on 128x128 simulated fluoroscopic images on a 5-fold cross validation.

	Fold n°1	Fold n°2	Fold n°3	Fold n°4	Fold n°5	5-Fold average
Dice index	0,896424	0,940579	0,928076	0,769533	0,954868	0,897896
Jaccard index	0,81229	0,887823	0,865803	0,625399	0,913635	0,82099
Precision	0,958135	0,97004	0,955243	0,879534	0,948368	0,942264
Recall	0,842181	0,912854	0,902411	0,683988	0,961458	0,860578

Now that this level of performance is achieved on a synthetic dataset, the rest of the evaluation is focused on the real dataset consisting of fluoroscopic images

derived from real 4D CT of lung tumor patients at the full 256x256 resolution with the optimized network.

5.2 Loss functions comparison

In Figure 5.1, the distribution of the Jaccard coefficients between the segmentation masks generated by the proposed neural network and the real segmentation masks is computed over the five folds for each of the five patients. Tables 5.2–5.6 present the average Dice index, the average Jaccard index, the worst and best Jaccard index, the intra-fold Jaccard index variance, the average precision, and the average recall metric of every loss for each Patient. The following observations can be done based on these metrics:

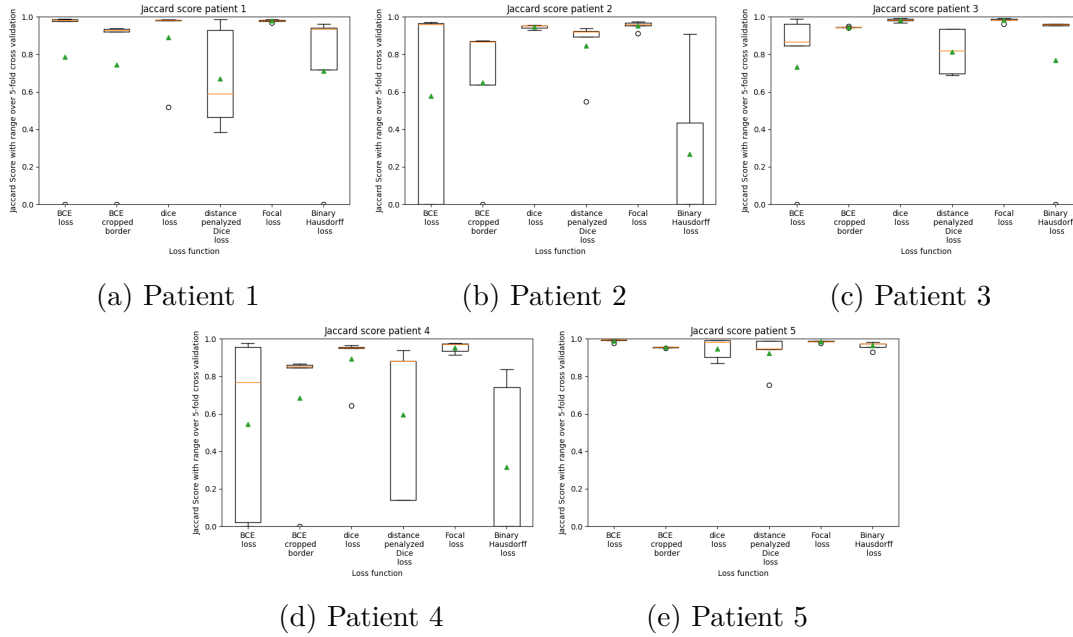


Figure 5.1: Distribution of the Jaccard loss of each fold for every loss function. The yellow bars represent the median values and the green triangles correspond to the average values

- A network trained with the same loss function can perform a great segmentation on one patient but perform terribly with another patient. This is particularly the case of networks trained with the two variations of BCE. On the test performed with the dataset of patient 5, all the loss functions lead to

a great segmentation, with an average Jaccard score greater than 0.92, and the two BCE loss based networks obtain a Jaccard index of 0.99 and 0.95 as shown in Figure 5.1(e) and Table 5.6. However, on the dataset of the second patient, the two networks trained with BCE loss functions perform among the worst segmentations as shown on Figure 5.1(b). Table 5.3 reports their respective Jaccard scores as 0.58 for the normal version and 0.65 for the version augmented with border cropping.

- In opposition with the previous observation, other loss functions lead to a very stable segmentation across all the patients. The networks trained with a focal loss function perform well with every one of the five patients, with a minimum average Jaccard score of 0.95 as shown in Table 5.5. Networks trained with a Dice loss are also relatively stable for every patient, with the exception of two outliers of patient 1 and Patient 4, with a worst-case Jaccard index of 0.519 and 0.64 respectively but median values on par with focal loss.
- Some loss functions can lead to good average segmentation performance, but sub-par worst-case segmentation performance. This is the case of the networks trained with dice loss. As shown in Figure 5.1, this loss function often leads to great average segmentation performances. It leads to a minimum average Jaccard score of 0.93, 0.96 and 0.869 for patients 2, 3 and 5 respectively over the 5-folds as shown in Table 5.2 - 5.2. However, for the evaluations performed with patient 1 and 4, the range of this Jaccard score is quite large and goes as low as 0.52 and 0.64 respectively as shown in Table 5.2 and Table 5.5. Other losses such as the two BCE and the Binary Hausdorff loss can even lead to a collapse resulting in the network predicting only zeros and leading to a null Jaccard score. Focal loss, on the other hand, is very consistent in performance segmentation over each fold, as apparent in Figure 5.1 where its range is always really small.
- Most of the selected loss functions manage the class imbalance very well, as the tumor pixels represent approximately 2% of the whole segmentation masks, and despite this fact, all of the loss functions result in segmentation with similar precision and recall scores. However, the function used to compute the recall and precision goes to zero for both when the resulting segmentation mask is empty, and the segmentation collapse described above could very well be caused by the class imbalance. Furthermore, for this application to radiology, we want to obtain a recall as high as possible, in order to make sure to be able to target every cancerous cell. Tables 5.2–5.6

show that Focal and Dice loss are the best at fulfilling that goal, as both of their recall scores stay higher than 0.99 for any patient.

- Finally, some of the networks provide consistent prediction performance across the different samples of the same fold, while others vary significantly on the different samples. Tables 5.2–5.6 show that networks trained with a Focal or Dice loss function have a very low intra-fold Jaccard index variance, while networks trained with BCE or Binary Hausdorff loss have significant intra-fold variance. This intra-fold variance is an indicator of the consistency of the predictions, as a low variance indicates consistent performances.

These observations all indicate that Focal loss is the best candidate for this task. It results in high segmentation performance, with a Jaccard and Dice index greater than almost all the models trained with different loss functions, or at least very close to the best one. Indeed, the only time Focal loss is beaten in terms of Jaccard index is with Patient 5, where its Jaccard score is only beaten by 0.003 by the network trained with BCE loss as shown in Table 5.6. The average Jaccard index of the networks trained with focal loss over every fold of every patient is **0,965**. Furthermore, its performance is very consistent, across the different patients as well as across the different folds or across the samples of the same fold. This consistency is very important in a domain such as medicine where unexpected results can result in bad consequences. Finally, networks trained with focal loss have an almost perfect recall, consistently higher than 0.99. This high recall means that this segmentation does not need a large security margin in order to make sure to target the whole tumor with radiations. For this reason, the loss used in the rest of this work is the focal loss with the same parameters as in this test.

A visual representation of the quality of a segmentation mask generated with the network trained with focal loss on patient 1 is shown in Figure 5.2. This sample has been selected because it represents well the average performance of the network with focal loss.

