UCLouvain

École polytechnique de Louvain

Organs detection in CT and CBCT using deep learning for radiotherapy applications

Author: **Luca Derumier**
Supervisor: **Benoit Macq**
Readers: **Eliott Brion, Vincent Francois-Lavet**
Academic year 2019–2020
Master [120] in Computer Science and Engineering

Abstract

Radiotherapy uses high doses of radiation to damage cancer cells. For prostate cancer patients, the current treatment workflow of radiotherapy does not take into account the organ deformations occurring between sessions. This can lead to uncertainties about the dose delivered to the tumor and surrounding healthy organs, possibly giving rise to unwanted side effects that impact the patient's living conditions. Segmenting those organs with deep learning methods on Computed Tomography (CT) or Cone Beam Computed Tomography (CBCT) scans acquired on treatment day would reduce these uncertainties. Currently, the scans are too large to fit inside a GPU, which is holding back the adoption of these methods. The aim of our work is to identify the organs of interest, locate them and reduce the size of the scan down to a small bounding box that contains them. This bounding box will then serve as input to a segmentation algorithm. We start by studying two algorithms based on traditional computer vision techniques and show that they are inefficient for our problem. We then use a deep learning model and train it to detect and localize bladder, rectum and prostate. We test our model on 450 2-dimensional CT images from 7 patients and obtain an average Intersection over Union (IoU) score of 0.681 ± 0.233, 0.496 ± 0.198 and 0.538 ± 0.245 for bladder, rectum and prostate respectively with and average inference time of 0.02 second per image. We extend the model in 3 dimensions by combining the slices of a same patient together. This time, we reach an IoU score of 0.822 ± 0.060, 0.688 ± 0.065 and 0.619 ± 0.192 for the same organs. Finally, we merge all the organ boxes to take out a final box that contains them and is directly usable by a segmentation algorithm. By doing this, we are able to get rid of 95.6% of the original data volume while still reaching 0.850 ± 0.033 of IoU and 1.72 seconds of processing speed on average. We define a baseline method to compare with ours. The baseline obtains 0.706 ± 0.097 of IoU on average and is beaten by our method on every single patient of the test set. Hence, our work brings out the great potential of deep learning methods for localizing organs and reducing the size of medical images. This algorithm could be used, not only for other pathologies than prostate cancer in radiotherapy or proton therapy, but also for any segmentation application of a large volume.

Contents

List of Abbreviations

Adam Adaptive Moment Estimation

CBCT Cone Beam Computed Tomography

CNN Convolutional Neural Network

CPU Central Processing Unit

CT Computed Tomography

CTV Clinical Target Volume

DSC Dice Similarity Coefficient

EBRT External Beam Radiation Therapy

GPU Graphic Processing Unit

GTV Gross Tumor Volume

HU Hounsfield Units

IoU Intersection over Union

IRT Internal Radiation Therapy

NN Neural Network

PT Proton Therapy

R-CNN Region-based Convolutional Neural Network

ReLU Rectified Linear Unit

ResNet Residual Networks

RT Radiation Therapy

SOBP Spread-Out Bragg Peak

SOTA State-of-the-art

SVM Support-vector Machine

VRAM Video Random Access Memory

YOLO You Only Look Once

List of Symbols

γ Confidence threshold

$\mathcal{L}$ Loss function

λ Loss component weight

μ Mean

σ Standard deviation

Acknowledgements

First of all, I would like to express my sincere gratitude to Pr. Benoît Macq for the opportunity he has given me to work on this fascinating subject. His guidance and support throughout the project have been essential to its completion.

I would also like to thank Eliott Brion for the incredible amount of time he has dedicated to helping me. The knowledge he has shared with me and the passion he shared it with gave me hope and motivation. I'm deeply grateful for that.

I greatly appreciate the advice of Victorien Sonneville and the Open Hub team during the early stages of this master's thesis.

Finally, I want to thank the CHU of Charleroi and Namur for the data they provided, without which none of this would have been possible.

Chapter 1

Introduction

1.1 Cancer and prostate cancer

Cancer is the name given to a collection of related diseases. In all types of cancer, some of the body's cells begin to divide without stopping. Prostate cancers are cancers that develop in the prostate which is gland from the male reproductive system found below the bladder and in front of the rectum [22]. We will be particularly interested in prostate cancer as part of this master's thesis.

Cancer can start almost anywhere in the human body, which is made up of trillions of cells [21]. This, along with some other factors, makes it the second biggest cause of mortality worldwide. In 2017, cancer was responsible for about 9.56 million of deaths, leading respiratory diseases by almost 6 million [38]. Prostate cancer is the fourth most common cancer worldwide (men and women) accounting for a total of 1.28 million cases in 2018 [31]. Although it is very common, very few people actually die from prostate cancer. In fact, the 10-year survival rate for local, regional, and distant prostate cancer combined is 98%, which mean that 98% of the patients diagnosed with prostate cancer will live at least ten years after the cancer is found [2]. There is, however, still a need for improving current treatment techniques as prostate cancer can have a significant impact on the patient's lifestyle.

Despite the impressive numbers, fighting cancer is not a lost cause. In fact, research and development in medicine and related technologies have made us way more efficient in curing cancer than we have ever been before. Figure 1.2 depicts the prediction from 1990 on the number of deaths due to cancer against the actual number that we know today. During a 31 years period, 5 290 000 cancer deaths have been avoided (3 517 000 in men and 1 773 000 in women). In 2019 alone are predicted to be avoided 237 000 in men and 122 000 in women, for a total of 359 000 [29].

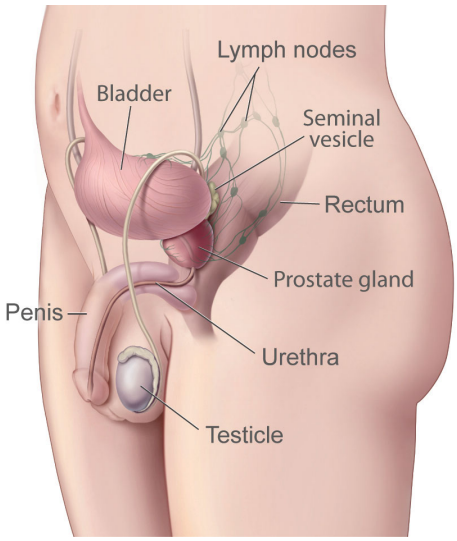


Figure 1.1: Schematic of the pelvic organs in a male body. Credits : [1].

There are many types of cancer treatment. The treatment that is given depends on the type and the stage of the tumor. The three most common are surgery, radiotherapy and chemotherapy [21]. Today, around 50% of all cancer patients require radiotherapy at some point and this number is predicted to rise up to 62.5% by 2025 [5].

1.2 Radiotherapy

The modus operandi of Radiation Therapy (RT) is to damage the tumor's DNA by using radiations. This can be achieved in multiple manners as there are many ways to induce these radiations.

Within the concept of RT, we can distinguish two techniques : Internal Radiation Therapy (IRT) and External Beam Radiation Therapy (EBRT). The former places radioactive material directly inside or next to the tumor while the latter directs x-ray or particle beams at a tumor from outside the body. EBRT either uses conventional x-ray, based on photons, or particles, usually protons (we then call it Proton Therapy (PT)).

The main asset of protons over photons is that protons, as do all charged

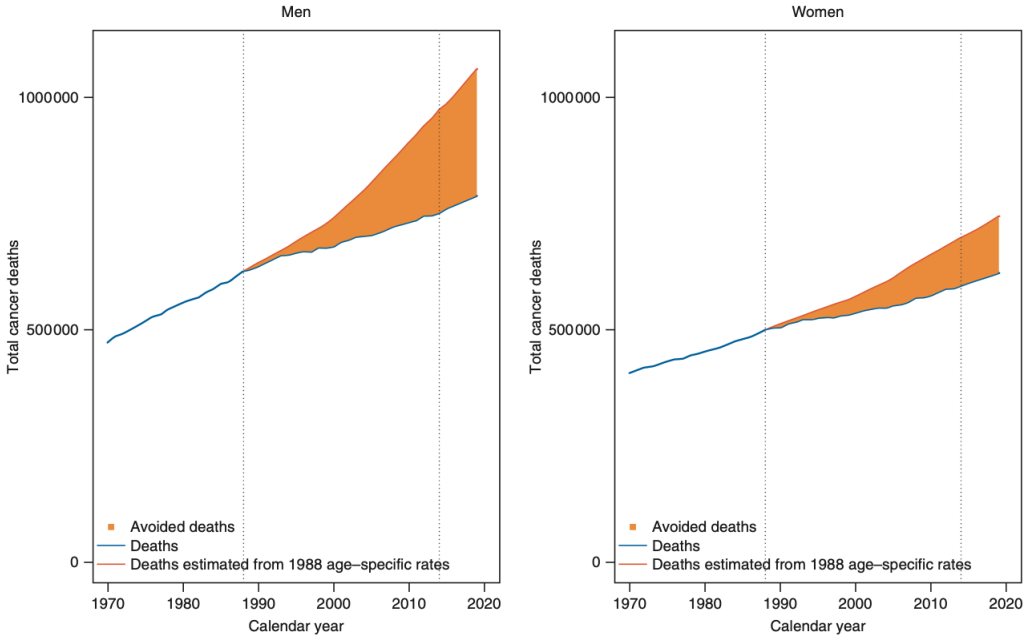


Figure 1.2: Total avoided cancer deaths for EU men between the top rate in 1988 and 2019 (gold, light grey area); observed numbers of cancer deaths from 1970 to 2014 and predicted cancer deaths from 2015 to 2019 (blue, dark grey line); estimated numbers of total cancer deaths by applying 1988 age-specific peak mortality rate (orange, light grey line). Credits : [29].

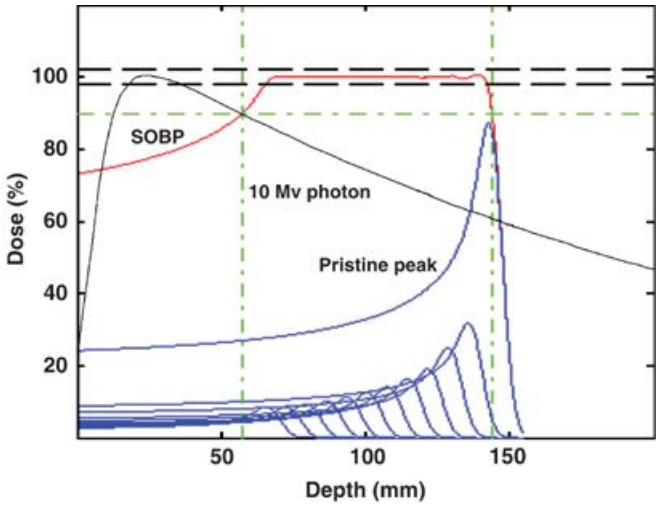


Figure 1.3: Depth–dose distributions for a SOBP (red), its constituent pristine Bragg peaks (blue), and a 10 MV photon beam (black).The dashed lines (black) indicate the clinical acceptable variation in the plateau dose of $\approx$ 2%. The dot–dashed lines (green) indicate the 90% dose and spatial, range and modulation width, intervals. Credits : [26].

particles, have a very rapid energy loss in the last few millimeters of penetration. This results in a sharply localised peak of dose, known as the Bragg peak, and reduces adverse effects to adjacent healthy tissues. To be able to cover enough area at a time, several individually modulated beams are joined together, creating a Spread-Out Bragg Peak (SOBP) (illustrated in figure 1.3) [26].

On the other hand, PT is associated with large economic charges, including both capital investment and operating costs [46]. It is also highly sensitive to changes in the patient's geometry [28]. This can lead to an undesired gap between optimal and delivered dose and ultimately cause unnecessary damage.

1.3 Treatment workflow

As we have seen, RT and PT both have their pros and cons. When a patient is diagnosed with cancer, he, along with the oncologist, will have to make a decision for what treatment option to choose. No matter what option is preferred (PT or RT), the patient will be subject to the clinical workflow described in this section.

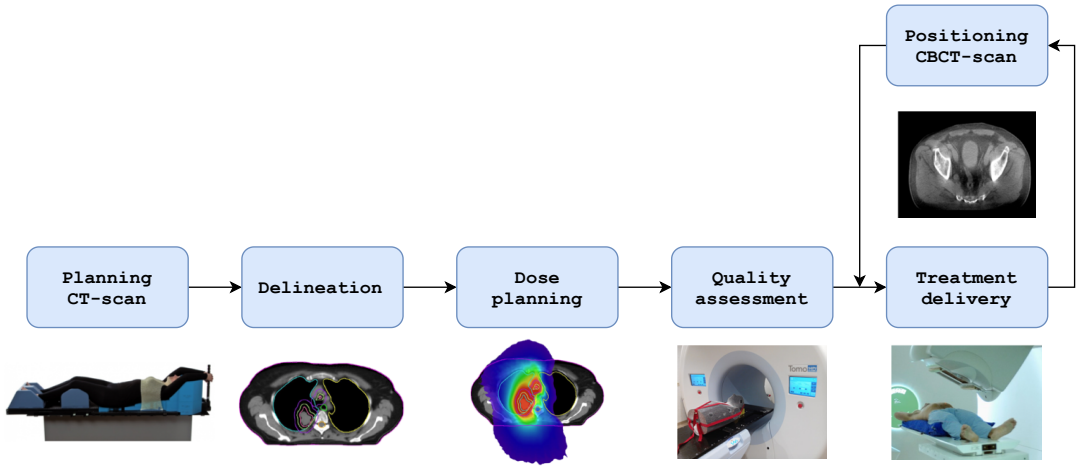


Figure 1.4: Schematic of the treatment process. Source : [33]

First, the patient will go through a planning process called "simulation". During that simulation, a Computed Tomography scan of the area to be treated is taken. Clinicians then delineate the tumor and surrounding healthy organs' volumes on that CT and compute the dose distribution. There is always a trade-off between the damage we want to inflect to the tumor and the risk of damaging surrounding healthy tissues, which we want to avoid. Consequently, it is crucial that the delineation is done in a very accurate manner. The whole process can take up to a week.

At this stage, the treatment can begin. The patient is placed on the table in the same position as for the simulation. To verify accuracy of the placement, a set of Cone Beam Computed Tomography, x-ray or even sometimes CT scans are taken and matched with the original simulation films. After the films have been matched, adjustments are made if that match is not optimal. When positioning is considered to be good enough, the patient is administered the radiation treatment. This is usually repeated five times a week for around four weeks. At least once a week, repeat CBCT or x-ray films will be taken to reconfirm proper positioning. These films will also be performed in most cases where there is a change in the treatment field or treatment plan. The provider will also examine the patient at least once a week to measure the progress [44].

Of course, this workflow is not perfect. One of its main drawbacks is that it does not take into account major changes in the patient's organs geometry. Section 1.4 describes why this is an issue, as well as how we could fix it.

1.4 Geometrical uncertainties and re-planning

A number of uncertainties arise from the treatment process of PT and RT in general. The physical dose delivered to a reference point in a patient can be modelled as

$$\hat{D} = D_I + b + \epsilon \tag{1.1}$$

where D_I is the prescribed dose, b is a bias introduced by baseline deviations between intended and delivered dose, and ϵ is a random variable describing the residual variation in the delivered dose. Epsilon and its parameters (mean and variance) can be defined as follows.

$$\epsilon \sim \mathcal{N}(0, \sigma_D^2)$$

$$\sigma_D^2 = \sigma_{pop}^2 + \sigma_F^2 = \sigma_{pop}^2 + \frac{\sigma_f^2}{N} \tag{1.2}$$

where σ_{pop}^2 is the patient-to-patient variability in a population, σ_F^2 is the variation resulting from inter-fraction or fraction level variability over a full course of N treatment fractions[1] and σ_f^2 is the variation between fractions. An example of the patient-to-patient variability could be, for example, deviations introduced at the stage of the planning CT scan or at simulation. Fraction level variability arises from deviations occurring at the time of therapy. This can be expressed as set-up variability or anatomy changes [4]. For a patient undergoing a pelvic RT treatment, these anatomy changes (and uncertainties related to the data) can arise in several manners, detailed below.

- The bladder is more or less filled.
- The tumor has shrunk.
- The patient has moved.
- The rectum is filled with more or less air.
- The patient has lost weight.
- Relative position between the tumor and the other organs is different.

[1] The full dose of radiation is divided in so called "fractions". They are administered one at a time, usually one fraction per day.

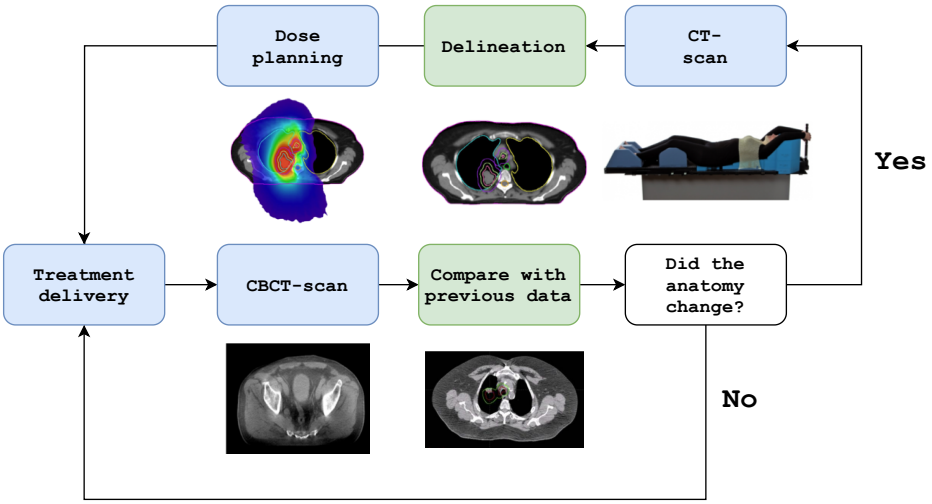


Figure 1.5: Schematic of online re-planning.

These uncertainties can have harmful consequences. Indeed, not knowing the exact location of all organs could lead to healthy tissues receiving a higher dose than intended while the Clinical Target Volume (CTV) (which is the volume that contains the Gross Tumor Volume (GTV) plus a margin for sub-clinical disease spread [6]) would receive a lower one. This can increase the risk of developing unwanted toxicities and failing to kill the tumor.

A way to overcome these geometric changes is to do what is called an *online re-planning*. It consists in re-contouring the organs at risk and the CTV on an image that was taken before the fraction is administered, and calculating a new plan based on that image. For this method to be applicable, it has to be done quickly. Naturally if the patient is to come in every day then the re-planning should only take a few minutes for it to be efficient [47].

Up until recently, online re-planning was not doable in practice because a doctor would have had to contour the images manually. Manual contouring takes hours and is a very costly procedure. But thanks to the recent progress that has been made in deep learning, in particular with State-of-the-art (SOTA) method for biomedical image segmentation "U-Net" [40], we are now able to develop organ contouring algorithms that are both accurate and fast. In 2020, Léger et al. [28] reached SOTA on male pelvic organ segmentation in CBCT with a Dice Similarity Coefficient (DSC)[2] of 0.874 ± 0.096, 0.814 ± 0.055, and 0.758 ± 0.101 for bladder,

[2]Metric that measures the overlap between two binary masks [28].

rectum and prostate, respectively [28]. One of the limitation of this method is that the data that is used for segmentation has to be cropped by hand before it can fit into the GPU. This is not an option in practice as it has to be done very quickly and cropping by hand is cost and time effective.

The latter problem requires an automated method that would detect the organs of interest and extract a bounding box containing all of them. This bounding box can then be input to the segmentation method previously described. In the scope of this master's thesis, we focus on the particular case of pelvis segmentation, but the problem of GPU memory size for heavy data sets is a recurrent problem in deep learning. Section 1.5 addresses this topic.

1.5 GPU memory capacity limitations

Deep learning and neural networks have made it possible to solve problems in a way that has never been done before. But this, inevitably comes at a cost. Indeed, where classical algorithms could usually function on a simple Central Processing Unit (CPU), the vast majority of deep learning models require a GPU to be trained efficiently. But this alone is not enough as SOTA models have massive memory footprints and many GPUs do not have enough Video Random Access Memory (VRAM) to train them. In medical applications, this is a significant limitation as many data sets are made of 3-dimensional images which are very large memory-wise.

The core of the problem resides in two major components : the batch size of training and the available VRAM of the GPU. The batch size is the number of samples that are used before updating the weights and biases of the network. That is, in every training step, a batch of samples is propagated through the model and then backward propagated to calculate gradients for every sample. These gradients will then be summed up and averaged. The average value we get is then used to update the model variables, in different ways depending on the optimizer. This means that the larger the batch size, the more intermediate calculations need to be stored before we can update the model and so discard those calculations. For this reason, training batch size has a huge impact on the required GPU memory for training a neural network [17]. One could argue that we should choose a smaller batch size in order to overcome this limitation. Though, it is not this easy. Batch size can have a consequential impact on training so it is essential to choose a batch size that suits the model and the problem. Larger batch size might have a quicker convergence rate but poor generalization while smaller batch size have the opposite effects. So usually, we want to choose a batch size that falls somewhere in between.

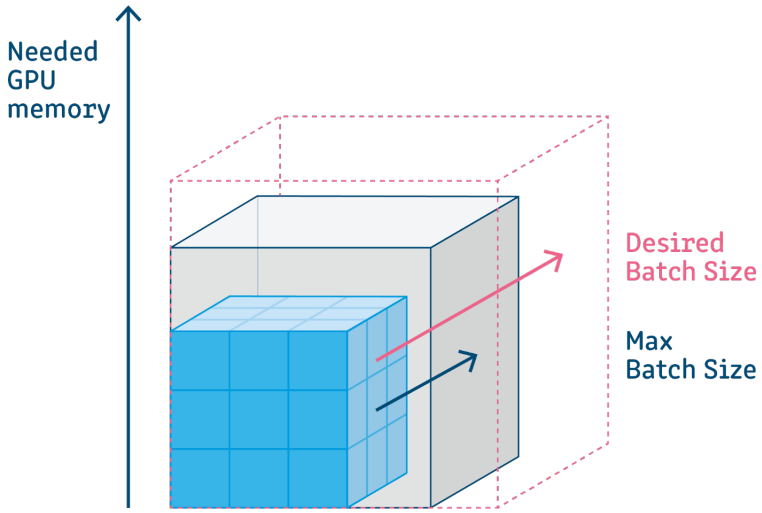


Figure 1.6: As batch size grows, more GPU memory is required. The grey cube represents hardware limitation while the blue cube represents the current batch size, both in terms of memory. Source : [17]

To allow the user to choose a batch size within a large range of values, we have to reduce the size of the input data. This also has a second impact on the model which helps saving GPU memory. Indeed the smaller the input data, the less weights your model will need and the less computation it will make.

In the scope of this master's thesis, our goal is to reduce the size of CT scans so we can fit a batch of at least two in a NVIDIA GeForce GTX 1080 (11 GB of VRAM) which are the batch size and hardware used by Léger et al. [28]. In practice, this corresponds to a final dimension shape of $160 \times 160 \times 128$ for an image and corresponding segmentation mask.

Doing so requires an efficient method for object detection. Section 1.6 explicits how this master's thesis aims at contributing to solve to the previously mentioned issues.

1.6 Contribution of this master's thesis

The goal of this master's thesis is to find an efficient algorithm that accurately detects organs (bladder, rectum and prostate for instance) and extracts a bounding box that contains all of them. This bounding box would then be used as the

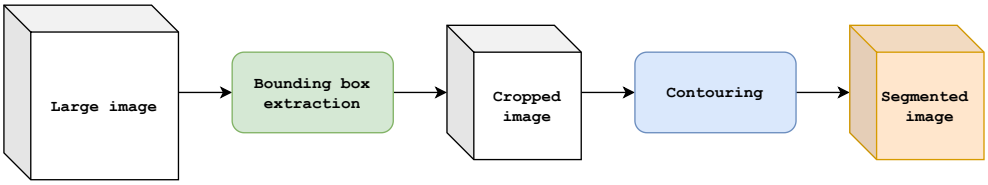


Figure 1.7: The original image is 3-dimensional and large. We feed that image to an object detection algorithm (on the image in green - this thesis) which outputs a smaller image that contains all the organs of interest. We then feed that smaller image to a segmentation algorithm (blue) and get the final segmented image (orange) which holds information about the exact location of the organs.

cropped image, for the contouring algorithm. Figure 1.7 illustrates the process from when the original image is received to when it is ready to be used. In practice, we bring a combination of methods that make sense directly in applications and others that are rather methodological.

To the best of our knowledge, we are the first ones to propose an organ detection and bounding box extraction method for pelvic CT and CBCT scans. This method along with segmentation could be used for different purposes :

1. Identify inter-fractions anatomical changes in the patient's anatomy on CBCT scans and determine if a full re-planning is necessary.
2. In the case of CT data, perform online re-planning.
3. Automatically delineate the organs on CT at the start of the treatment without the doctor's intervention.

Application one and two are highlighted in green in figure 1.5. The difference between the second and third application is that the online re-planning is done after the treatment plan has been set up for the first time. Which implies that the first segmentation has still been done by a doctor. The third application aims to avoid that first human intervention independently of online re-planning.

But globally, our goal is also to show that we can extend a deep learning model that performs well on regular images and get good results on medical images which are harder to work with because of the poor contrast they offer. This method could be used in many other applications as long as they benefit from the reduction of their images input data size.

From an academic point of view, we propose three main innovations :

1. We design the split and crop segmentation method based on computer vision concepts.
2. We take an already existing deep learning model and modify it to fit our problem and data.
3. We extend that model to perform extraction of 3-dimensional bounding boxes.

The first contribution is detailed in section 4.1.2. The deep learning model we use is You Only Look Once (YOLO) and is the topic of section 4.2. All the code that was written and used is available.

1.7 Structure of this document

The rest of the document is organized as follows. In chapter 2, we present the SOTA methods for object detection, compare them to our model of choice and detail some particular applications to organ detection. In chapter 3, we describe the materials that were used and the techniques associated to these materials. In chapter 4, we explain in detail the different methods that were set up and used to solve our problem. In chapter 5, we present the results of the previously described methods and study the impact of some parameters of interest on efficiency and accuracy. We discuss the main results, the limitations of our algorithms and the possible improvements for future work.

1.8 Summary

Prostate cancer is the collection of diseases occurring in the prostate for which some of the body's cells begin to divide without stopping. In 2018, 1.28 million cases of prostate cancer were reported (section 1.1). Radiotherapy aims at damaging the DNA of the tumor using radiations, in order to kill it (section 1.2). Geometrical uncertainties about organs location are jeopardizing the quality of the treatment delivery and the current clinical workflow does not adapt to anatomical changes. When a CBCT or CT scan is acquired before the treatment, it could be used by a segmentation algorithm to re-contour the organs. However the original scan does not fit inside the GPU (section 1.3, 1.4 and 1.5). The purpose of this master's thesis is to find an efficient algorithm that accurately detects organs (bladder, rectum and prostate for instance) and extracts a bounding box that contains all of them and fits inside the GPU (section 1.6).

Chapter 2

State of the art

In chapter 1, we saw that handling geometrical uncertainties and anatomical changes can have a considerable impact on the treatment of prostate cancer patients. We also saw that object detection can be an important step to solve this issue. In this chapter, we introduce some key concepts for object detection (section 2.1) then describe the current SOTA methods for object detection and classification (section 2.2) and their application in medicine (section 2.3).

2.1 Deep learning

Nowadays, most approaches to object detection require machine learning. Machine learning is the branch of artificial intelligence that allows computers to "learn" from data, to enhance their performance based on past experiences and to execute tasks without explicit instructions. Deep learning is the subset of machine learning that allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. It discovers intricate structure in large data sets and has dramatically improved SOTA in many domains including visual object recognition and object detection [25].

Machine learning can be divided into three main paradigms : supervised learning, unsupervised learning and reinforcement learning. Amongst them, supervised learning is the most popular and is the topic of section 2.1.1. Reinforcement learning is the most complex and is discussed in section 2.1.2. In the scope of this master's thesis, we will not talk about unsupervised learning.

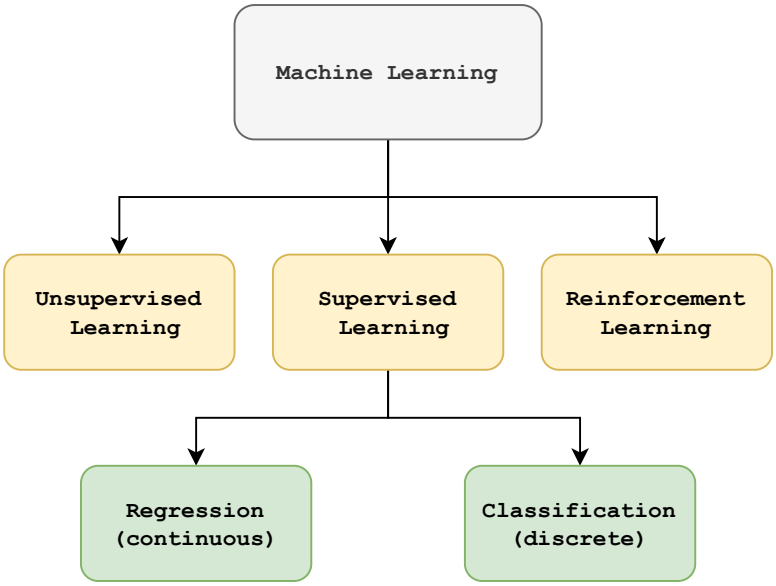


Figure 2.1: Hierarchical diagram of machine learning algorithms with a focus on supervised learning.

2.1.1 Supervised learning

The target of a supervised learning algorithm is to estimate the mapping function (f) from the input variables (x) to the output variables (y). Giving rise to a classification algorithm if the output data is discrete, and to a regression algorithm if it is continuous (figure 2.1).

To understand how it works, imagine that we want to build a system that can classify images as containing, say, a dog or a cat. We first need to collect a lot of images of dogs and cats. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. Our final goal is that after training, the machine outputs the highest score for a dog (respectively a cat) when it is shown an image of a dog (respectively a cat). To do so, we compute an objective function (or loss function) that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable parameters (often called weights) to reduce this error [25].