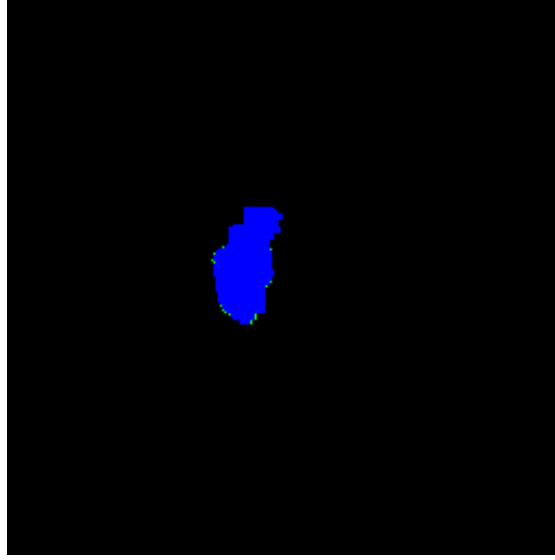


Figure 5.2: Visual representation of the precision and recall of a segmentation mask generated with the network trained with focal loss on patient 1. The intersection between the real mask and the predicted mask is displayed in blue, the pixels of the predicted mask not present on the real mask are shown in green and the pixels present on the real mask not predicted by the network are shown in red. As with the majority of the samples, there are no red pixels as the network has an almost perfect recall.

Table 5.2: Patient 1 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average precision and recall metric of every loss.

	BCE loss	BCE loss with crop	Dice loss	DP Dice loss	Focal loss	Binary Hausdorff loss
Average Dice index	0.793287	0.771482	0.930079	0.777843	0.989341	0.749479
Average Jaccard index	0.786741	0.745036	0.89073	0.671315	0.978967	0.711751
Jaccard index (worst fold)	0	0	0.519396	0.383194	0.967169	0
Jaccard index (best fold)	0.988221	0.938708	0.986543	0.986646	0.986478	0.963258
Average intra-fold Jaccard index variance	0.154754	0.138805	0.034475	0.059722	4.37E-05	0.134517
Average Precision	0.794324	0.778988	0.898572	0.684828	0.979501	0.78628
Average Recall	0.79234	0.765301	0.991219	0.983503	0.999439	0.724986

Table 5.3: Patient 2 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.

	BCE loss	BCE loss with crop	Dice loss	DP Dice loss	Focal loss	Binary Hausdorff loss
Average Dice index	0.58939	0.713848	0.972226	0.907294	0.975781	0.31156
Average Jaccard index	0.579235	0.64935	0.946641	0.844857	0.953199	0.268771
Jaccard index (worst fold)	0	0	0.930415	0.547795	0.910872	0
Jaccard index (best fold)	0.971521	0.874399	0.956707	0.938724	0.974881	0.909885
Average intra-fold Jaccard index variance	0.223691	0.113552	9.16E-05	0.022287	0.000506	0.131006
Average Precision	0.590191	0.696587	0.975256	0.891703	0.956392	0.386866
Average Recall	0.588672	0.750112	0.969919	0.950501	0.996734	0.2739

Table 5.4: Patient 3 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.

	BCE loss	BCE loss with crop	Dice loss	DP Dice loss	Focal loss	Binary Hausdorff loss
Average Dice index	0.759407	0.97157	0.990948	0.893966	0.990991	0.78344
Average Jaccard index	0.732946	0.944731	0.98231	0.814753	0.982244	0.767642
Jaccard index (worst fold)	0	0.940781	0.96938	0.687287	0.961905	0
Jaccard index (best fold)	0.988525	0.949301	0.99065	0.934779	0.991926	0.963762
Average intra-fold Jaccard index variance	0.137245	7.8E-06	5.64E-05	0.01172	0.000115	0.147332
Average Precision	0.759323	0.977222	0.989473	0.838232	0.982844	0.785095
Average Recall	0.773351	0.966302	0.992672	0.975274	0.999374	0.782041

Table 5.5: Patient 4 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.

	BCE loss	BCE loss with crop	Dice loss	DP Dice loss	Focal loss	Binary Hausdorff loss
Average Dice index	0.575242	0.73799	0.938642	0.666816	0.976321	0.352764
Average Jaccard index	0.5448	0.685127	0.89419	0.596583	0.954685	0.316056
Jaccard index (worst fold)	0	0	0.643019	0.139793	0.914792	0
Jaccard index (best fold)	0.976536	0.86658	0.966283	0.936975	0.978248	0.838397
Average intra-fold index variance	0.195744	0.117399	0.015799	0.139029	0.000632	0.150768
Average Precision	0.746523	0.750556	0.914929	0.63015	0.960529	0.372434
Average Recall	0.59505	0.729918	0.977909	0.96394	0.993214	0.335985

Table 5.6: Patient 5 : Average Dice index, average Jaccard index, worst Jaccard index, best Jaccard index, intra-fold Jaccard index variance, average Precision and Recall metric of every loss.

	BCE loss	BCE loss with crop	Dice loss	DP Dice loss	Focal loss	Binary Hausdorff loss
Average Dice index	0.995114	0.977031	0.972569	0.958399	0.992705	0.981575
Average Jaccard index	0.990311	0.955115	0.948144	0.924394	0.985541	0.964084
Jaccard index (worst fold)	0.976529	0.951214	0.869961	0.753056	0.978327	0.930633
Jaccard index (best fold)	0.995171	0.957413	0.992206	0.989158	0.989662	0.984565
Average intra-fold index variance	4.81E-05	4.66E-06	0.002632	0.007703	1.48E-05	0.000358
Average Precision	0.995354	0.978753	0.95176	0.944331	0.985974	0.990046
Average Recall	0.99488	0.975404	0.99604	0.979566	0.999554	0.973512

5.3 Robustness evaluation

5.3.1 Robustness to noise

Figure 5.3 shows the evolution of the Jaccard index between the real segmentation masks and the segmentation masks predicted by the same network, for varying levels of Gaussian noise in the input images. This network is trained with a training set that has not been augmented by random noise. Tables A.1–A.5 display the Dice index, the Jaccard index, the Jaccard index variance, and the average precision and recall. As we can see in Figure 5.3, the effect of a particular noise level is not consistent with the segmentation performance deterioration, but a general tendency can be observed.

- A noise level with a variance smaller than 1% of the maximal intensity of the image has little to no effect on the segmentation performance. We can see that each Jaccard index stays higher than 0.9 with minimal variations compared to the validation set with no noise.
- A noise level with a variance around 2% of the maximal intensity of the image can in some cases be sufficient to degrade the segmentation performance significantly, with the Jaccard index of the segmentation over the first and second patient falling drastically under 0.5. For the segmentations of patient 4, 5 and 6 however, the performances remain perfectly stable and well above a Jaccard index of 0.9.
- A noise level with a variance around 10% of the maximal intensity is enough to lower the Jaccard index of the segmentation under 0.8 for every patient, with the segmentation performance going almost to zero for patients 1 and 2 as shown in Figure 5.3.
- We can observe that the robustness to Gaussian noise varies widely from one patient to another. This could be explained by a variety of factors such as the presence of natural noise (not added by us) in some datasets or a difference in the image intensity distribution since the added noise is dependent on the maximum intensity.

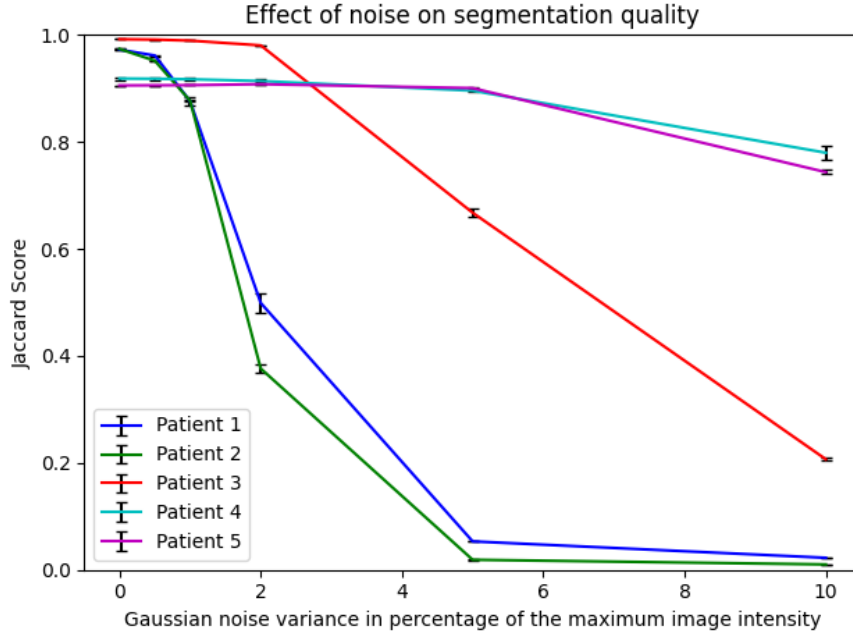


Figure 5.3: Evolution of the average Jaccard index of the segmentation for different levels of Gaussian noise in the validation dataset. The network has not been trained on noisy data.