In the scope of object detection, we feed an image and corresponding annotations to a Neural Network (NN) (usually a Convolutional Neural Network (CNN)) which will output a vector, matrix or tensor that can either take discrete (clas-

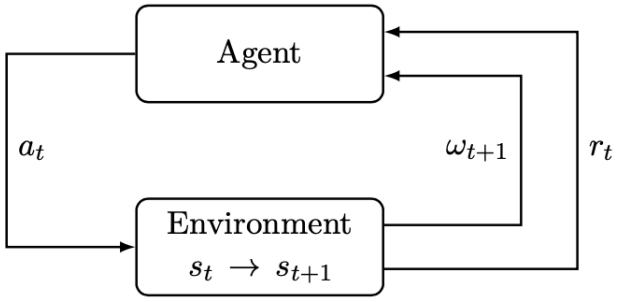


Figure 2.2: Agent-environment interaction in reinforcement learning. The agent takes an action on the environment. Following this action, the environment transitions from state s_t to state s_{t+1} and gives an observation back to the agent as well as a reward or penalty depending on how well it performed. Credits : [13]

sification) or continuous (regression) values. The output can take various forms. If the problem we are facing is a classification problem only (we have an image and we want to know if the object in it is from a certain class), then the output will usually be a vector of length matching the number of classes. Each element can either take 0 if the system thinks the object does not belong to that class or 1 if it thinks the opposite. It could also take values anywhere in between if we want to give the probability that the object belongs to a certain class. If we are facing a localization problem on the other hand, the output can be more complex because it needs to hold more information. Usually, it takes the form of a matrix or tensor that contains information about the location of the bounding boxes (center, height and width, corner coordinates etc.) and associated class probabilities.

2.1.2 Reinforcement learning

Reinforcement learning is the area of machine learning that deals with sequential decision-making. The reinforcement learning problem can be formalized as an agent that has to make decisions in an environment to optimize a given notion of cumulative rewards. Formally, the agent starts in a given state within its environment $s_0 \in \mathcal{S}$, by gathering an initial observation $\omega_0 \in \Omega$. At each time step t, the agent has to take an action $a_t \in \mathcal{A}$. As illustrated in figure 2.2, it follows three consequences: (i) the agent obtains a reward $r_t \in \mathcal{R}$, (ii) the state transitions to $s_{t+1} \in \mathcal{S}$, and (iii) the agent obtains an observation $\omega_{t+1} \in \Omega$ [13].

On the particular case of object detection, the process of localizing objects with

bounding boxes can be seen as a control problem with a sequence of steps to refine the geometry of the box. Determining the exact location of a target object in a scene requires active engagement to understand the context, change the fixation point, identify distinctive parts that support recognition, and determine the correct proportions of the box. With this in mind, we can define a generic action space to be $\mathcal{A} = \{\text{move}, \text{scale}, \text{stop}\}$. The actions that belong to $\mathcal{A}$ respectively perform a displacement (move up, down, left, right), a change in aspect ratio (enlarge, shrink) or a stop if it is done. The agent has a state representation with information of the currently visible region and past actions. It gets rewarded depending on how much the predicted bounding box improved from time step t_i to time step t_{i+1} in terms of overlapping area with the ground truth [7].

2.2 Object detection and classification

Many detection systems have been built over the years to perform object detection. The system we have chosen is a supervised learning algorithm called YOLO. The details of its design are deeply covered by section 4.2. This section focuses on comparing high level differences and similarities between YOLO and other efficient object detection algorithms.

Detection pipelines generally extract features from the image before performing classification or localization to identify the object in the feature space. This process can take various forms. The reason we chose YOLO over other detection systems is because in addition to being both accurate and fast, the open-source code and online resources made it easy to work with, and adapt. The following sections highlight key similarities and differences between top detection frameworks and YOLO [34].

2.2.1 Deformable parts model

Deformable parts model uses a sliding window approach to object detection [12]. It uses a disjoint pipeline to perform each task (such as extracting features, classify regions etc.). YOLO replaces all those parts by a unique neural network that performs all tasks simultaneously. This leads to a faster, more accurate model [34].

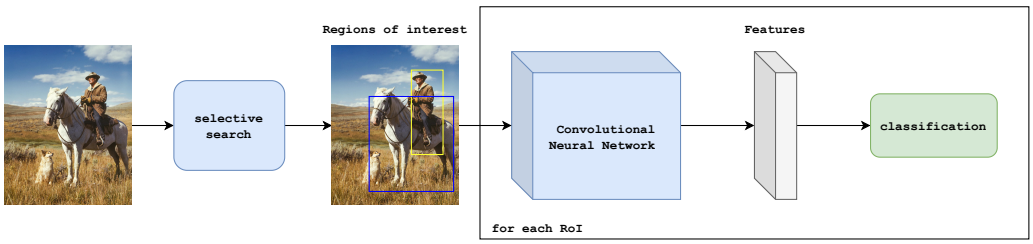


Figure 2.3: Schematic of a simplified R-CNN pipeline.

2.2.2 R-CNN

R-CNN is based on region proposals. Its pipeline relies on four critical steps. It generates potential bounding boxes using Selective Search [45], extract features from each box using a CNN, scores each box with an Support-vector Machine (SVM) and eliminates duplicate detections with non-max suppresion algorithm [15]. This complex pipeline architecture (illustrated in figure 2.3) involves a lot of computation because of the large number of boxes it generates, resulting in slow detection (47 seconds per image).

Like R-CNN, YOLO proposes potential boxes and scores them using a CNN. The difference is that the number of proposed boxes is much smaller than for R-CNN (about a 100 compared to 2000). This speeds up detection significantly.

2.2.3 Fast R-CNN and Faster R-CNN

Fast R-CNN improves the original framework by feeding the image directly to a CNN instead of the region proposals. The CNN will produce a feature map and the regions of interest are then identified from that feature map. Each region is reshaped to a fixed sized feature vector. Each feature vector is fed into a sequence of fully connected layers that finally branch into two sibling output layers : one that produces softmax probability estimates for the classes and another layer that outputs offset values for the bounding box [14]. The whole pipeline is illustrated by figure 2.4. This speeds up the algorithm as the CNN does not have to process 2000 region proposals for the image.

Faster R-CNN eliminates the selective search algorithm and replaces it by a NN of its own for region proposals. This enables learning for the region proposal task and speeds up the process because selective search is very time consuming [36].

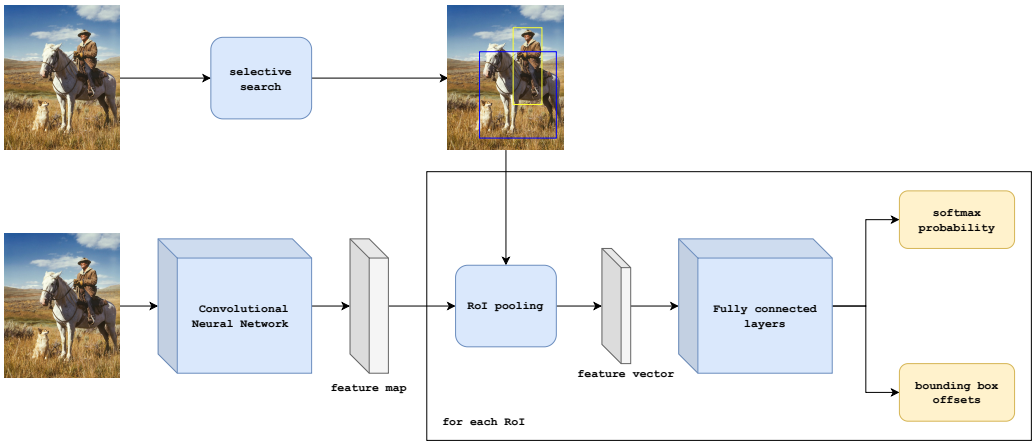


Figure 2.4: Schematic of a simplified Fast R-CNN pipeline.

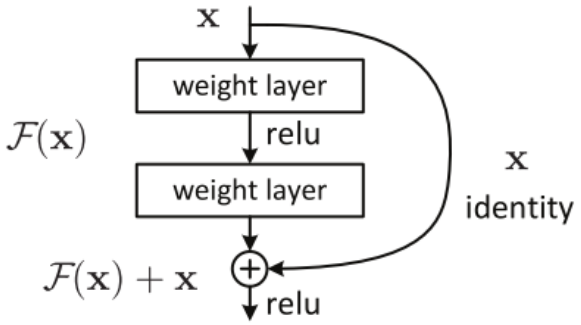


Figure 2.5: Single building block of the residual learning framework. Credits : [18].

Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design [34].

2.2.4 ResNet

ResNet introduces a *deep residual learning* framework that overcomes the issue of vanishing gradient for very deep CNN. This is realized by feedforward neural networks with "shortcut connections" as shown in figure 2.5. The hypothesis is that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping [18].

The second version of YOLO takes inspiration from ResNet and uses that iden-

tity mapping to concatenates the higher resolution features with the low resolution features.

2.3 Medical imaging applications

Object detection and classification have many fields of application and medicine is definitely one of them. As a matter of fact, many studies involving object detection algorithms have demonstrated promising results in complex diagnostics spanning dermatology, radiology, ophthalmology, and pathology [10]. Within the medical imaging applications, we can usually distinguish the classification and detection tasks. Classification was one of the first areas in which deep learning made a major contribution to medical image analysis. In exam classification one typically has one or multiple images (an exam) as input with a single diagnostic variable as output (e.g., disease present or not). Detection, on the other hand, often requires parsing of 3-dimensional volumes [27]. We will focus on the latter task. Specifically, we introduce three methods for organ localization that are based on concepts we introduced in section 2.1 and 2.2.

2.3.1 Multi-label CNN

In 2018, Humpire et al. [20] published a paper on efficient organ localization using multi-label CNN in thorax-abdomen CT scans. The idea is to detect and locate multiple anatomical structures (amongst 11 structures of interest) using a single multi-label CNN for each orthogonal view of a CT scan. Each network takes extra slices around the current slice as input to provide extra context. A sigmoid layer is used to perform multi-label classification. The output of the three networks (sagittal, coronal and axial) is subsequently combined to compute a 3D bounding box for each structure.

They train the neural network and evaluate it on a large set of 1884 thorax-abdomen CT scans from patients undergoing oncological workup with reference bounding boxes annotated by human observers. With this method, they reach SOTA for all 11 structures [20].

2.3.2 Region proposal network

In 2019, Xuanang et al. [48] propose a method for multiple organ localization in CT image using 3D region proposal network. This method takes full advantage of the spatial context by directly implementing a CNN in 3 dimensions instead of detecting the target organs in different slices and assembling them. A novel backbone network architecture is proposed to generate high-resolution feature maps to further improve the localization performance on small organs. Figure 2.6 illustrates this workflow in more details.

The method is evaluated on two clinical data sets, where 11 body organs and 12 head organs are included. The proposed method achieves higher detection precision and localization accuracy than the available SOTA methods with approximate 4 to 18 times faster processing speed [48].

2.3.3 Reinforcement learning

The most recent paper on the subject has been published by Sekuboyina et al. [41] and differentiates itself from the previous papers because it uses reinforcement learning. In this work, an artificial agent is actively self-taught to localize organs in CT by learning from its asserts and mistakes. This helps to reduce the amount of annotated data that is needed by the system to be trained.

The state $s \in \mathcal{S}$ is defined by the voxel values contained inside the current bounding box $b = [b_x, b_y, b_z, b_w, b_h, b_d] \in \mathbb{R}^6$ with the 3D CT scan itself as environment. The set of actions is defined by $\mathcal{A} = [t_x^+, t_x^-, t_y^+, t_y^-, t_z^+, t_z^-, s^+, s^-, d_x, d_y, d_z] \in \mathbb{R}^{11}$. The six first actions correspond to positive and negative translation in all directions. The two following actions are a zoom in and zoom out. Finally, the action thinner d_x flatter d_y and taller d_z represent a deformation on one of the faces of the bounding box. The reward function is linearly related to how much the overlapping areas between the ground truth box and the state box b has improved from state s_t to state s_{t+1}.

The model is trained on 70 CT scans and evaluated on 20. It does not achieve SOTA but reaches good performances with far less data [41].

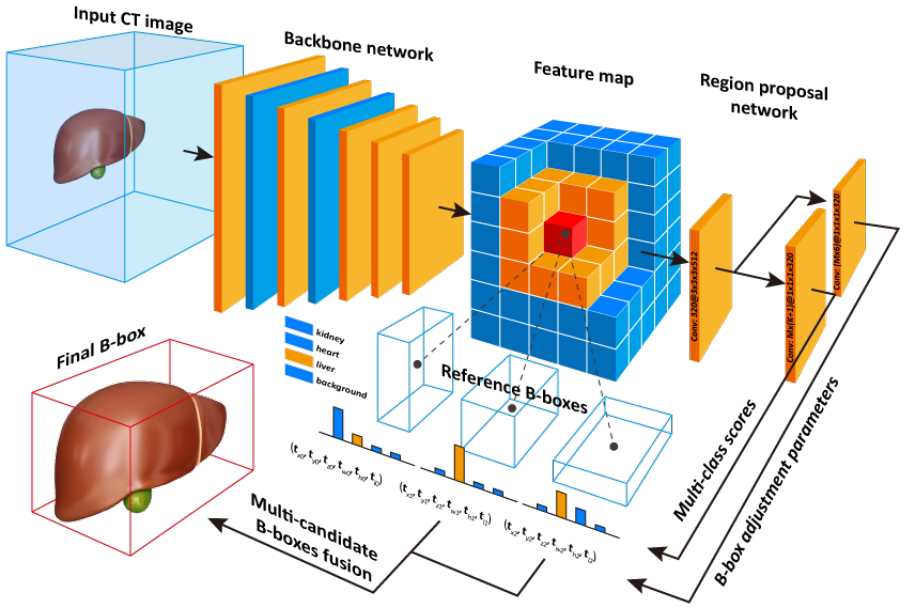


Figure 2.6: Schematic representation of the region proposal network method. For an input CT image, a CNN first extracts its feature map and assign a set of reference bounding boxes with different sizes and shapes to each feature map cell. This feature map is then fed into a subsequent region proposal network to predict multi-class scores and bounding box adjustment parameters for each reference box. By applying the adjustment parameters on the reference bounding boxes, a substantial number of class-specific candidate boxes are generated. Finally, a multi-candidate fusion strategy is applied on the candidate boxes to eliminate redundancy and produce the final B-box of the target organ. The entire method is fully implemented in 3D manner. Credits : [48]

2.4 Summary

Object detection is a core problem in computer vision and it has been studied for many years. Nowadays, most approaches to object detection require machine learning. Machine learning can be divided into three main paradigms : supervised learning, unsupervised learning and reinforcement learning (section 4.2). Supervised learning is the most common type. The target of a supervised learning algorithm is to estimate the mapping function from the input variables to the output variables (section 2.1.1). Reinforcement learning is the area of machine learning that deals with sequential decision-making (section 2.1.2). Many detection systems have been built over the years to perform object detection. Detection pipelines generally extract features from the image before performing classification or localization to identify the object in the feature space (section 2.2). Object detection and classification have many fields of applications and medicine is definitely one of them. Detection often requires parsing of 3-dimensional volumes (section 2.3). Several papers have proposed novelties in order to perform organ detection such as multi-label CNN (section 2.3.1), region proposal network (section 2.3.2) or even reinforcement learning (section 2.3.3).

Chapter 3

Materials

In chapter 2 we introduced the current SOTA methods for object detection with a focus organ detection. In this chapter, we detail the materials that we used to perform organ detection. Section 3.1 explains the imaging techniques that are operated for retrieving the original data. Section 3.2 and 3.3 describe how we constructed our data sets.

3.1 Medical imaging techniques

3.1.1 Computed tomography

A CT scan is a medical imaging procedure that uses computer-processed combinations of many x-ray measurements taken from different angles to produce cross-sectional (tomographic) images (virtual "slices") of specific areas of a scanned object, allowing the user to see inside the object without cutting [8]. The CT scanner uses a motorized x-ray source that rotates around the patient and produces one cross-sectional image (or slice) every rotation by shooting protons on one side and measuring their arrival on the other side. The patient moves across the section as the narrow beams of x-ray are being shot through the body. This allows the CT scanner to produce images that correspond to multiple sections of the area of interest. Once the desired number of slices is collected, they can either be displayed and stored individually as 2D images or stacked together to produce a 3D image [30].

CT scans have several advantages over 2D medical radiography. First, it completely eliminates the superimposition of images of structures outside the area of interest. Second, because of its inherent high-contrast resolution, differences

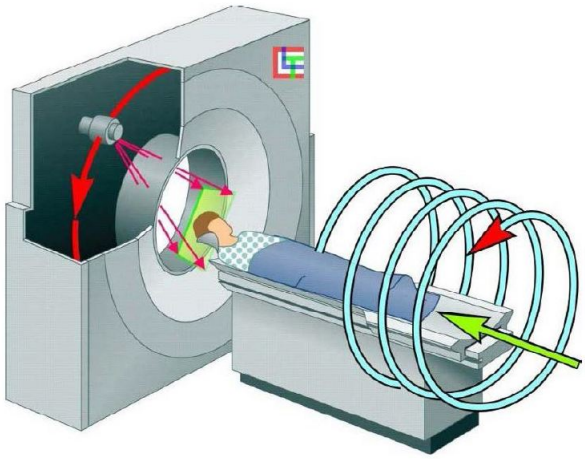


Figure 3.1: Schematic of CT scan. The fan beam transmits radiations in the form of a spiral. Credits : [23].

between tissues that differ in physical density by less than 1% can be distinguished[1]. Finally, data from a single CT imaging procedure consisting of either multiple contiguous or one helical scan can be viewed as images in the axial, coronal, or sagittal planes, depending on the diagnostic task [8]. These benefits enable for more effective medical management by, among others, determining when surgeries are necessary, reducing the need for exploratory surgeries and improving cancer diagnosis and treatment [3].

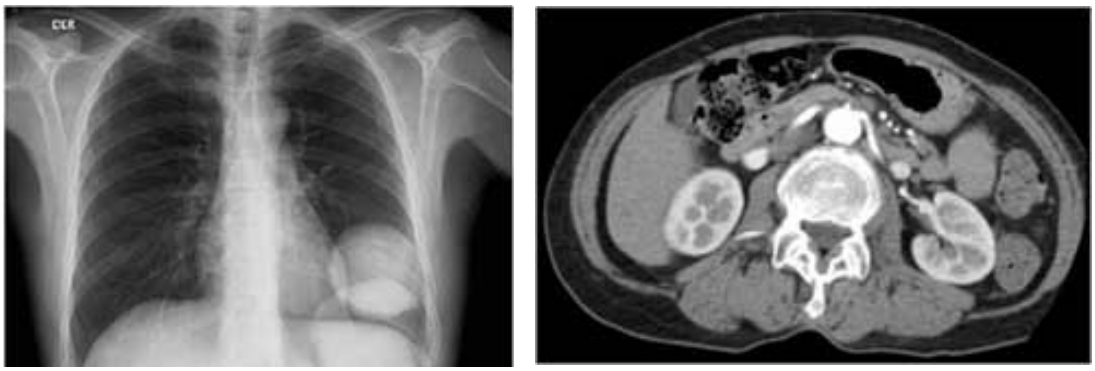
CT scans produce ionizing radiation as does an x-ray. Research shows that this kind of radiation may damage your DNA and lead to cancer, which is controversial as it is used for cancer treatments. But the risk of developing a fatal cancer because of a CT scan remains very small[2]. In fact, most studies on the subject seem to reveal that a single CT scan increases the average patient's risk of developing a fatal tumor of less than one percent[3].

Voxels in a tomographic image represent the relative radiodensity, which itself is opacity to the radio wave and x-ray portion of the electromagnetic spectrum.

[1]Although the patient has to take a special dye called a contrast material to help soft tissues appear clearly.

[2]Some studies that estimate a considerable excess in the risk of cancer from CT scans over the past several decades rely largely on a potentially misleading data set: cancer rates among the long-term survivors of the atomic bomb blasts in World War II [43].

[3]However, this number effectively depends how scientists measure the underlying link between radiation and cancer in the first place [43].



(a) Image of a x-ray scan. (b) Slice of a CT scan.

Figure 3.2: Comparison between x-ray and CT scan images both showing internal body structures. The CT scan image is one of many 2D slices that build the 3D image. Credits : [3].

That is, the relative inability of those kinds of electromagnetic radiation to pass through a particular material [9]. These values are stored in Hounsfield Units (HU) and vary between around -1000 (air, white on the grey scale) and $+3000$ (dense bone, black on the grey scale) [37].

Although CT is quite efficient in terms of imaging, it has its limitations. The first one is that the dose radiation encountered by the patient is relatively high. The second one is that, at the stage of treatment, the scans are taken using what is called a CT-on-rails, which makes use of a sliding table to take the patient from the CT scanner across to the other side of the room where the treatment delivery machine is. This can have a negative impact on accuracy if the patient moves during the process.

3.1.2 Cone beam computed tomography

A CBCT scanner uses a cone shaped beam that radiates from an x-ray source. Thanks to that shape, it covers a large volume with a single rotation. The x-ray images are then sent to a computer and processed to create high resolution 3D images.

CBCT is faster (a single rotation) and cheaper than CT while often exposing the patient to less radiation [24]. Furthermore, the machine itself occupies less space than a CT scanner. As for CT, the output values of a CBCT scan are stored in HU.

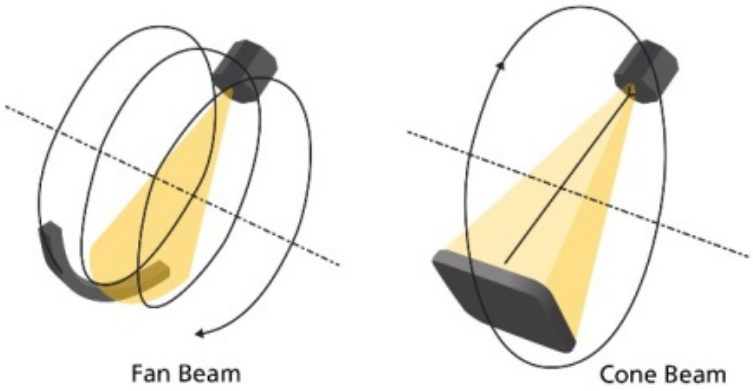
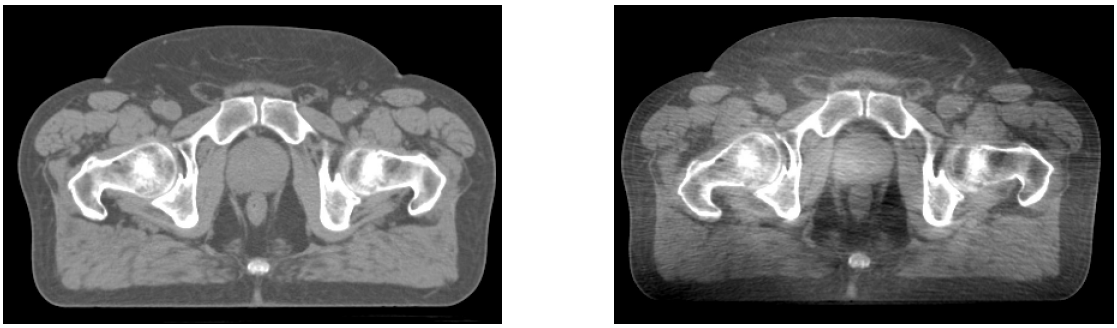


Figure 3.3: Schematic of a classical fan beam CT scan (left) and a CBCT scan (right). Credits : [16].



(a) Slice of CT scan.

(b) Slice of a CBCT scan.

Figure 3.4: Comparison of CT and CBCT scans of the pelvis. Credits : [28]

But CBCT has several drawbacks. It provides a poor contrast resolution which means some softer tissues cannot be viewed. It is also more likely to encounter artifacts including noise, beam hardening, and scattering. In particular, scattering is an important limitation that could rule out the use of CBCT for radiotherapy [28]. Figure 3.4 shows the differences between a CT and CBCT scan.

3.2 3-dimensional data set and pre-processing

Our database is composed of 76 3-dimensional CT scans and 55 3-dimensional CBCT scans from 76 different patients with prostate cancer. The files were originally encoded in `DICOM` format and converted in `MAT` format. Each scan comes with associated segmentation masks of bladder, rectum and prostate also in `MAT` format.

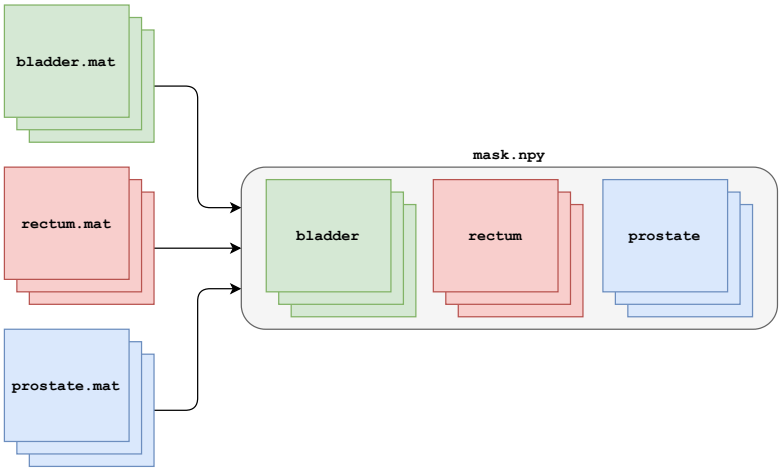


Figure 3.5: Illustration of the mask.npy file encoding.

The `CT.mat` and `CBCT.mat` files are stored in HU and the segmentation masks are binary matrices of the same size as the original scan where a voxel is 1 if it belongs to the contour and 0 otherwise. The CBCT scans have a height and width of 192 pixels and a depth of a 160 slices while the CT scans have a height and width of 500 pixels and various depths ranging from 226 to 555.

To provide easier usage of the data, the `.mat` files were divided into two `.npy` files each, one for the scan itself and one the segmentation mask of all the associated organs, stored as a 4-dimensional array (3 dimensions for the mask and 1 dimension for the organs as illustrated in figure 3.5).

3.3 2-dimensional data set extraction from the original data

Some of the algorithms that were designed or used require 2-dimensional data as input. For that reason, we built a 2-dimensional data set by extracting each slices and corresponding segmentation mask of the 3-dimensional CT scans. Doing so gave us access to a total of 7019 images.

In order to spare some memory on the hard disk, the 3-dimensional images are processed to take values in the range $[0; 255]$ instead of the original Hounsfield

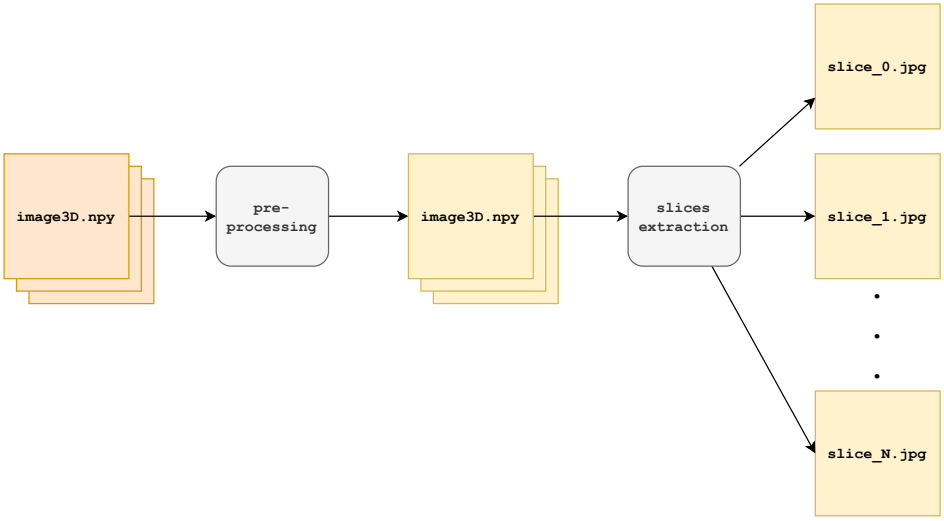


Figure 3.6: Schematic of the extraction process of the 2-dimensional data set from the original data.

Units. For an image that has P pixels, the value of pixel p_i becomes

$$p_i = \frac{p_i - \min_j(p_{j \in P})}{\max_j(p_{j \in P}) - \min_j(p_{j \in P})} \cdot 255 \ , \forall i \in P$$

The slices are then then encoded as `.jpg` files. The whole extraction process appears on figure 3.6.

3.4 Summary

The most common medical imaging techniques for visualizing the patient's anatomy in the scope of EBRT are CT and CBCT. They are both based on x-ray propagation through the patient's body. The CBCT scans are of lower quality than CT scans but they are faster and more affordable (section 3.1). Our data set is composed of 76 CT scans and 55 CBCT scans in 3 dimensions, from 76 different patients with prostate cancer (section 3.2). Some of the algorithms that were designed or used require 2-dimensional data as input. We extracted each slice of the 3-dimensional CT scans to create a 2-dimensional data set containing 7019 images (section 3.3).

Chapter 4

Methods

In chapter 3, we talked about the imaging techniques and the resulting data that was used in the EBRT treatment workflow. We also described the data sets that were available to us. In this chapter, we introduce the methods that were used in order to build a detection system for that data. Section 4.1 focuses on one classical and one new method of computer vision while section 4.2 introduces YOLO, a deep learning based method. The details of implementation are available in section 4.3.

4.1 Computer vision and classical methods

Computer vision is a field of artificial intelligence that deals with the techniques and algorithms a computer can use to understand and extract information from digital images. Image processing, on the other hand, makes use of computers to process digital images. The applications of computer vision and image processing are very diverse, ranging from self-driving cars to medical diagnostics as we shall see. For many years, scientists have studied the theory and application behind those concepts. The following section presents two algorithms. The first was introduced by Horowitz and Pavlidis in 1976 and has proven efficiency on some simple segmentation tasks [19]. It is meant to serve as a simple baseline for our problem. The second is an innovation that we suggest. It is based on the former method but is less complex.