Figure 5.4 shows the evolution of the Jaccard index between the real segmentation masks and the segmentation masks predicted by the same network, for varying levels of Gaussian noise in the input images. In this case the network is trained with a training set that has been augmented by a random Gaussian noise with a mean of zero and a variance of 1% of the maximal intensity.

We can note that the segmentation performance of the network is more robust to noise for almost all patients, except for a small diminution for patient 5 as shown in Figure 5.4. The general tendencies that can be observed are the following:

- A noise level with a variance smaller than 2% of the maximal intensity of the image has little to no effect on the segmentation performance. The average Jaccard index of every segmentation stays almost constant and higher than 0.9 for all patients.
- A noise level with a variance around 5% of the maximal intensity of the image can be sufficient to degrade the segmentation performance in a noticeable way, but the Jaccard index of the segmentation stays higher than 0.8 for all patients.

- A noise level with a variance around 10% of the maximal intensity is enough to lower the Jaccard index of the segmentation under 0.8 for every patient except for the fourth patient. However, even with this level of noise, none of the Jaccard scores of the segmentation goes to zero, which is a drastic improvement compared to the networks trained with no noise in the input.
- Overall, the effect of noise augmentation during the training phase on the performance of the network in noisy conditions has been overwhelmingly positive. This addition increases the robustness of the network for every patient, and has an even greater effect on patients with whom the network had an initially poor robustness to noise. As a result, the robustness of every segmentation to noise is largely more homogeneous than before the augmentation of the training dataset.

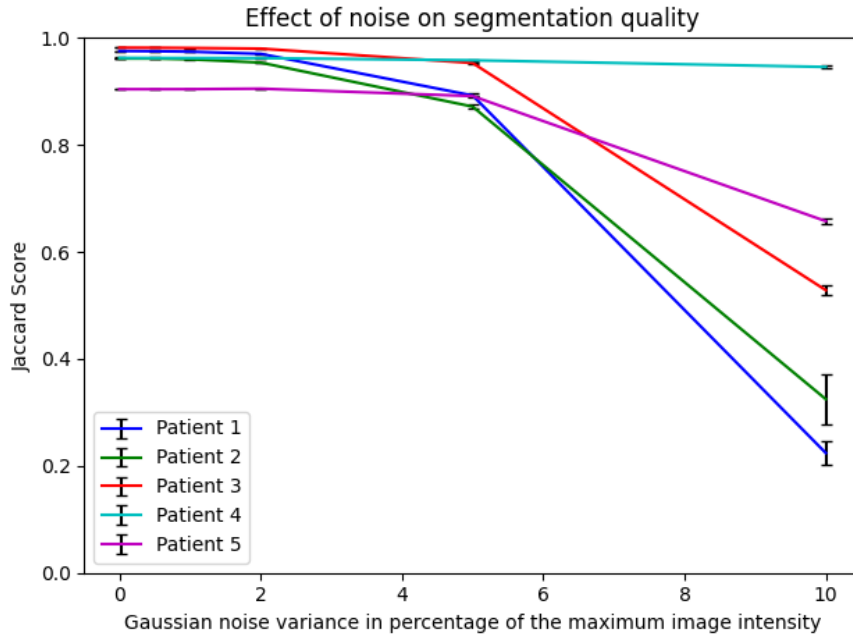


Figure 5.4: Evolution of the average Jaccard index of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.

Because of these observations, it would be wise to include a low-level of random Gaussian noise in the training phase of the network as a default option. This would reduce the impact of noise in the fluoroscopic imaging on the performance

of the network and lead to better generalization capabilities. This amelioration comes with no cost in terms of performance on a noise-free dataset according to this test. In further work, it would be interesting to test different levels of noise in the training dataset, instead of single fixed value, similarly to what was tested for the validation dataset.

5.3.2 Robustness to deformations

In Tables 5.7 and 5.8, the Dice index, the Jaccard index, and the average precision and recall are displayed over two different tested datasets for patients 1 and 4 respectively. Dataset 4 corresponds to data generated with an amplitude following a Gaussian distribution between 0.6 and 1.3, representing a normal respiration cycle. Dataset 5 corresponds to data generated with an amplitude following a Gaussian distribution between 0.5 and 1.5, representing a deeper respiration cycle than the one seen during training.

On these two tests, the network performs slightly better with the shallower respiration range but the difference is really small and is not statistically significant. The diminution in terms of Jaccard index is of only 0.62% for patient 1 and 0.65% for patient 4. Furthermore, the average recall stays higher than 0.99 for both patients, and the tumor is therefore almost entirely detected despite the larger amplitude variations. The small variations in terms of precision are less concerning in an application such as radiotherapy where the main concern is to target the whole tumor with the radiations. Moreover, the average precision on the datasets with larger amplitudes is still of 0.973 for patient 1 and 0.97 for patient 4, which are still really high results for the detection of a mobile tumor.

This indicates that the network is robust against small variations in respiratory amplitude. However, the deformations tested here all result from the openTPS tool. It might be possible that this tool is not able to generate every possible deformation due to respiratory movements. Furthermore, the difference between the two tested amplitude ranges is fairly minor, in order not to push the limits of the deformation tool of openTPS which might not result in realistic deformations with very high amplitudes.

Table 5.7: Patient 1 : Average Dice index, average Jaccard index, average Precision and Recall metric on evaluation sets following a different deformation amplitude distribution range.

	Gaussian distribution of amplitude between 0.6 and 1.3	Gaussian distribution of amplitude between 0.5 and 1.5
Dice index	0.98617	0.9830
Jaccard index	0.97288	0.9669
Average Precision	0.9757718	0.9727
Average Recall	0.9968605	0.9936

Table 5.8: Patient 4 : Average Dice index, average Jaccard index, average Precision and Recall metric on evaluation sets following a different deformation amplitude distribution range.

	Gaussian distribution of amplitude between 0.6 and 1.3	Gaussian distribution of amplitude between 0.5 and 1.5
Dice index	0.9878	0.9845
Jaccard index	0.9760	0.9697
Average Precision	0.9762	0.9703
Average Recall	0.9997	0.9994

5.3.3 Robustness to rotations

Tables 5.9 and 5.10 show that the effects of rotation and cropping is not negligible. This modification results in both a lower precision and a lower recall, which results in a lower Jaccard and Dice index. The Dice index of patient 1 falls by a value 0.05 and the Dice of patient 4 falls by 0.15. However this reduction in performance is under control, and the segmentation performances do not completely collapse as the Dice index between the predicted segmentation masks and the real masks are 0.93 and 0.83 respectively for patients 1 and 4. Furthermore, the variance of the Jaccard index increases but stays under 0.02 for both patients.

The reduction in performance can also be explained by the reduction in resolution of the images due to the cropping. This cropping was applied in order to avoid black borders in the rotated image, and would not appear in the case of a misaligned patient.

In conclusion, this system would still work if the patient is not perfectly aligned in the same position as in the training images but would result in a non-negligible reduction in the performance of the system. One component that could be added to this system is an automatic image rotation, several effective methods can provide excellent automatic image alignment like the principal component analysis based approach of Rehman and Lee [103]. Applying this sort of technique in the preprocessing of the dataset could negate the detrimental effects of image rotation on the segmentation quality.

Table 5.9: Patient 1 : Average Dice index, average Jaccard index, variance of the Jaccard index, average Precision and Recall metric of the same network evaluated on one set without rotations and one set augmented with random image rotations uniformly distributed between -5 and 5 degrees.

	Validation set	Validation set with random rotations
Dice index	0.9841	0.9267
Jaccard index	0.9690	0.8668
Jaccard index variance	0.000315	0.0059
Average Precision	0.9710	0.9413
Average Recall	0.9977	0.9138

Table 5.10: Patient 4: Average Dice index, average Jaccard index, variance of the Jaccard index, average Precision and Recall metric of the same network evaluated on one set without rotations and one set augmented with random image rotations uniformly distributed between -5 and 5 degrees.

	Validation set	Validation set with random rotations
Dice index	0.9819	0.8259
Jaccard index	0.9647	0.7133
Jaccard index variance	0.000519	0.016887
Average Precision	0.9654	0.8389
Average Recall	0.9992	0.8152

5.4 Comparison against a static treatment planning

Tables 5.11-5.13 show the difference in segmentation performance between the proposed method and a method based on a static mask inspired by conventional treatment planning. The proposed method has an increase of 5.6% to 8.6% in terms of Dice index and an increase of 9.8% to 17.3% in terms of precision. However, the static mask offers a typically bigger recall and a slightly lower variance. Nevertheless, this advantage in terms of recall could be reversed if the evaluation is done on a dataset with a larger amplitude variation than on the training set. Furthermore, it is possible to add a slight security margin around the segmentation of the proposed system in order to balance the segmentation more in favor of recall, while balancing the static method in favor of precision would not be as easily done. The difference in the performance of the two methods can be visualized in Figure 5.5.

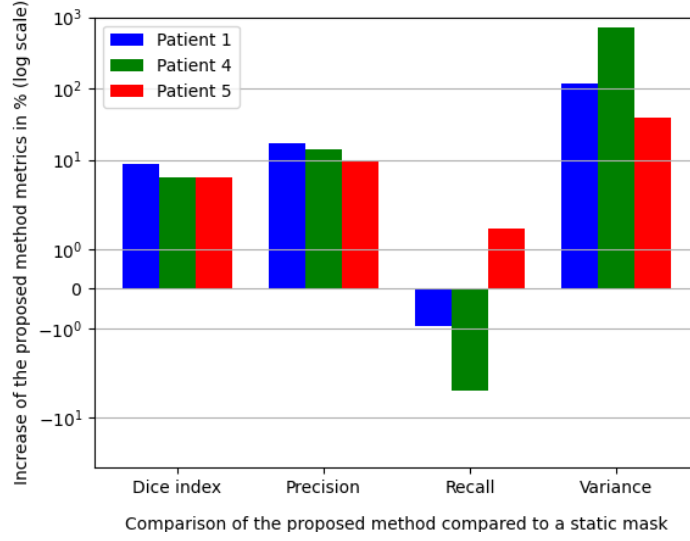


Figure 5.5: Increase in percent of the proposed method in terms Dice index, precision, recall and variance compared to a static mask inspired by static treatment planning (plotted with a symmetric logarithmic vertical scale).

Table 5.11: Patient 1 : Comparison between the segmentation of the proposed method with a static mask inspired by treatment planning

	Dice index	Precision	Recall	Variance
Proposed method	0.9889	0.9810	0.9889	2.3421e-04
Static mask	0.9105	0.8368	0.9983	1.0730e-04

Table 5.12: Patient 4: Comparison between the segmentation of the proposed method with a static mask inspired by treatment planning

	Dice index	Precision	Recall	Variance
Proposed method	0.9720	0.9720	0.95778	0.00121
Static mask	0.9206	0.8530	0.9997	1.479e-04

Table 5.13: Comparison between the segmentation of the proposed method with a static mask inspired by treatment planning (Patient 5)

	Dice index	Precision	Recall	Variance
Proposed method	0.9518	0.940	0.9642	2.473e-04
Static mask	0.9009	0.8569	0.9498	1.783e-04

5.5 Inference time evaluation

Table 5.14 shows the distribution of the end-to-end latency of the proposed network over 100 repetitions. The average latency of the whole system is of 5.698 ms. Overall, the latency of the system is pretty stable with a 0.9 quantile of only 5.684 ms, but a few outliers put the maximum time at 16.161 ms. The average latency allows for the segmentation of 167 images per second on average, with a very short delay compared to the image capture. This means that the system, if coupled with a fast image acquisition setup, could be used on a real-time application. As an adult human typically has a respiratory rate of 12 to 15 respiration per minute [104], every time we get a segmentation, the lungs will only have moved between 0.119% and 0.15% of the whole range of motion of the respiratory cycle. This should be sufficiently a small window to not induce any large error in the segmentation. Even in the worst-case scenario, this movement should only be of about 0.385% of the respiratory range.

We can also note that this inference time marks great improvement compared to the methods of Terunuman et al. [94]. Their method achieved an average delay of 25 ms. This large amelioration could be explained by differences in the network topology, as their network uses a convolutional kernel of size 7x7. Another possible explanation for this gap in performance is the recent advancement in GPU computational power, as their work dates from 2018 and uses a NVIDIA GTX 1080 GPU that is one generation older and not as well optimized for deep-learning workflow than the NVIDIA Quadro RTX 6000 used in this work.

Table 5.14: Metrics and quantiles of the distribution of the end-to-end latency of the network for a single 256x256 image over 100 repetitions.

average inference time	0.1 quantile	0.25 quantile	0.5 quantile	0.75 quantile	0.9 quantile	minimum	maximum
5.698 [ms]	5.589 [ms]	5.609 [ms]	5.638 [ms]	5.658 [ms]	5.684 [ms]	5.428 [ms]	16.191 [ms]

5.6 Summary

In this chapter, we have developed and discussed the results of the different evaluations performed on the proposed neural network. We began by comparing the relative performance of the network with different loss functions and concluded that the optimal one seems to be the focal loss. This loss function offers an almost perfect recall greater than 0.99 for every test, with the highest Jaccard index in the overwhelming majority of the tests. These good results are also very consistent, offering a Jaccard index that never went below 0.91, while other loss functions

were sometimes susceptible to a collapse in performance. Then, the robustness of the network to deformations such as Gaussian noise, rotations, and variations in amplitude was evaluated. These tests showed that the network is able resist to minor deformations, as well as some options to mitigate the effect of these deformations even further. Subsequently, we compared the performance of the proposed network against a static treatment planning. This test showed a gain of 5.6% to 8.6% in terms of Dice index and an increase of 9.8% to 17.3% in terms of precision and the associated cost in terms of recall and stability. Finally, we analyzed the delay associated with the segmentation with the proposed method on our testing hardware. This last test showed that the inference time of the network is short enough for a real-time application.

Chapter 6

Conclusion

As we have seen in this work, proton therapy is a very promising technique for radiotherapy, which is slowly becoming more and more common and offers several advantages compared to conventional radiotherapy. This technique allows to deliver the dose of radiation to the tumor with a high degree of accuracy thanks to a particularity of its energy delivering curve known as the Bragg peak. However, some tumors which are located on organs such as the lung or the liver are highly susceptible to movement due to the respiratory cycle of the patient. Furthermore, the only imaging modality accessible in real-time during the radiotherapy treatment is a 2D low-intensity X-ray fluoroscopic imaging system. Compromises are therefore done in the planning phase in order to consider the mobility of the tumor across the respiratory cycle, and the radiation dose is delivered in a zone larger than the exact tumor location in order to target it at every phase of the respiratory cycle.

Several methods have been developed to overcome the negative effects of the respiratory movement, in order to maintain the effectiveness of the treatment while decreasing the dose of radiation delivered to the adjacent tissue. Some of these methods include controlling the breathing of the patient, timing the dose delivery at specific phases of the respiration cycle, and real-time tumor segmentation. This last solution can be implemented either with the use of implanted fiducial markers around the tumor volume, which is precise but very invasive, or by using more complex markerless tracking.

In this context, the objective of this work was to develop a markerless frame by frame lung tumor tracking system, based on Convolutional Neural Networks derived from the U-net architecture. This type of network has shown promising results in many fields, including medical imaging segmentation, even based on low quality or obstructed images. To first test the feasibility of this project, a synthetic simulation of simplified fluoroscopic images was developed in parallel with a convolutional

neural network. The network was evaluated on this synthetic dataset, in order to determine its architecture and explore the segmentation capabilities of this system on partial information.

Once the system was judged performant enough on the synthetic dataset, a new dataset was developed, based on 4D CT images of five lung tumor patients. This dataset simulates realistic fluoroscopic images and the associated segmentation masks, for any point in the respiratory cycle. The network was analyzed on different datasets, in order to determine its optimal parameters as well as its current capabilities and limitations.