4.1.1 Split and merge segmentation

The split and merge segmentation algorithm is an image processing technique that seeks to segment an image by recursively split the image and merge the similar parts of the image together, based on a homogeneity criterion.

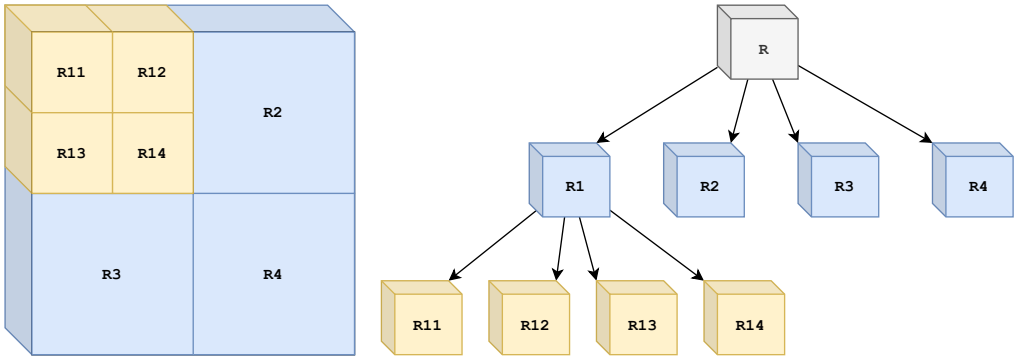


Figure 4.1: Representation of the split method with image regions and associated quadtree for 3-dimensional images.

The split method for segmentation begins with the entire image as the initial segment. Then it successively splits each current segment into quarters if the segment is not homogeneous enough [39]. Practically, each quadrant is a node of a quadtree and each node is split if it is not homogeneous to its parent node as illustrated by figure 4.1. Once the split method is complete, we need to check each region one by one and merge the neighbouring regions that are homogeneous. We then recompute the homogeneity criterion on the merged regions and repeat the process until no more regions can be merged.

There are many ways to define homogeneity, a common one is to use intensity (pixel value) as reference. Let $\bar{I}_i$, σ_i be the mean intensity and its standard deviation of region i. We say that regions r_1 is homogeneous to region r_2 if

$$|\bar{I}_1 - \bar{I}_2| \leq k\,\sigma_2 \tag{4.1}$$

where k is a threshold [32].

Merging two regions can also take various forms. We decide to implement the merging as a squared average intensity of the two regions. Mathematically,

$$I_{merged} = \left[\frac{\bar{I}_1 + \bar{I}_2}{2}\right]^2 \tag{4.2}$$

for all pixels in the merged region.

We work directly with the 3-dimensional images and treat it as if it was 2-dimensional. This means each region is 3-dimensional and a quadrant is only define

by is height and width while the depth remains constant. This can also be easily visualized in figure 4.1. Algorithm 1 shows a pseudo-code implementation of the split and merge method.

Algorithm 1 Split and Merge

```
ProcessList ← Image
while ProcessList not empty do
  Region ← ProcessList.pop()
  if ProcessElem is homogeneous then
    add Region to RegionList
  else
    split Region in 4 sub-regions
    add the sub-regions to processList
  end if
end while
while some regions can still be merged do
  Region1 ← RegionList.pop()
  for Region2 in RegionList do
    if Region1 and Region2 are homogeneous and neighbours then
      merge Region1 and Region2
      remove Region1 and Region2 from RegionList
      add merged region to RegionList
    end if
  end for
end while
```

Although this method can be useful for identifying dense and homogeneous zones in an image, it is limited for our purpose as it does not define an area of interest on its own. The next method was designed to overcome this issue.

4.1.2 Split and crop segmentation

The split and crop method is a new algorithm that was designed by the author. It takes strong inspiration from the split and merge method but is more straightforward. Instead of recursively splitting the image based on homogeneity, we start by diving the image into N equal slices vertically. We compare each slice to the whole image using equation 4.2 as a criterion. We label each slice with true or false based on whether the slice is homogeneous to the image or not. Finally, we discard all the slices on the side until we have at least M consecutive homogeneous ones.

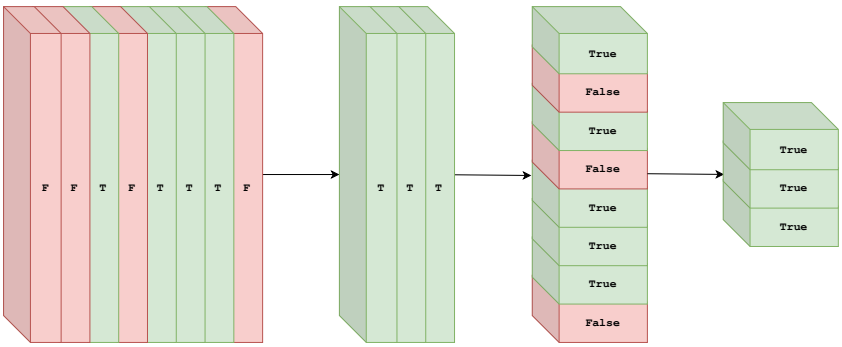


Figure 4.2: Representation of the split and crop method with $M = 3$ and $N = 8$ (we divide the image in 8 slices and discard all slices until 3 homogeneous slices are found).

We repeat the algorithm for horizontal slices to get our final cropped image. This process is illustrated in figure 4.2.

As for the split and merge method, we worked directly with 3-dimensional images and kept the depth constant for each slice. This is also represented visually in figure 4.2. Algorithm 2 shows a pseudo-code implementation of the split and crop algorithm.

Algorithm 2 Split and Crop

slice image in n regions.
for $r \in [0; n[$ **do**
  check if r is homogeneous.
  label r with true if it is, with false if it is not.
end for
crop image from each side until at least m regions labelled true are found.

Although these methods can be efficient for some simple tasks, they are limited when the task at hands becomes more complicated. Section 4.2 introduces a modern approach to solving our problem, which can help us overcome these limitations.

4.2 Deep learning detection system

The problem we are trying to solve here is complex. In addition to that, the data we are working with can be extremely homogeneous. Accordingly, classical methods based on simple computer vision concepts can lack efficiency and robustness for this kind of problem. But modern methods have recently pushed the boundaries of performing hard tasks such as this one. It is quite naturally that machine learning, and specifically deep learning, came to mind as the next step. The ability of deep learning to learn intricate structures and extract features from them makes it more suited to our problem the the former methods.

4.2.1 Unified detection

The detection system that we use is based on YOLO, a method for object detection developed by Joseph Redmon and Ali Farhadi from the University of Washington and Allen Institute of Artificial Intelligence. This paper first came out in 2015 and was followed by a second version[1] in 2017. The system described in the second article has been our model of choice because of its accuracy and its speed, but also because it was easy to work with, which differentiated it from other already existing detection systems.

This model frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. This means that a single NN predicts both the bounding boxes and the class probabilities. The network uses features from the entire image to predict each bounding box and predicts all of them simultaneously. This enables end-to-end training and real-time speed while maintaining high average precision.

4.2.2 Data normalization

Data normalization is the first step to take before performing any computation on the data. It is crucial to obtain good results as well as to fasten significantly the calculations [42]. The idea is to make the data set follow a normal distribution by removing the mean and dividing by the standard deviation. Mathematically, let μ be the mean and σ the standard deviation of all the pixels in the training data set.

[1] A third incremental version was released in 2018 but the changes were not as significant as the ones that were made between the first and the second version.

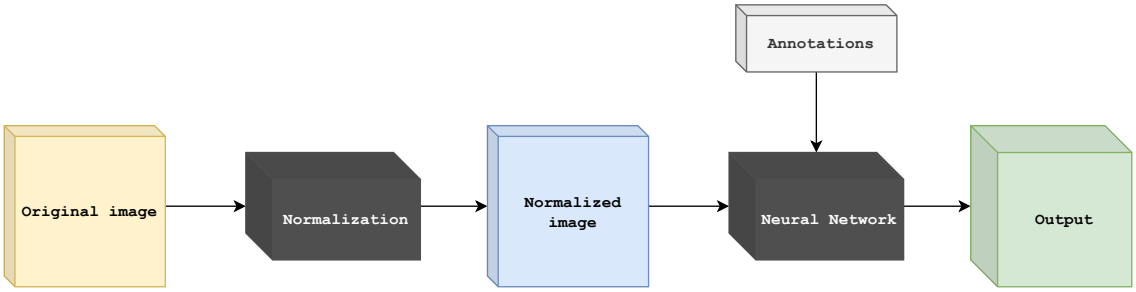


Figure 4.3: Schematic of the overall workflow.

Then for every pixel p of each image of the sets (training, validation and test), its normalized counter part is defined by :

$$p_{\text{norm}} = \frac{p - \mu}{\sigma} \tag{4.3}$$

Once the data has been normalized, we can pass it as input of our NN.

4.2.3 Design

The model is implemented as a CNN. It is made of 5 maximum pooling layers and 24 convolutional layers, all followed by a batch normalization layer and leaky Rectified Linear Unit (ReLU) except for the last one. Leaky ReLU is defined as follows :

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \tag{4.4}$$

Table B.2 shows the NN architecture. Table B.1 shows the model summary in terms of how it is defined by keras' `model.summary()` method.

The network takes a $416 \times 416 \times 3$ (RGB) image as input. Images that are of a different shape are resized (using bicubic interpolation) by the algorithm before they are fed to the network. The convolution layers work as features extractors while the batch normalization layer improves convergence and eliminates the need for other forms of regularization [35].

The convolutional layers downsample the image by a factor 32 so the input image gives us an output feature map of 13×13 cells ($S = 13$). Each cell is

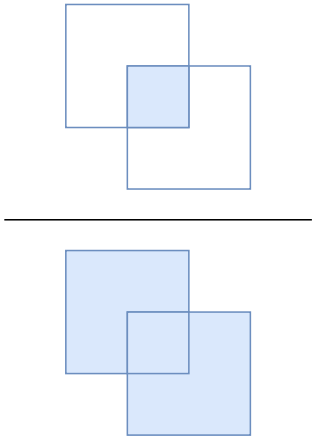


Figure 4.4: Representation of the IoU metric.

associated with $B = 5$ boxes. Each bounding box consists of 5 predictions : x,y,w,h and confidence. The (x, y) coordinates represent the center of the box and (w, h) are its width and height. The confidence score reflects how confident the model is that the box contains an object as well as how accurately it thinks that box is located. Formally we define confidence as $P(object) * IoU_{pred}^{truth}$, where $P(object)$ is the probability that there is an object and IoU_{pred}^{truth} is the Intersection over Union between the ground thruth and the predicted box. The IoU is defined as the ratio between the area of intersection of the two boxes and the union of their areas, as shown in figure 4.4. Each grid cell also predicts C conditional class probabilities, $P(Class_i|Object)$ which is the probability the the object belongs to $Class_i$ knowing there is one. At test time we multiply the conditional class probabilities and the individual box confidence predictions,

$$P(Class_i|Object) * P(Object) * IoU_{pred}^{truth} = P(Class_i) * IoU_{pred}^{truth} \tag{4.5}$$

which gives us class-specific confidence scores for each box [34]. In the end, a box is defined by $b = (x, y, w, h, o, c_i)\forall i \in C$. The predictions are thus encoded as a $S \times S \times B * (5 + C)$ tensor (the boxes are flatten in the last dimension) where S is the grid size, B is the number of boxes per cell and C is the number of classes we are trying to predict. In our case $S = 13$, $B = 5$ and $C = 3$ for bladder, rectum and prostate.

Antoher important aspect of the network architecture is the *passthrough layer*. As we mentioned previously, this model predicts detections on a 13×13 feature map. While this is sufficient for large objects, it may benefit from finer grained features for localizing smaller objects. That is why a *passthrough layer* is added

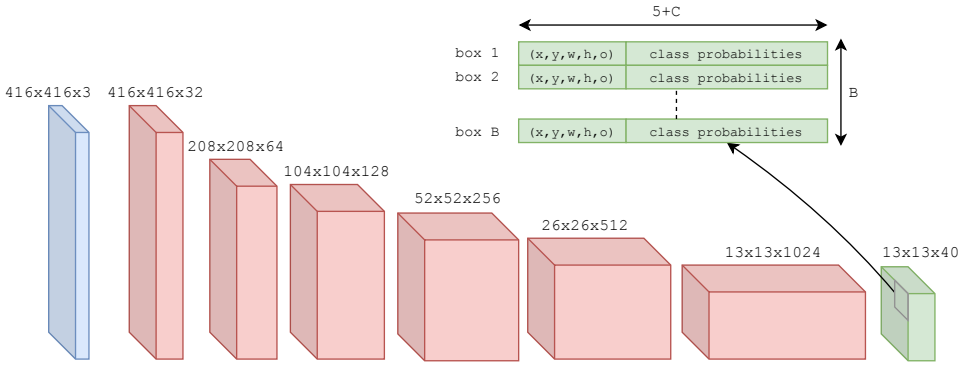


Figure 4.5: Simplified schematic of the network's components. The green block on top of the image represents a single cell of the grid's output. Although it is depicted as a 2-dimensional array for the sake of clarity, all that information is contained in a 1-dimensional array of length $B \times (5 + C)$.

to the network, bringing features from an earlier layer at 26×26 resolution. The layer concatenates the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations, similar to the identity mappings in ResNet ([18], [35]).

4.2.4 Anchor boxes and direct location prediction

The network predicts 5 bounding boxes at each cell. But if those predictions can be placed anywhere on the image, there is a risk that the model becomes unstable. To overcome this issue, each cell is associated with $B = 5$ anchor (or prior) boxes of pre-defined height p_h and width p_w[2]. Given those anchor boxes, the model predicts offsets (t_x, t_y) and scales (t_w, t_h). We use these predictions along with the prior boxes to compute the final bounding box parameters. Practically, if the cell is offset from the top left corner of the image by (c_x, c_y) then the box parameters are

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$b_o = \sigma(t_0)$$

[2]The dimension of those boxes are defined by running k-means clustering (where $k = B$) on an image data set's bounding boxes dimensions.

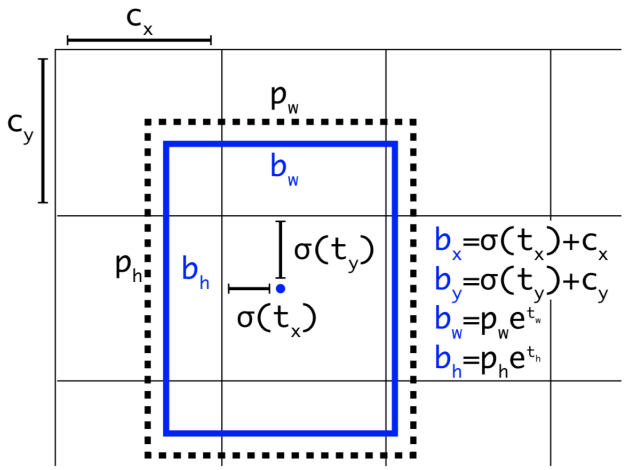


Figure 4.6: Bounding boxes with dimension priors and location prediction. Credits : [35]

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid activation function. Figure 4.6 illustrates this process.