From there, we concluded that a neural network trained with a focal loss function was both the most performant and consistent system among the other proposed versions. Compared with the other versions of the network which are susceptible to a high variation and frequent collapses of the segmentation quality, this one offers a consistent segmentation with an almost perfect recall score greater than 0.99, an average Jaccard index of 0.97, and an average Dice index of 0.98 with a very low variation on 5-fold cross-validation derived from 300 fluoroscopic images over the 5 patients. This system also proved to be robust to variations in respiratory amplitudes, some levels of Gaussian noise, and slight rotations and cropping of the input images.

Furthermore, the proposed system was shown to be reliably more effective than a static treatment planning approach, beating its classification for every patient in the dataset. And most importantly, the system was shown to have a short enough end-to-end latency to be implemented in real-time on current hardware. Indeed, The average latency for one single image of 256x256 pixels is 5ms with very low variance, allowing to compute up to 200 segmentations per second.

Now that we are at the end of this study, we realize that new perspectives can be explored. This study was performed on a relatively small dataset of five patients with lung tumors. In the future, it would be interesting to test this system on a larger number of patients with both lung and liver tumors. Furthermore, it would be interesting to test this system with a more extensive set of meta parameters and loss functions, or even a different network topology in order to further refine its optimal configuration. Besides, this study only includes segmentation on simulated fluoroscopic images derived from 4D CT images, it would therefore be interesting to compare this performance with segmentation on real X-ray fluoroscopic images. This segmentation network could also be extended to perform 3D segmentations, on the basis of the 2D fluoroscopic images. A 3D segmentation would provide the

exact depth of the tumor at each point in time, and allow to exploit the advantage of the Bragg peak to its full extent. Furthermore, this frame by frame segmentation could be combined with a system that takes into account the continuity between the frames such as Kalman filter or Particle filter based approaches, allowing for an even more robust implementation. Moreover, in this work, we focus on a single segmentation system that uses exclusively X-ray fluoroscopic imaging. This system could be replaced by a group of multiple independent systems which would combine their output in order to create the optimal segmentation. These systems based on X-ray fluoroscopic imaging could also be combined with other real-time data sources such as plethysmographic data to measure the current respiratory phase. Besides, in the future proton therapy machines could be equipped with a more advanced imaging system such as magnetic resonance imaging, and this kind of technology could allow the implementation of a 3D to 3D image segmentation with an even higher level of details.

Finally, we can conclude that the perspectives of using CNN-based real-time tumor tracking on fluoroscopic images offer promising results for a more efficient dose delivery in proton therapy. This technique still needs to be analyzed in further detail in order to prove its stability before being implemented in the treatment of cancer patients instead of the conventional procedure. However, with some refinements and additional testing, we hope that this approach could help reduce the negative side effects of radiotherapy and accordingly improve the treatment effectiveness and quality of life of lung cancer patients.

Bibliography

- [1] *World Health Organization: Cancer*. Feb. 3, 2022. URL: <https://www.who.int/news-room/fact-sheets/detail/cancer> (visited on 05/20/2022).
- [2] Carrie F. Dunne-Daly. “Principles of radiotherapy and radiobiology”. In: *Seminars in Oncology Nursing* 15.4 (1999). Radiation Oncology, pp. 250–259. ISSN: 0749-2081. DOI: [https://doi.org/10.1016/S0749-2081\(99\)80054-0](https://doi.org/10.1016/S0749-2081(99)80054-0). URL: <https://www.sciencedirect.com/science/article/pii/S0749208199800540>.
- [3] John Yarnold. “Molecular aspects of cellular responses to radiotherapy”. In: *Radiotherapy and Oncology* 44.1 (1997), pp. 1–7.
- [4] Lei Li, Michael Story, and Randy J Legerski. “Cellular responses to ionizing radiation damage”. In: *International Journal of Radiation Oncology*Biology*Physics* 49.4 (2001), pp. 1157–1162. ISSN: 0360-3016. DOI: [https://doi.org/10.1016/S0360-3016\(00\)01524-8](https://doi.org/10.1016/S0360-3016(00)01524-8). URL: <https://www.sciencedirect.com/science/article/pii/S0360301600015248>.
- [5] T. Sridhar and R. Paul Symonds. “Principles of chemotherapy and radiotherapy”. In: *Obstetrics, Gynaecology & Reproductive Medicine* 19.3 (2009), pp. 61–67. ISSN: 1751-7214. DOI: <https://doi.org/10.1016/j.ogrm.2008.11.011>. URL: <https://www.sciencedirect.com/science/article/pii/S1751721408002418>.
- [6] Fondation contre le Cancer. *Fondation contre le Cancer Radiothérapie*. 2017. URL: <http://https://www.cancer.be/les-cancers/traitements/radioth-erapie> (visited on 04/15/2022).
- [7] Radhe Mohan and David Grosshans. “Proton therapy – Present and future”. In: *Advanced Drug Delivery Reviews* 109 (2017). Radiotherapy for cancer: present and future, pp. 26–44. ISSN: 0169-409X. DOI: <https://doi.org/10.1016/j.addr.2016.11.006>. URL: <https://www.sciencedirect.com/science/article/pii/S0169409X16303192>.

- [8] Tian Xiufang et al. “The evolution of proton beam therapy: Current and future status (Review)”. In: *Molecular and Clinical Oncology* (Nov. 2017). DOI: 10.3892/mco.2017.1499.
- [9] Jorge Otero et al. “Acoustic Localization of Bragg Peak Proton Beams for Hadrontherapy Monitoring”. In: *Sensors* 19 (Apr. 2019), p. 1971. DOI: 10.3390/s19091971.
- [10] Youngyih Han. “Current status of proton therapy techniques for lung cancer”. In: *Radiat Oncol J.* 37.4 (2019), pp. 232–248. DOI: 10.3857/roj.2019.00633. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6952710/#>.
- [11] Wei Liu et al. “Robust optimization of intensity modulated proton therapy”. In: *Medical Physics* 39.2 (2012), pp. 1079–1091. DOI: <https://doi.org/10.1118/1.3679340>. eprint: <https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1118/1.3679340>. URL: <https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3679340>.
- [12] Joe Hanley et al. “Deep inspiration breath-hold technique for lung tumors: The potential value of target immobilization and reduced lung density in dose escalation”. In: *International Journal of Radiation Oncol Biol Phys* 36 (Jan. 1996).
- [13] Darren M. C. Poon et al. “Proton Therapy for Prostate Cancer: Challenges and Opportunities”. In: *Cancers* 14.4 (2022). ISSN: 2072-6694. DOI: 10.3390/cancers14040925. URL: <https://www.mdpi.com/2072-6694/14/4/925>.
- [14] Thomas Bortfeld and Jay Loeffler. “Three ways to make proton therapy affordable”. In: *Nature* 549 (Sept. 2017), pp. 451–453. DOI: 10.1038/549451a.
- [15] Anthony Laurence Zietman. “Particle Therapy at the "Tipping Point": An Introduction to the Red Journal’s Special Edition.” In: *International journal of radiation oncology, biology, physics* 95 1 (2016), pp. 1–3.
- [16] *Particle Therapy Co-Operative Group : Particle therapy facilities in clinical operation*. May 2022. (Visited on 05/08/2022).
- [17] Thomas J. FitzGerald et al. “The Future of Proton Therapy”. In: *Proton Therapy*. Ed. by Thomas J. FitzGerald and Maryann Bishop-Jodoin. Rijeka: IntechOpen, 2021. Chap. 9. DOI: 10.5772/intechopen.97935. URL: <https://doi.org/10.5772/intechopen.97935>.
- [18] Yann LeCun, Y. Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