4.2.5 Loss function

As we have already seen in section 4.2, any supervised learning algorithm has an associated loss function $\mathcal{L}$. This loss function maps the outcome of the network to a real number,

$$\mathcal{L} : S^2 \times (B + C) \longmapsto \mathbb{R} \tag{4.6}$$

the network then modifies its internal parameters in order to minimize this number. We need to find a loss function that translates our expectation of the algorithm in terms of mathematics.

Before we dig into the function itself, we need to define some expressions.

$$\mathbb{1}_{ij}^{obj} = \begin{cases} 1, & \text{if box } j \text{ from cell } i \text{ is responsible for detecting an object.} \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

$$\mathbb{1}_{ij}^{found} = \begin{cases} 1, & \text{if box } j \text{ from cell } i \text{ found an object.} \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

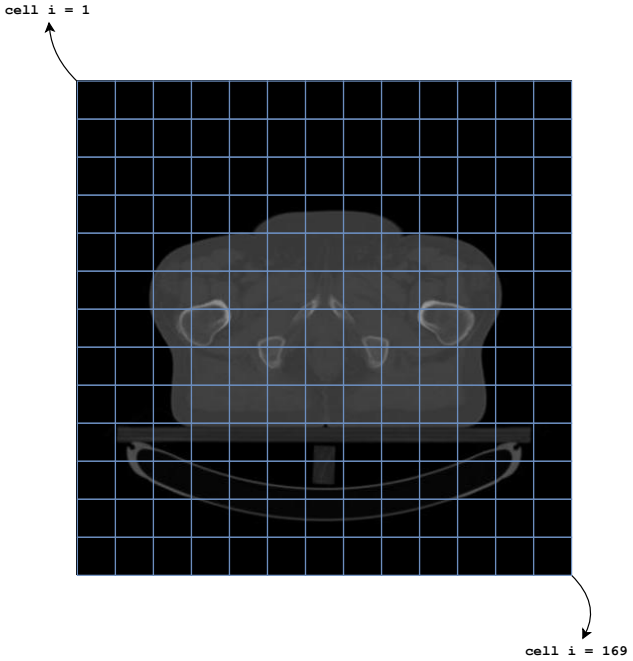


Figure 4.7: Visualization of the 13×13 grid on an image from our data set.

In practice, a box is responsible for detecting an object when that box has the best IoU score with the object. We also consider that a box has found an object when it has an IoU score over 0.6.

We define three distinct components for our function : coordinates loss, classification loss and confidence loss. Each component is responsible for detecting a different kind of error :

1. Coordinates loss : responsible for bounding box location errors.
2. Classification loss : responsible for classification errors, namely if an object from class i is classified as being an object from another class j.
3. Confidence loss : responsible for identifying when the model finds an object but there is none and vice-versa.

Let's recall that we predict $S^2 \times B$ vectors $(\hat{x}, \hat{y}, \hat{w}, \hat{h}, \hat{o}, \hat{p}(c_1), ..., \hat{p}(c_C))$ and that we have some ground truth bounding boxes defined by $(x, y, w, h, c), c \in [1, C]$.

$$\mathcal{L}_{coord} = \lambda_{coord} \cdot \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} \cdot \left[(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2 + (w_{ij} - \hat{w}_{ij})^2 + (h_{ij} - \hat{h}_{ij})^2 \right] \quad (4.9)$$

$$\mathcal{L}_{class} = \lambda_{class} \cdot \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} \cdot \sum_{c \in classes} \Big(p_{ij}(c) - \hat{p}_{ij}(c) \Big)^2 \tag{4.10}$$

$$\mathcal{L}_{conf} = \Big[\lambda_{obj} \cdot \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} \cdot (1 - \hat{o}_{ij})^2 \Big] + \Big[\lambda_{noobj} \cdot \sum_{i=1}^{S^2} \sum_{j=1}^{B} (1 - \mathbb{1}_{ij}^{found}) \cdot (1 - \mathbb{1}_{ij}^{obj}) \cdot (\hat{o}_{ij})^2 \Big] \tag{4.11}$$

The λ parameters operate like weights, they enable to give more or less importance to a certain component in $\mathcal{L}$. For instance we do not want to weight localization and classification errors equally as we want to maximize average precision. So we should keep $\lambda_{coord} > \lambda_{class}$. Also, in every image, most of the gird cells do not contain any object. Those cells have a confidence score of 0 which can overpower the gradient of the cells that actually contain an object. Thus we should keep $\lambda_{obj} > \lambda_{noobj}$. Note that

$$p_{ij}(c) = \begin{cases} 1, & \text{if an object of class } c \text{ falls into box } j \text{ from cell } i \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

but $\hat{p}(c)$ corresponds to the class probability of c and can take any value between 0 and 1. The final loss function is a linear combination of those components.

$$\mathcal{L} = 0.5 \cdot (\mathcal{L}_{coord} + \mathcal{L}_{class} + \mathcal{L}_{conf}) \tag{4.13}$$

The loss function only penalizes localization and classification errors if the box is responsible for finding the box [34]. The confidence loss is increased only if the box is responsible or if there is no object but it still found one.

4.2.6 Threshold filtering and non-max suppression

When we run the model, it predicts a large number of boxes everywhere across the image. We carry out two steps to get rid of the boxes that are duplicated or less likely to have detected an object.

The first step is threshold filtering. That is, we set a confidence threshold and discard all the boxes that are under that threshold. At this stage, the threshold can be quite low because we do not want to take any risk and discard a box that is actually detecting an object with low confidence. This step serves as a first clean up.

Because of this low threshold, there can still be a lot of remaining overlapping boxes. This is why we apply the non-max filtering. The key of non-max filtering resides in the concept of IoU, which is the metric that we already talked about in section 4.2.3 and that we will also use in chapter 5.

We use the IoU to execute non-max suppression in the following way :

1. Select the box with highest confidence
2. Discard all boxes that have an IoU above a certain threshold with this box.
3. Loop by selecting the box with the next highest confidence until there is no more box to visit.

This closes the sections on technical features. Many other models that perform object detection are available. Section 4.2.7 explains how we extend this 2D model to 3D.

4.2.7 3-dimensional bounding box extraction

As we have seen throughout the previous sections, YOLO is a model that works with 2-dimensional images. Therefore we need to extend this model to reconstruct 3-dimensional images from the slice predictions. To achieve this we perform the following steps :

1. Gather all slices from the same 3-dimensional image.
2. Feed the slices one by one to the network to perform predictions on each organ.
3. Discard all boxes that have a confidence score lower than a threshold γ.
4. Extract the 3-dimensional bounding box by taking the maximum corner values of all boxes (so every box from every slice above the threshold fits in the resulting box).

Those steps are shown in figure 4.8.

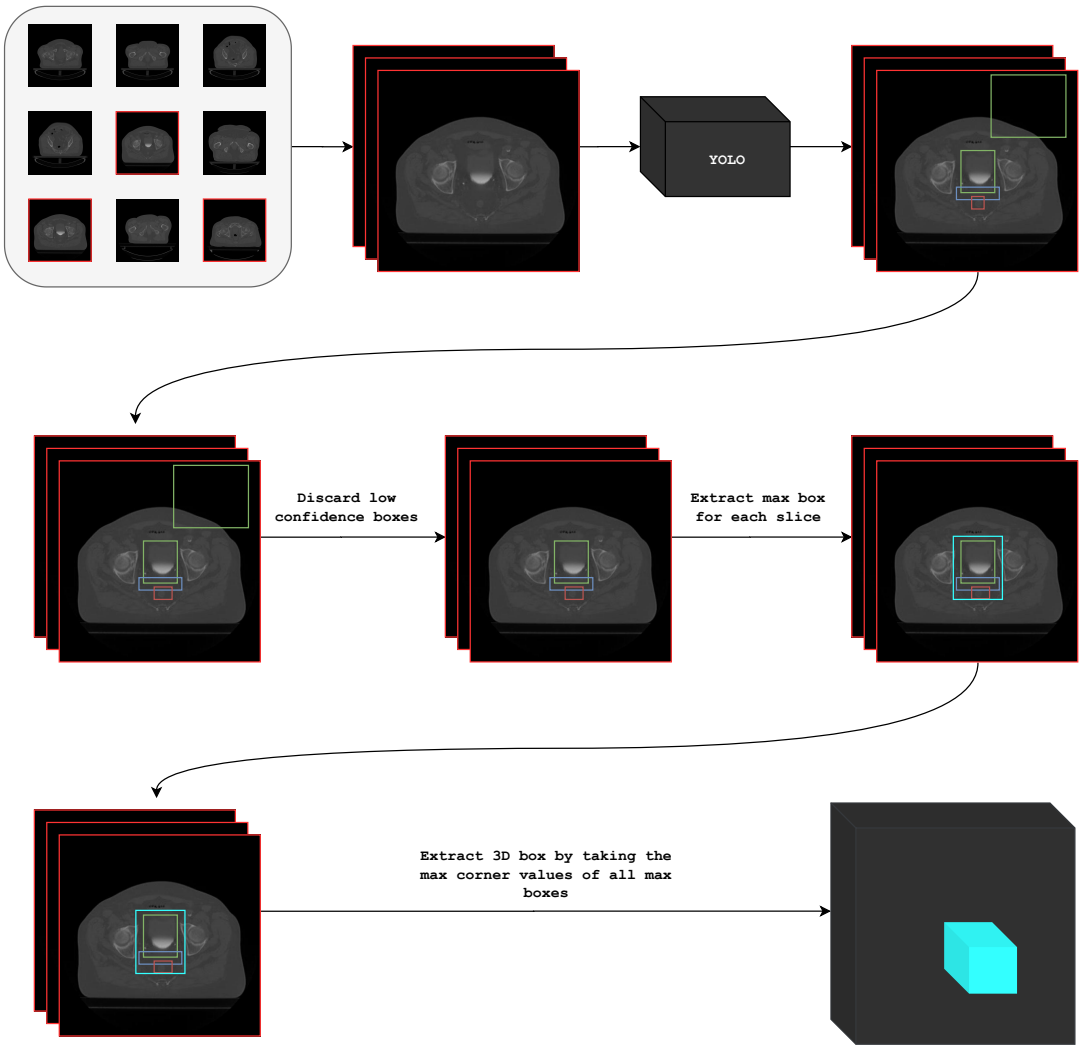


Figure 4.8: Visual representation of the extraction process for a full 3-dimensional image. We start by gathering the slices from a same patient and order them. We feed them individually to our neural network which performs extraction on each organ separately. We discard the boxes with low confidence and merge the boxes from the organs into one common box per slice. We then combine all the boxes from the different slices together. In the end, the blue box should contain all the organs in order to be input to a segmentation algorithm. It should also be small enough so we can fit several of them in the GPU.

4.2.8 Particularity of medical data

Another aspect to consider is that YOLO was originally made to perform detection on coloured images with high contrast. Most of which a human can just glance at and already know what objects are in it. On a purely visual standpoint, our data set is much more challenging. In fact, only trained doctors or people that are very familiar with that kind of images are able to find the organs and their locations within those images. That being said, our data set only contains three classes : bladder, rectum and prostate. Whereas the original model is made to detect hundreds of classes. That makes the model far less likely to misclassify the objects that he finds on our data set. To deal with both of these particularities, we choose different values for the λ parameters of the loss function until we find the configuration that best fits our data.

4.3 Implementation

The implementation of the above-mentioned methods as well as an explanation on usage with custom data is available at `https://github.com/lucaderumier/pelvis_detection`. Appendix C.1 explains the modifications we made to the original implementation in more details.

4.4 Summary

In this chapter, we presented the methods that were used. In section 4.1, we explained the split and merge segmentation method (section 4.1.1) and we introduced the split and crop, a new algorithm (section 4.1.2). The problem with these methods is that they lack of accuracy when dealing with low contrast medical data. Deep learning can discover intricate structures to help us overcome this issue (section 4.2). We decide to use YOLO to solve our problem (section 4.2.1). As part of our method, we use a complex pipeline that is made of several components such as data normalization (section 4.2.2), neural network (section 4.2.3), anchor boxes (section 4.2.4), loss function (section 4.2.5), threshold filtering and non-max suppression (section 4.2.6). In order to perform 3D extraction, we extend the base model (section 5.3.3). Finally, we covered the particularities related to our data (section 4.2.8) and the details of implementation (section 4.3).

Chapter 5

Results

In this chapter, we present the tests and results from the methods described in chapter 4. We introduce the metrics that were used in order to evaluate the algorithms' performances (section 5.1) then we detail the testing process for the classical and computer vision methods (section 5.2) and the modern deep learning approach (5.3).

5.1 Metrics

Before jumping into the results, we want to define the metrics that we used and explain why we chose them. Evaluating the smooth running of a program can be a tricky task sometimes and we wanted to make sure that the way of computing our results was representative of how the algorithm was performing. We also want to give some intuition about what those metrics represent to help understanding the results.

The first metric, which we has already been explained in section 4.2.6, is the IoU. It helps us define the accuracy of our model in terms of localization. For the other metrics, we start by defining the confusion matrix as follows :

True positives Pixels that are in the ground truth bounding box and the predicted bounding box	**False positives** Pixels that are not in the ground truth bounding box but are in the predicted bounding box
False negatives Pixels that are in the ground truth bounding box but not in the predicted bounding box	**True negatives** Pixels that are not in the ground truth bounding box nor in the predicted bounding box

From there, we derive three other metrics of interest.

- Precision $= \frac{\text{True positives}}{\text{True positives+False positives}}$
- Recall $= \frac{\text{True positives}}{\text{True positives+False negatives}}$
- False negative rate $= \frac{\text{False negatives}}{\text{True positives+False negatives}}$

Precision represents the proportion of pixels identified as positive (inside the predicted bounding box) that actually were positives (inside the ground truth bounding box). Recall gives us the proportion of positive pixels (inside the ground truth bounding box) that were identified correctly by the system. The false negative rate is a bit more specific but very useful in this context. It gives the proportion of positives pixels that were missed by the system. This is important because we do not want to miss any part of an organ as this would make the organ's segmentation impossible.

5.2 Classical and computer vision methods

In this section we analyse and evaluate the results from the algorithms of section 4.1. Therefore we run some experiments on 55 CBCT scans all coming from different patients. The reason we use CBCT scans and not CT scans is because it was the only data that was available at this stage. We start by experimenting on the split and crop algorithm and extend our first experiment in order to evaluate the split and merge algorithm as well.

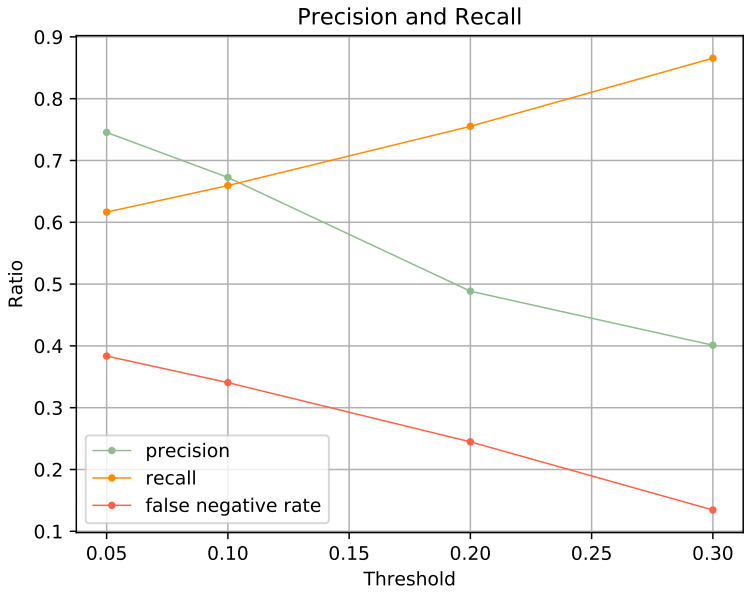


Figure 5.1: Evolution of the different metrics on average in terms of threshold for 55 CBCT scans coming from different patients.

5.2.1 Split and crop

In this section, we aim at analyzing the efficiency of the split and crop algorithm to extract a bounding box that contains the bladder and the rectum in a 3-dimensional CBCT scan. The reason we focus on the bladder and the rectum is because we were missing prostate annotations in the CBCT data set but also because it is usually sufficient for extracting a single bounding box. Indeed, the prostate is located between those two organs and will rarely exceed a box that was made to fit both bladder and rectum.

We run the algorithm on the 55 images, set $N = 100$ and $M = 20$ and make the threshold k from equation 4.2 vary. We record the results and evaluate the precision, recall and false negative rate. Figure 5.1 illustrates these results.

The CBCT scans all have dimensions $192 \times 192 \times 160$. From the 160 slices, approximately 40 contain information about the bladder and 50 contain information about the rectum. We select 18 slices out of those 160 that are representative in terms of organ's information and depict the ground truth and predicted bounding boxes on them. Figures 5.2, 5.3 and 5.4 illustrate this for thresholds equal to 0.05,

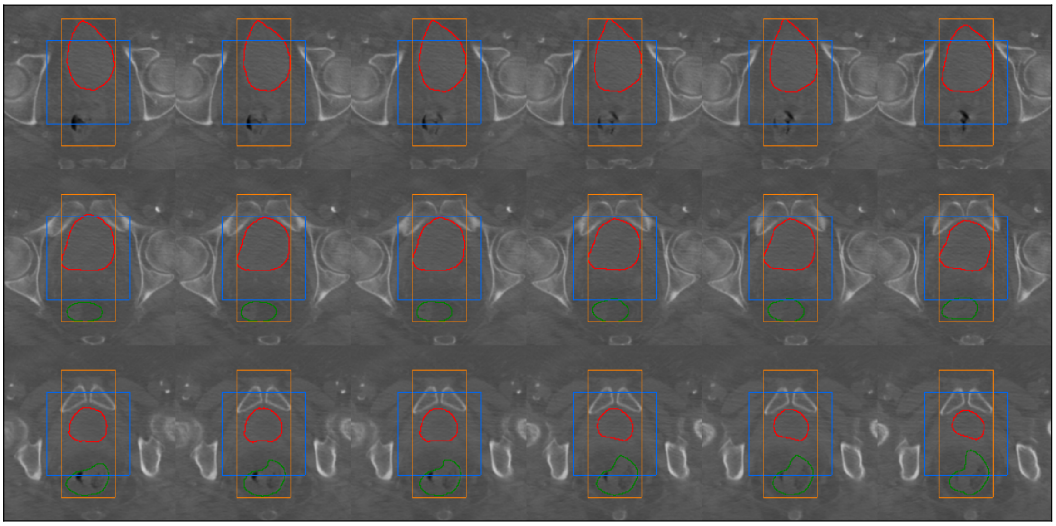


Figure 5.2: Results of the split and crop with threshold 0.05. The blue box is the result of the algorithm's cropping while the orange box is the ground truth bounding box. The precision for this image is 0.648, its recall is 0.59 and its false negative rate is 0.345 (computed for the 3D box corresponding to this patient). All slices in the image come from the same patient that has been chosen because it represents an average case.

0.1 and 0.3 respectively. The results are computed on the whole data set of CBCT scans.

5.2.2 Split and merge

In this section, we aim at analyzing the efficiency of the split and merge algorithm the same way we did for the split and crop. The difference is that split and merge cannot be used on its own to extract a bounding box. It is rather used to highlight similar regions in an image. With this in mind, we start by running the split and merge algorithm on our data set, then we run the split and crop on the resulting data the same way we did for the first experiment.

Figures 5.6, 5.7 and 5.8 illustrate the results for the same slices that were chosen in experiment one.

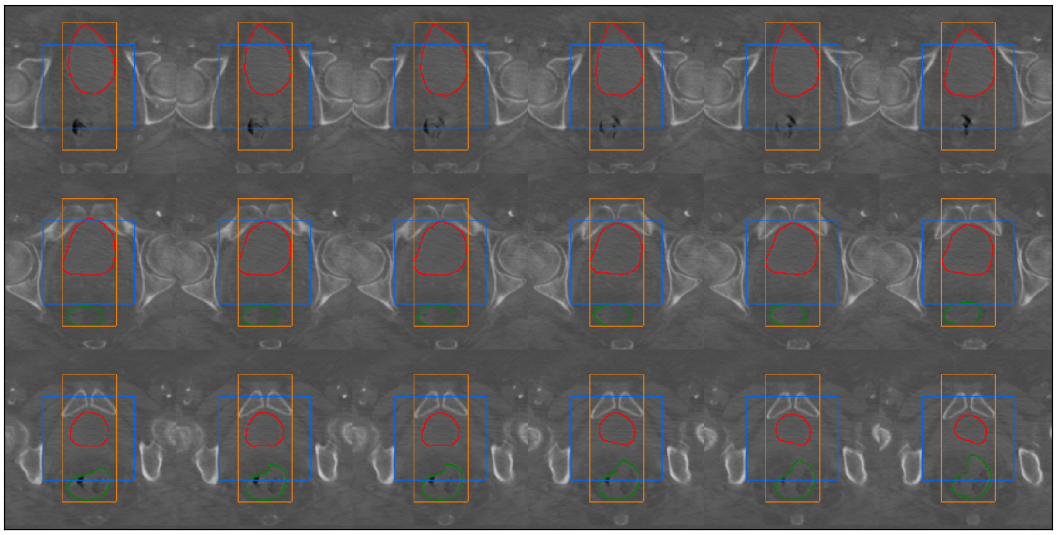


Figure 5.3: Results of the split and crop with threshold 0.1. The blue box is the result of the algorithm's cropping while the orange box is the ground truth bounding box. The precision for this image is 0.59, its recall is 0.654 and its false negative rate is 0.345 (computed for the 3D box corresponding to this patient). All slices in the image come from the same patient that has been chosen because it represents an average case.

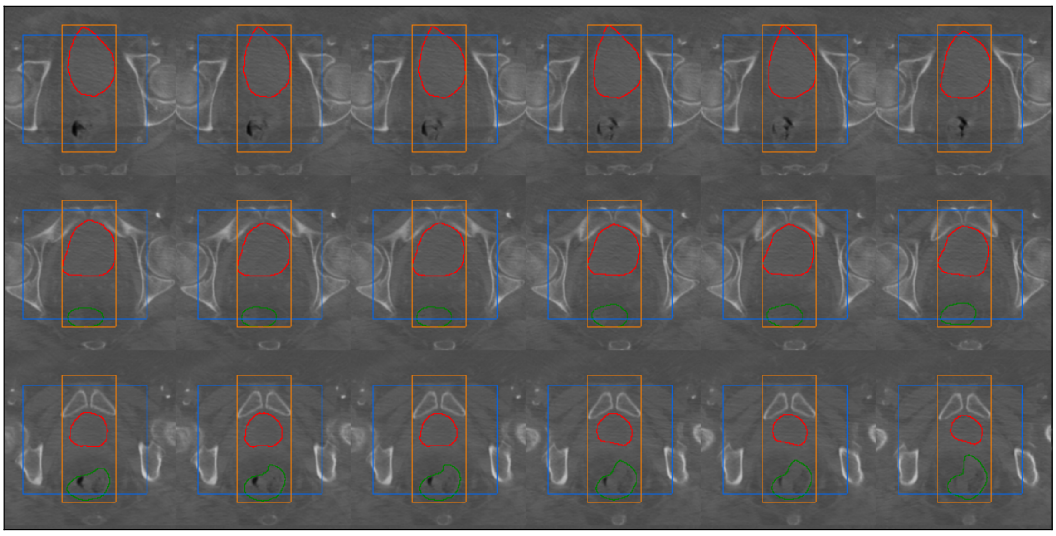


Figure 5.4: Results of the split and crop with threshold 0.3. The blue box is the result of the algorithm's cropping while the orange box is the ground truth bounding box. The precision for this image is 0.433, its recall is 0.856 and its false negative rate is 0.143 (computed for the 3D box corresponding to this patient). All slices in the image come from the same patient that has been chosen because it represents an average case.

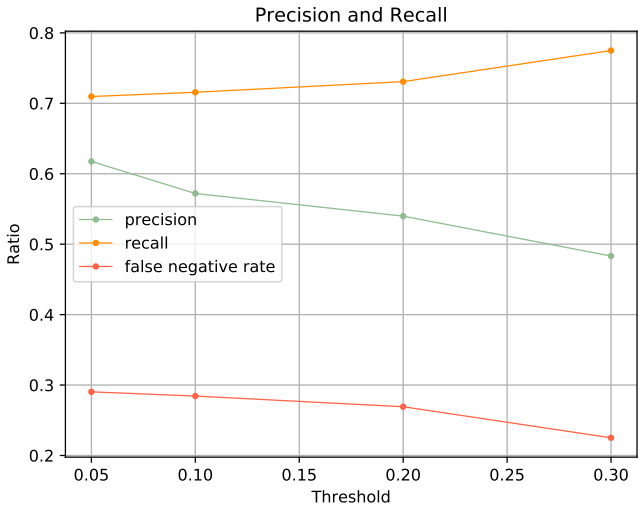


Figure 5.5: Evolution of the different metrics on average in terms of threshold for 55 CBCT scans coming from different patients.

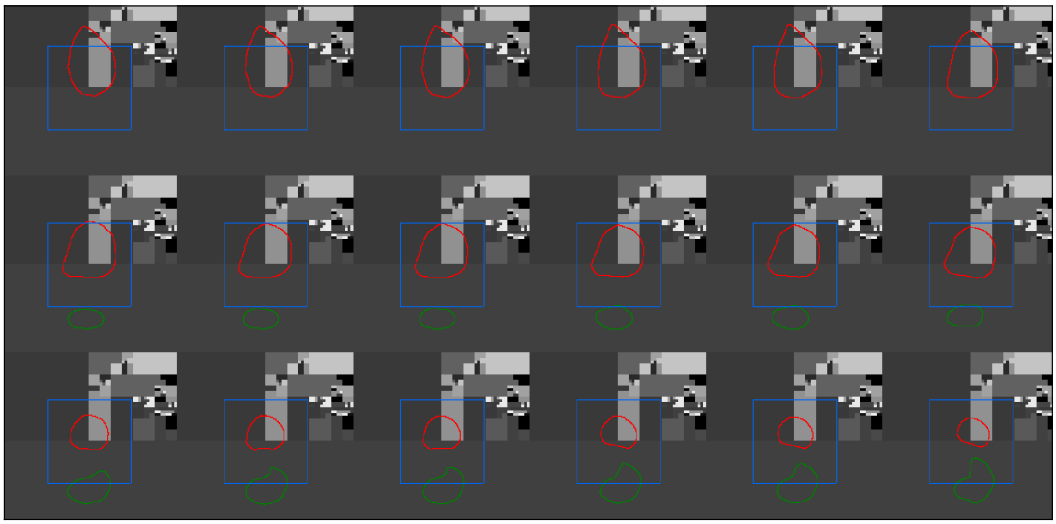


Figure 5.6: Results of the split and crop with threshold 0.05 following a split and merge. The blue box is the result of the algorithm's cropping. The precision for this image is 0.648, its recall is 0.654 and its false negative rate is 0.345 (computed for the 3D box corresponding to this patient). All slices in the image come from the same patient that has been chosen because it represents an average case.

	Precision	Recall	False negative rate
Split and crop	**0.745**	0.616	0.383
Split and merge	0.617	**0.709**	**0.290**

Table 5.1: Comparison of the results for a threshold $k = 0.05$.

5.2.3 Conclusion on classical methods accuracy

The results of those experiments highlight an obvious lack of accuracy when using these methods. A first important aspect that makes the algorithms obsolete for our purpose is that they always miss some part of the organs, whatever threshold we set. When looking at the results, we see that the bigger the threshold the less false negative pixels but also the more false positives. This is obviously related to the fact that a large threshold means a bigger surface inside the bounding box but we see that precision decreases (because of the increasing false positives) as the threshold increases which also proves the lack of efficiency. Table 5.1 compares the results of both methods.

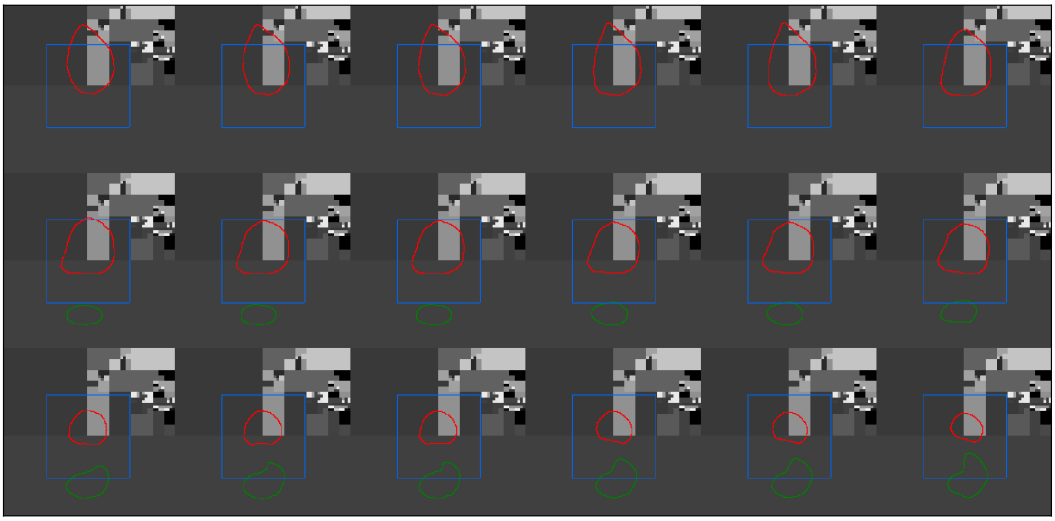


Figure 5.7: Results of the split and crop with threshold 0.1 following a split and merge. The blue box is the result of the algorithm's cropping. The precision for this image is 0.648, its recall is 0.654 and its false negative rate is 0.345 (computed for the 3D box corresponding to this patient). All slices in the image come from the same patient that has been chosen because it represents an average case.

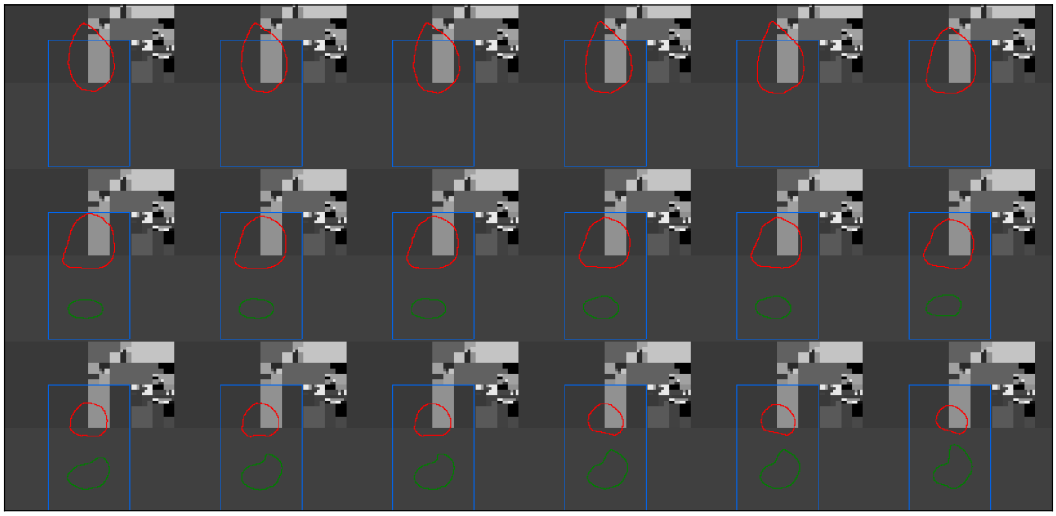


Figure 5.8: Results of the split and crop with threshold 0.3 following a split and merge. The blue box is the result of the algorithm's cropping. The precision for this image is 0.542, its recall is 0.848 and its false negative rate is 0.151 (computed for the 3D box corresponding to this patient). All slices in the image come from the same patient that has been chosen because it represents an average case.

5.3 Deep learning detection system

We have seen in the previous section that the classical and computer vision methods we tried out cannot be trusted to solve our problem. In this section, we analyse and evaluate the results we obtain from the detection system described in section 4.2. We start by expliciting the data sets and configuration of our model (section 5.3.1). We then run different experiments in order to find the best parameters and evaluate the performances for those parameters (section 5.3.2). We use the 3D extension of our model to extract boxes for each organ as well as all of them together and monitor the same performance indicators as before (section 5.3.3). We use those results to study the impact on memory savings (section 5.3.4). Finally, we compare our results with other methods from the literature (section 5.3.5).

5.3.1 Data sets and training configurations

In order to train the network, we retrieve 5400 images from the 2-dimensional CT data set[1] described in section 3.3 and split them intro three sets. The training set, containing 4500 images from 63 patients, and the validation and test set holding 450 images each coming from 6 and 7 different patients respectively. All of the 5400 images come with bounding box annotations for at least one organ (some of them contain a bladder, a prostate and a rectum, some of them only a rectum etc.).

In order to detect the range of accuracy in which our model can operate, we study the effect of the coefficients λ_{coord} and λ_{obj} described in section 4.2.5 on the model by making them vary. We also study the impact of training the model with custom anchor box sizes. We train the model using a learning rate of 10^{-3} and use Adaptive Moment Estimation (Adam) optimizer as well as a batch size of 16.

5.3.2 Performances of the model on each organ

To evaluate the model, we follow a simple procedure. We start by training it until convergence is reached in terms of IoU on the validation set. We pick the point in time where the model was best performing (also in terms of IoU on the validation set) and we use the corresponding weights to perform prediction on our test set.

We have four λ parameters : $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}})$. The initial hypothesis we make is that increasing a λ parameter should also increase the results in its

[1] In fact we retrieve the subset of the data that contains information for at least one organ. This enables to work with useful images only.

	IoU with VOC anchors	IoU with custom anchors
Bladder	0.226 ± 0.087	$\mathbf{0.689 \pm 0.237}$
Rectum	0.083 ± 0.076	$\mathbf{0.497 \pm 0.203}$
Prostate	0.197 ± 0.126	$\mathbf{0.447 \pm 0.233}$

Table 5.2: IoUs and standard deviations for $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (1, 1, 1, 1)$ of the best performing models evaluated on the validation set with and without custom anchors. We see that the model using custom anchors outperforms the one that does not on each organ by a significant amount.

domain of responsibility. That is, increasing λ_{coord} should give us less localization errors, increasing λ_{obj} and λ_{noobj} should improve the model in its capability of detecting whether there is or not an object and increasing λ_{class} should lead to less classification errors. We also assume that, as we have only three classes and the model is made to detect hundreds, λ_{class} will not have a significant effect on the results and thus can be fixed to 1. Same goes for λ_{noobj} because the boxes that detect an objet when there is none tend to vanish naturally as we train the model[2].

The other parameter we study is the pre-defined size of the anchor boxes. As mentioned in section 4.2.4, the system predicts 5 prior boxes of pre-defined height and width. Originally these heights and widths are defined by running 5-means clustering on the ground truth bounding boxes of the VOC data set. We also run a 5-means clustering on our data set to get custom anchor boxes ratios. We set $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (1, 1, 1, 1)$ and train our model with and without custom anchors during four hours (100 epochs). We compare the results as shown in table 5.2. We see that there is a notable difference between the model that uses custom boxes and the one that does not, which makes perfect sense as the custom boxes are exactly suited to our data set. Therefor we decide to keep the custom anchor boxes for any further experiment.

The second experiment is designed to study the impact of λ_{coord} and λ_{obj} on the IoU. We set both other parameters, λ_{noobj} and λ_{class}, to one as the hypothesis is that they should not have a significant impact on the results. For each configuration, we choose the model that has the best results in terms of average IoU of the three organs (on the validation set). We show these averages in figure 5.10. From the graph we can clearly identify that the best configuration is $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (10, 4, 1, 1)$. Therefore, we choose this configuration to evaluate our model.

[2]This is something we observed when running our experiments.

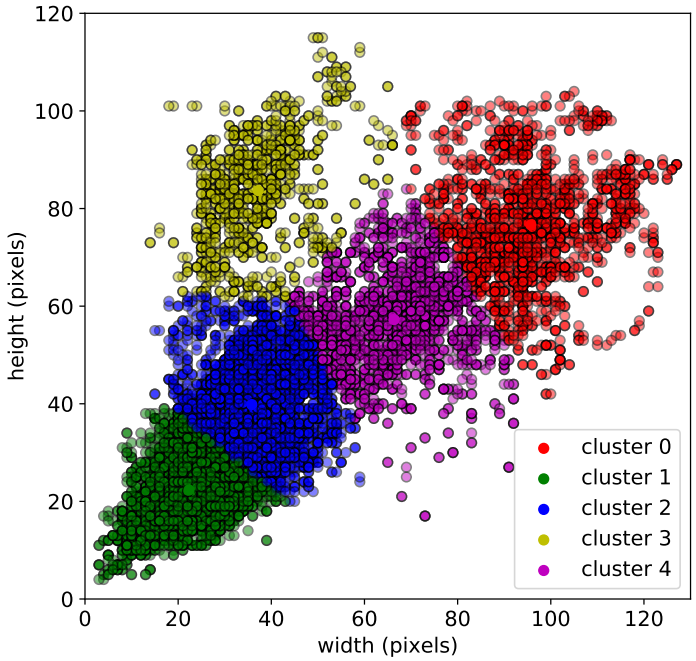


Figure 5.9: Representation of the dimension clusters for the anchor boxes. Each color is a cluster, the final height and width of the anchor boxes are the centroids of each cluster. These clusters are computed by running 5-means clustering on the (height,width) pairs of all boxes of the 7019 slices from the 76 patients of the data set.

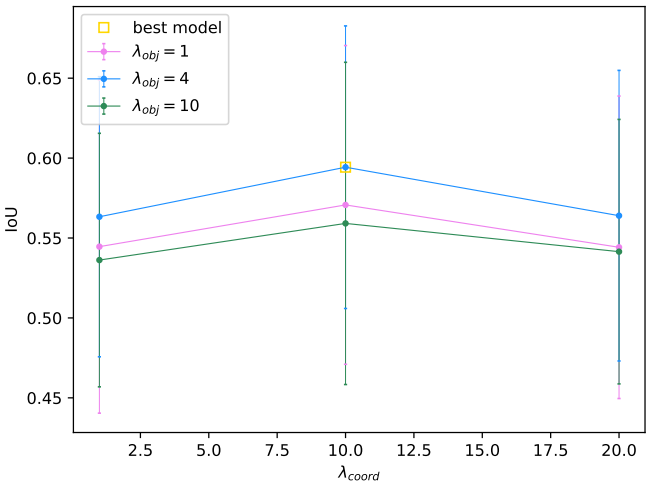
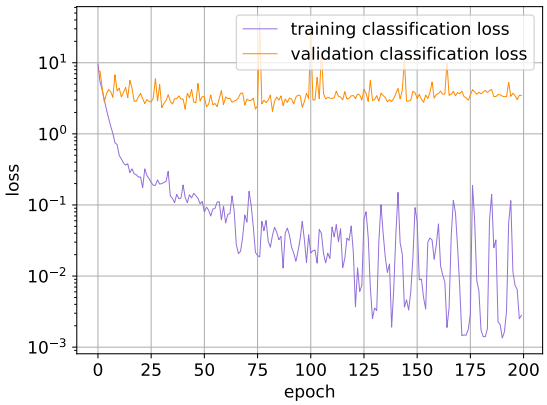


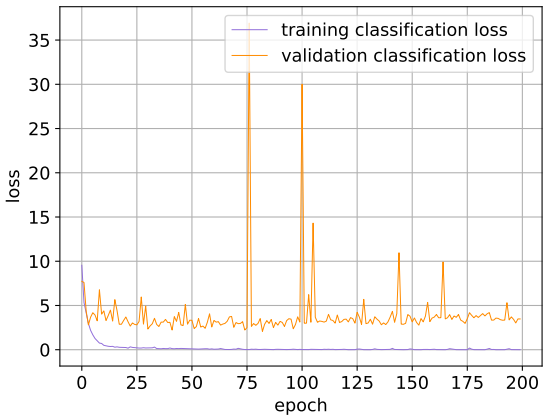
Figure 5.10: Evolution the IoU (with standard deviation) in terms of λ_{coord} and λ_{obj}. The IoU is computed as the average IoU on the validation set (450 images, 6 patients) of the three organs for each configuration. We see the best performing model is the one which has $(\lambda_{\text{coord}}, \lambda_{\text{obj}}) = (10, 4)$.

Figure 5.11 and 5.12 illustrate the classification, confidence, coordinates and total loss of the model with $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (10, 4, 1, 1)$ in terms of epochs. We train the model for eight hours (200 epochs) to make sure it has reached convergence in terms of IoU on the validation set.

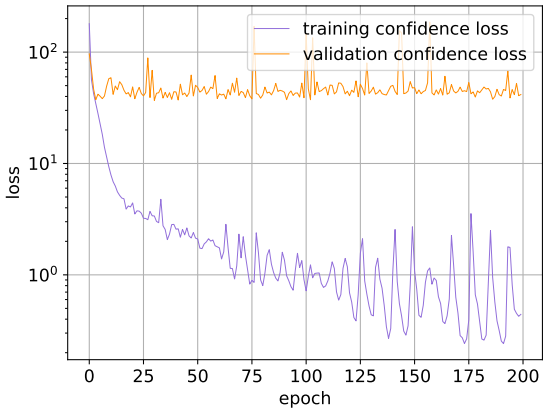
By looking at figures 5.11 and 5.12, we observe that the total loss' curve mainly follows the same path as the coordinates loss. This is obviously to be expected as we set λ_{coord} to 10, putting the coordinates way above the other components in the final loss. Another observation we make is that there is very few classification errors. This means when the system finds an organ, most of the time it is the right one, which verifies the assumption we made earlier. The last element that we notice is that the loss is unstable. It has large peaks for some epochs while being quite smooth for others. For this reason, we plot the graph with linear and logarithmic y axis as some of the linear graphs result in a straight line with peaks, and so no information can really be retrieved. We put some efforts into investigating why this was happening. One thing that we tried was to reduce the learning rate to 10^{-4} instead of 10^{-3} but the results were similar. Up to this day, the reason why these instabilities occur remains an open question. This being said, it does not really affect the final results as we are able to pick the best performing model out



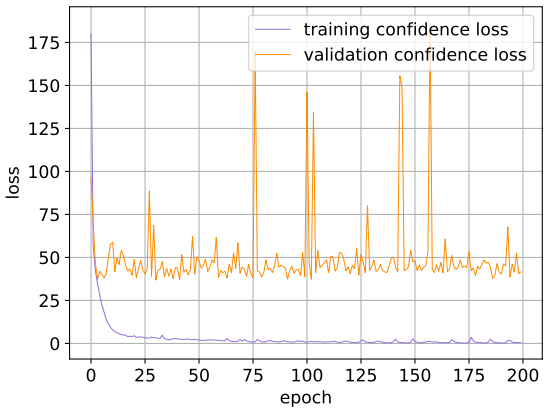
(a) Classification loss (log scale)



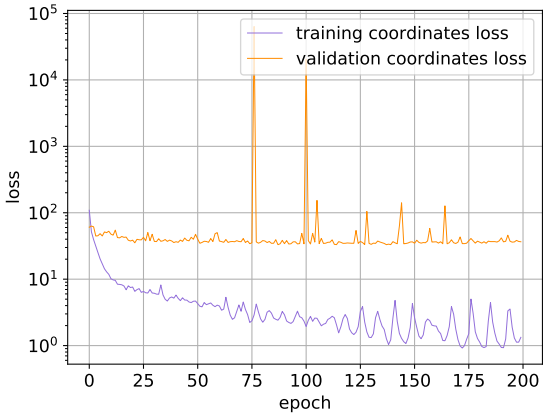
(b) Classification loss (linear scale)



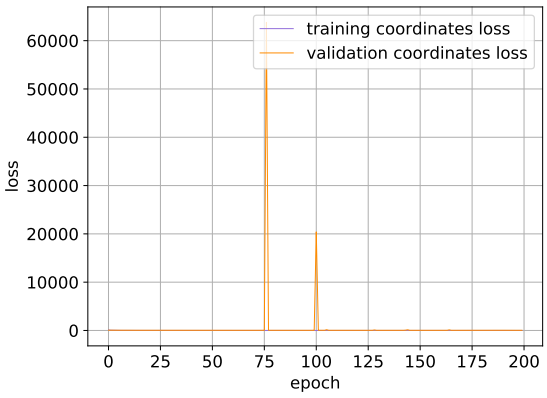
(c) Confidence loss (log scale)



(d) Confidence loss (linear scale)



(e) Coordinates loss (log scale)



(f) Coordinates loss (linear scale)

Figure 5.11: Loss' components as functions of time (epoch). The model is trained with $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (10, 4, 1, 1)$ and custom anchor boxes.

(a) Logarithmic scale

(b) Linear scale