- [19] Dinggang Shen, Guorong Wu, and Heung-Il Suk. “Deep Learning in Medical Image Analysis”. In: *Annual Review of Biomedical Engineering* 19.1 (2017). PMID: 28301734, pp. 221–248. DOI: 10.1146/annurev-bioeng-071516-044442. URL: <https://doi.org/10.1146/annurev-bioeng-071516-044442>.
- [20] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [21] Myeongsu Kang and Noel Jordan Jameson. “Machine Learning: Fundamentals”. In: *Prognostics and Health Management of Electronics*. John Wiley & Sons, Ltd, 2018. Chap. 4, pp. 85–109. ISBN: 9781119515326. DOI: <https://doi.org/10.1002/9781119515326.ch4>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119515326.ch4>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119515326.ch4>.
- [22] Christoph von der Malsburg. “Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms”. In: *Brain Theory* (Jan. 1986), pp. 245–248. DOI: 10.1007/978-3-642-70911-1_20.
- [23] Marius-Constantin Popescu et al. “Multilayer perceptron and neural networks”. In: *WSEAS Transactions on Circuits and Systems* 8 (July 2009).
- [24] Grace W. Lindsay. “Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future”. In: *Journal of Cognitive Neuroscience* 33.10 (Sept. 2021), pp. 2017–2031. ISSN: 0898-929X. DOI: 10.1162/jocn_a_01544. eprint: https://direct.mit.edu/jocn/article-pdf/33/10/2017/1962083/jocn_a_01544.pdf. URL: https://doi.org/10.1162/jocn%5C_a%5C_01544.
- [25] David H. Hubel. “Vision and the Brain”. In: *Bulletin of the American Academy of Arts and Sciences* 31.7 (1978), pp. 17–28. ISSN: 0002712X. URL: <http://www.jstor.org/stable/3823436> (visited on 04/22/2022).
- [26] Kuniyiko Fukushima. “Neocognitron: A hierarchical neural network capable of visual pattern recognition”. In: *Neural Networks* 1.2 (1988), pp. 119–130. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(88\)90014-7](https://doi.org/10.1016/0893-6080(88)90014-7). URL: <https://www.sciencedirect.com/science/article/pii/S0893608088900147>.
- [27] Kuniyiko Fukushima. “Neocognitron for handwritten digit recognition”. In: *Neurocomputing* 51 (2003), pp. 161–180. ISSN: 0925-2312. DOI: [https://doi.org/10.1016/S0925-2312\(02\)00614-8](https://doi.org/10.1016/S0925-2312(02)00614-8). URL: <https://www.sciencedirect.com/science/article/pii/S0925231202006148>.

- [28] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.
- [29] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9 (2018), pp. 611–629. URL: <https://doi.org/10.1007/s13244-018-0639-9>.
- [30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: 10.48550/ARXIV.1505.04597. URL: <https://arxiv.org/abs/1505.04597>.
- [31] *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Dec. 15, 2018. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a> (visited on 05/20/2022).
- [32] Jongbin Ryu, Ming-Hsuan Yang, and Jongwoo Lim. “DFT-based Transformation Invariant Pooling Layer for Visual Classification”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.
- [33] Huang Jin Jie and Putra Wanda. “RunPool: A Dynamic Pooling Layer for Convolution Neural Network”. In: *International Journal of Computational Intelligence Systems* 13 (1 2020), pp. 66–76. ISSN: 1875-6883. DOI: <https://doi.org/10.2991/ijcis.d.200120.002>. URL: <https://doi.org/10.2991/ijcis.d.200120.002>.
- [34] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8). URL: <https://www.sciencedirect.com/science/article/pii/0893608089900208>.
- [35] *Pytorch manual : Tanh*. URL: <https://pytorch.org/docs/stable/generated/torch.nn.Tanh.html> (visited on 05/15/2022).
- [36] *Pytorch manual : Sigmoid*. URL: <https://pytorch.org/docs/stable/generated/torch.nn.Sigmoid.html> (visited on 05/15/2022).
- [37] *Pytorch manual : RELU*. URL: <https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html> (visited on 05/15/2022).
- [38] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

- [39] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.
- [40] Hinton Geoffrey, Nitish Srivastava, and Kevin Swersky. *Lecture 6a : Overview of mini-batch gradient descent*. 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
- [41] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: <http://jmlr.org/papers/v12/duchi11a.html>.
- [42] Thorvald Julius Sørensen. *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons*. I kommission hos E. Munksgaard, 1948.
- [43] Lee R. Dice. “Measures of the Amount of Ecologic Association Between Species”. In: *Ecology* 26.3 (1945), pp. 297–302. DOI: <https://doi.org/10.2307/1932409>. eprint: <https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.2307/1932409>. URL: <https://esajournals.onlinelibrary.wiley.com/doi/abs/10.2307/1932409>.
- [44] Xue Ying. “An Overview of Overfitting and its Solutions”. In: *Journal of Physics: Conference Series* 1168 (Feb. 2019), p. 022022. DOI: [10.1088/1742-6596/1168/2/022022](https://doi.org/10.1088/1742-6596/1168/2/022022). URL: <https://doi.org/10.1088/1742-6596/1168/2/022022>.
- [45] Payam Refaeilzadeh, Lei Tang, and Huan Liu. “Cross-Validation”. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 532–538. ISBN: 978-0-387-39940-9. DOI: [10.1007/978-0-387-39940-9_565](https://doi.org/10.1007/978-0-387-39940-9_565). URL: https://doi.org/10.1007/978-0-387-39940-9_565.
- [46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: [10.48550/ARXIV.1505.04597](https://arxiv.org/abs/1505.04597). URL: <https://arxiv.org/abs/1505.04597>.
- [47] Xiaodan Hu et al. “RUNet: A Robust UNet Architecture for Image Super-Resolution”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.
- [48] Yin Wang et al. “RailNet: A Segmentation Network for Railroad Detection”. In: *IEEE Access* 7 (2019), pp. 143772–143779. DOI: [10.1109/ACCESS.2019.2945633](https://doi.org/10.1109/ACCESS.2019.2945633).

- [49] Xiuli Bi et al. *D-UNet: A Dual-encoder U-Net for Image Splicing Forgery Detection and Localization*. 2020. DOI: 10.48550/ARXIV.2012.01821. URL: <https://arxiv.org/abs/2012.01821>.
- [50] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. “A U-Net Based Discriminator for Generative Adversarial Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [51] Huimin Huang et al. “UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 1055–1059. DOI: 10.1109/ICASSP40776.2020.9053405.
- [52] Zitao Zeng et al. “RIC-UNet: An Improved Neural Network Based on Unet for Nuclei Segmentation in Histology Images”. In: *IEEE Access* 7 (2019), pp. 21420–21428. DOI: 10.1109/ACCESS.2019.2896920.
- [53] Xiaomeng Li et al. “H-DenseUNet: Hybrid Densely Connected UNet for Liver and Tumor Segmentation From CT Volumes”. In: *IEEE Transactions on Medical Imaging* 37.12 (2018), pp. 2663–2674. DOI: 10.1109/TMI.2018.2845918.
- [54] Yu Weng et al. “NAS-UNet: Neural Architecture Search for Medical Image Segmentation”. In: *IEEE Access* 7 (2019), pp. 44247–44257. DOI: 10.1109/ACCESS.2019.2908991.
- [55] Maurício Cordeiro. *Medium : Creating a Very Simple U-Net Model with PyTorch for Semantic Segmentation of Satellite Images*. URL: <https://medium.com/analytics-vidhya/creating-a-very-simple-u-net-model-with-pytorch-for-semantic-segmentation-of-satellite-images-223aa216e705>.
- [56] Philippe Meyer et al. “Survey on deep learning for radiotherapy”. In: *Computers in Biology and Medicine* 98 (2018), pp. 126–146. ISSN: 0010-4825. DOI: <https://doi.org/10.1016/j.combiomed.2018.05.018>. URL: <https://www.sciencedirect.com/science/article/pii/S0010482518301318>.
- [57] Qianru Zhang et al. “Recent advances in convolutional neural network acceleration”. In: *Neurocomputing* 323 (2019), pp. 37–51. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2018.09.038>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231218311007>.
- [58] Daniel Strigl, Klaus Kofler, and Stefan Podlipnig. “Performance and Scalability of GPU-Based Convolutional Neural Networks”. In: *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*. 2010, pp. 317–324. DOI: 10.1109/PDP.2010.43.

- [59] Lindsay P. Busby, Jesse L. Courtier, and Christine M. Glastonbury. “Bias in Radiology: The How and Why of Misses and Misinterpretations”. In: *RadioGraphics* 38.1 (2018). PMID: 29194009, pp. 236–247. DOI: 10.1148/rg.2018170107. eprint: <https://doi.org/10.1148/rg.2018170107>. URL: <https://doi.org/10.1148/rg.2018170107>.
- [60] *WHAT IS RECAPTCHA?* URL: <https://www.google.com/recaptcha/about/> (visited on 05/08/2022).
- [61] John Launchbury. *A DARPA perspective on artificial intelligence*. 2017. URL: <https://www.darpa.mil/about-us/darpa-perspective-on-ai> (visited on 05/08/2022).
- [62] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal adversarial perturbations”. In: *CoRR* abs/1610.08401 (2016). arXiv: 1610.08401. URL: <http://arxiv.org/abs/1610.08401>.
- [63] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: <http://arxiv.org/abs/1311.2901>.
- [64] Amelie Echle et al. “Deep learning in cancer pathology: a new generation of clinical biomarkers”. In: *British Journal of Cancer* 124 (Nov. 2020), pp. 1–11. DOI: 10.1038/s41416-020-01122-x.
- [65] Pavol Harar et al. “Voice Pathology Detection Using Deep Learning: a Preliminary Study”. In: *2017 International Conference and Workshop on Bioinspired Intelligence (IWOBI)*. 2017, pp. 1–4. DOI: 10.1109/IWOBI.2017.7985525.
- [66] Pegah Khosravi et al. “A Deep Learning Approach to Diagnostic Classification of Prostate Cancer Using Pathology–Radiology Fusion”. In: *Journal of Magnetic Resonance Imaging* 54.2 (2021), pp. 462–471. DOI: <https://doi.org/10.1002/jmri.27599>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/jmri.27599>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jmri.27599>.
- [67] Roslidar Roslidar et al. “A Review on Recent Progress in Thermal Imaging and Deep Learning Approaches for Breast Cancer Detection”. In: *IEEE Access* 8 (2020), pp. 116176–116194. DOI: 10.1109/ACCESS.2020.3004056.
- [68] Santiago Tello-Mijares, Fomuy Woo, and Francisco Flores. “Breast Cancer Identification via Thermography Image Segmentation with a Gradient Vector Flow and a Convolutional Neural Network”. In: *Journal of Healthcare Engineering* 2019 (2019).

- [69] Roslidar Roslidar et al. “A Study of Fine-Tuning CNN Models Based on Thermal Imaging for Breast Cancer Classification”. In: Aug. 2019, pp. 77–81. DOI: 10.1109/CYBERNETICSCOM.2019.8875661.
- [70] Matheus de Freitas Oliveira Baffa and Lucas Grassano Lattari. “Convolutional Neural Networks for Static and Dynamic Breast Infrared Imaging Classification”. In: *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (2018), pp. 174–181.
- [71] Mohamed Abdel-Nasser, Antonio Moreno, and Domenec Puig. “Breast Cancer Detection in Thermal Infrared Images Using Representation Learning and Texture Analysis Methods”. In: *Electronics* 8.1 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8010100. URL: <https://www.mdpi.com/2079-9292/8/1/100>.
- [72] Sebastien Jean Mambou et al. “Breast Cancer Detection Using Infrared Thermal Imaging and a Deep Learning Model”. In: *Sensors* 18.9 (2018). ISSN: 1424-8220. DOI: 10.3390/s18092799. URL: <https://www.mdpi.com/1424-8220/18/9/2799>.
- [73] Yaniv Bar et al. “Chest pathology detection using deep learning with non-medical training”. In: *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)* (2015), pp. 294–297.
- [74] Phillip Chlap et al. “A review of medical image data augmentation techniques for deep learning applications”. In: *Journal of Medical Imaging and Radiation Oncology* 65.5 (2021), pp. 545–563. DOI: <https://doi.org/10.1111/1754-9485.13261>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/1754-9485.13261>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/1754-9485.13261>.
- [75] Agnieszka Mikołajczyk and Michał Grochowski. “Data augmentation for improving deep learning in image classification problem”. In: *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. 2018, pp. 117–122. DOI: 10.1109/IIPHDW.2018.8388338.
- [76] Connor Shorten and Taghi Khoshgftaar. “A survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data* 6 (July 2019). DOI: 10.1186/s40537-019-0197-0.
- [77] Fernando Pérez-García, Rachel Sparks, and Sébastien Ourselin. “TorchIO: A Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning”. In: *Computer Methods and Programs in Biomedicine* 208 (2021), p. 106236. ISSN: 0169-2607. DOI: <https://doi.org/10.1016/j.cmpb.2021.106236>. URL: <https://www.sciencedirect.com/science/article/pii/S0169260721003102>.

- [78] Zeynettin Akkus et al. “Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions”. In: *Journal of digital imaging* 30 (June 2017). DOI: 10.1007/s10278-017-9983-4.
- [79] Fausto Milletari et al. “Hough-CNN: Deep learning for segmentation of deep brain regions in MRI and ultrasound”. In: *Computer Vision and Image Understanding* 164 (2017). Deep Learning for Computer Vision, pp. 92–102. ISSN: 1077-3142. DOI: <https://doi.org/10.1016/j.cviu.2017.04.002>. URL: <https://www.sciencedirect.com/science/article/pii/S1077314217300620>.
- [80] Mehrdad Noori, Ali Bahri, and Karim Mohammadi. “Attention-Guided Version of 2D UNet for Automatic Brain Tumor Segmentation”. In: *2019 9th International Conference on Computer and Knowledge Engineering (ICCCKE)*. 2019, pp. 269–275. DOI: 10.1109/ICCCKE48569.2019.8964956.
- [81] Fan Xu et al. “LSTM Multi-modal UNet for Brain Tumor Segmentation”. In: *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*. 2019, pp. 236–240. DOI: 10.1109/ICIVC47709.2019.8981027.
- [82] Albert Comelli et al. “Deep Learning-Based Methods for Prostate Segmentation in Magnetic Resonance Imaging”. In: *Applied Sciences* 11.2 (2021). ISSN: 2076-3417. DOI: 10.3390/app11020782. URL: <https://www.mdpi.com/2076-3417/11/2/782>.
- [83] Nazanin Beheshti and Lennart Johnsson. “Squeeze U-Net: A Memory and Energy Efficient Image Segmentation Network”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, pp. 1495–1504. DOI: 10.1109/CVPRW50498.2020.00190.
- [84] Imran Ahmed et al. “Comparison of Deep-Learning-Based Segmentation Models: Using Top View Person Images”. In: *IEEE Access* PP (July 2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.3011406.
- [85] *Advanced Radiation Treatment with MRI-Guided Linear Accelerator (MRI-LINAC)*. URL: <https://www.brighamandwomens.org/radiation-oncology/mri-guided-linear-accelerator> (visited on 05/08/2022).
- [86] B.M. Oborn et al. “MRI Guided Proton Therapy: pencil beam scanning in an MRI fringe field”. In: *Radiotherapy and Oncology* 118 (Feb. 2016), S78–S79. DOI: 10.1016/S0167-8140(16)30160-8.
- [87] Bradley M. Oborn et al. “Future of medical physics: Real-time MRI-guided proton therapy”. In: *Medical Physics* 44 (2017), e77–e90.

- [88] Trang Thanh Pham et al. “Magnetic resonance imaging (MRI) guided proton therapy: A review of the clinical challenges, potential benefits and pathway to implementation”. In: *Radiotherapy and Oncology* (2022). ISSN: 0167-8140. DOI: <https://doi.org/10.1016/j.radonc.2022.02.031>. URL: <https://www.sciencedirect.com/science/article/pii/S0167814022001165>.
- [89] Aswin Hoffmann et al. “MR-guided proton therapy: A review and a preview”. In: *Radiation oncology (London, England)* 15 (May 2020), p. 129. DOI: 10.1186/s13014-020-01571-x.
- [90] Mikado Imura et al. “Insertion and fixation of fiducial markers for setup and tracking of lung tumors in radiotherapy”. In: *International Journal of Radiation Oncology*Biophysics* 63.5 (2005), pp. 1442–1447. ISSN: 0360-3016. DOI: <https://doi.org/10.1016/j.ijrobp.2005.04.024>. URL: <https://www.sciencedirect.com/science/article/pii/S0360301605007133>.
- [91] *radiologyinfo.org : Fiducial Marker Placement, What are the benefits vs. risks?* 2021. URL: <https://www.radiologyinfo.org/en/info/fiducial-marker#:~:text=Fiducial%5C%20markers%5C%20are%5C%20small%5C%20metal,of%5C%20radiation%5C%20beams%5C%20during%5C%20treatment>. (visited on 05/05/2022).
- [92] Qianyi Xu et al. “Lung tumor tracking in fluoroscopic video based on optical flow”. In: *Medical physics* 35 (Jan. 2009), pp. 5351–9. DOI: 10.1118/1.3002323.
- [93] Ying Cui et al. “Multiple template-based fluoroscopic tracking of lung tumor mass without implanted fiducial markers”. In: *Physics in Medicine and Biology* 52.20 (Oct. 2007), pp. 6229–6242. DOI: 10.1088/0031-9155/52/20/010. URL: <https://doi.org/10.1088/0031-9155/52/20/010>.
- [94] Toshiyuki Terunuman, Aoi Tokui, and Takeji Sakae. “Novel real-time tumor-contouring method using deep learning to prevent mistracking in X-ray fluoroscopy”. In: *Radiological Physics and Technology* 11 (2018), pp. 43–53.
- [95] Xiao-Yun Zhou and Guang-Zhong Yang. “Normalization in Training Deep Convolutional Neural Networks for 2D Bio-medical Semantic Segmentation”. In: *CoRR* abs/1809.03783 (2018). arXiv: 1809.03783. URL: <http://arxiv.org/abs/1809.03783>.
- [96] *Pytorch manual : CONVTRANSPOSE2D*. URL: <https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html> (visited on 05/25/2022).

- [97] Thom Lane. *Medium : Transposed Convolutions explained with... MS Excel!* Nov. 2, 2018. URL: <https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8> (visited on 05/25/2022).
- [98] .
- [99] *Scikit-learn : sklearn.metrics.logloss*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html (visited on 05/07/2022).
- [100] Francesco Caliva et al. “Distance Map Loss Penalty Term for Semantic Segmentation”. In: *International Conference on Medical Imaging with Deep Learning – Extended Abstract Track*. London, United Kingdom, Aug. 2019. URL: <https://openreview.net/forum?id=B1eIcvS45V>.
- [101] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2020), pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.
- [102] Davood Karimi and Septimiu E. Salcudean. *Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks*. 2019. arXiv: 1904.10030 [eess.IV].
- [103] Hafiz Zia Ur Rehman and Sungon Lee. “Automatic Image Alignment Using Principal Component Analysis”. In: *IEEE Access* 6 (2018), pp. 72063–72072. DOI: 10.1109/ACCESS.2018.2882070.
- [104] Kim E. Barrett et al. “Respiratory Physiology: Introduction”. In: *Ganong’s Review of Medical Physiology, 25e*. New York, NY: McGraw-Hill Education, 2018. URL: accessmedicine.mhmedical.com/content.aspx?aid=1115832208.

Appendix A

Additional evaluation metrics

A.1 Robustness to noise

Table A.1: Patient 1 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,972499	0,961596	0,876048	0,498979	0,053296	0,022629
Jaccard index	0,985941	0,980111	0,931365	0,655054	0,100617	0,044236
Jaccard index variance	0,000441	0,001099	0,007873	0,017987	0,000341	1,11E-05
Precision	0,974258	0,964218	0,883904	0,528675	0,054758	0,022748
Recall	0,998036	0,997054	0,989677	0,894311	0,627984	0,79896

Table A.2: Patient 2 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,973877	0,951734	0,877028	0,376342	0,018947	0,010225
Jaccard index	0,986711	0,975139	0,934007	0,541348	0,036776	0,02006
Jaccard index variance	0,00021	0,00048	0,001557	0,006954	0,000222	9,56E-05
Precision	0,975167	0,957594	0,896906	0,426541	0,034188	0,0195
Recall	0,998612	0,993459	0,97502	0,747473	0,040638	0,020991

Table A.3: Patient 3 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,992034	0,991317	0,989403	0,980487	0,667704	0,206797
Jaccard index	0,995998	0,995635	0,994663	0,990109	0,796743	0,340529
Jaccard index variance	1,28E-05	1,81E-05	3,94E-05	0,000147	0,008851	0,001922
Precision	0,992542	0,992448	0,991386	0,986711	0,696545	0,220215
Recall	0,999482	0,998848	0,997973	0,993584	0,940235	0,760566

Table A.4: Patient 4 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,918556	0,918166	0,917213	0,913891	0,895825	0,779816
Jaccard index	0,956757	0,956545	0,956042	0,954248	0,94439	0,870794
Jaccard index variance	0,002772	0,002772	0,002714	0,00265	0,002249	0,013237
Precision	0,937482	0,93729	0,936966	0,935344	0,926482	0,928747
Recall	0,976976	0,976744	0,976051	0,974089	0,963222	0,831714

Table A.5: Patient 5 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data with no added noise.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,905611	0,905867	0,906104	0,907977	0,900629	0,743591
Jaccard index	0,950459	0,9506	0,950729	0,951753	0,947636	0,851493
Jaccard index variance	3,18E-05	3,26E-05	3,6E-05	5,7E-05	0,000274	0,003744
Precision	0,94964	0,949532	0,949109	0,948135	0,933547	0,79334
Recall	0,951486	0,95188	0,952563	0,955607	0,962549	0,924255

Table A.6: Patient 1 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,97568	0,975202	0,974329	0,970321	0,892436	0,2241
Jaccard index	0,987611	0,987363	0,98692	0,984835	0,942051	0,343735
Jaccard index variance	0,000299	0,000311	0,000293	0,000381	0,003606	0,021714
Precision	0,980262	0,97999	0,979396	0,976483	0,913448	0,436507
Recall	0,995122	0,994903	0,994617	0,993406	0,973925	0,296734

Table A.7: Patient 2 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,961503	0,961492	0,960059	0,953941	0,871636	0,324708
Jaccard index	0,980223	0,980205	0,979443	0,976216	0,929793	0,448707
Jaccard index variance	0,000555	0,000599	0,000657	0,00077	0,004746	0,046385
Precision	0,970642	0,970982	0,971093	0,970128	0,956646	0,811788
Recall	0,990088	0,989707	0,988059	0,982556	0,907023	0,341756

Table A.8: Patient 3 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,981844	0,981682	0,981276	0,980086	0,95316	0,528447
Jaccard index	0,990824	0,990741	0,990534	0,989925	0,975727	0,686429
Jaccard index variance	5,78E-05	5,88E-05	5,94E-05	6,9E-05	0,001054	0,009171
Precision	0,982269	0,98216	0,981891	0,98112	0,956432	0,532462
Recall	0,999558	0,999502	0,999362	0,998923	0,996436	0,986495

Table A.9: Patient 4 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,962791	0,9626	0,962512	0,962319	0,958296	0,945792
Jaccard index	0,980888	0,980787	0,980744	0,980631	0,978457	0,971654
Jaccard index variance	0,000578	0,000582	0,000575	0,000621	0,000909	0,001744
Precision	0,967167	0,966999	0,967061	0,967169	0,964988	0,960516
Recall	0,995232	0,99521	0,995047	0,994695	0,992548	0,983296

Table A.10: Patient 5 : Evolution of the average Dice index, Jaccard index, Jaccard index variance, Precision and Recall of the segmentation for different levels of Gaussian noise in the validation dataset. The network has been trained on data augmented with Gaussian noise with a variance equal to 1% of the maximum intensity of the fluoroscopic image.

	noise std : 0%	noise std : 0,5%	noise std : 1%	noise std : 2%	noise std : 5%	noise std : 10%
Dice index	0,904163	0,904177	0,904268	0,90522	0,891322	0,65749
Jaccard index	0,949655	0,949663	0,949713	0,950236	0,942403	0,790608
Jaccard index variance	5,01E-05	5,01E-05	5,11E-05	5,58E-05	0,000446	0,006188
Precision	0,956718	0,956609	0,95647	0,957499	0,960894	0,891573
Recall	0,942712	0,942834	0,943067	0,943109	0,92499	0,714626

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl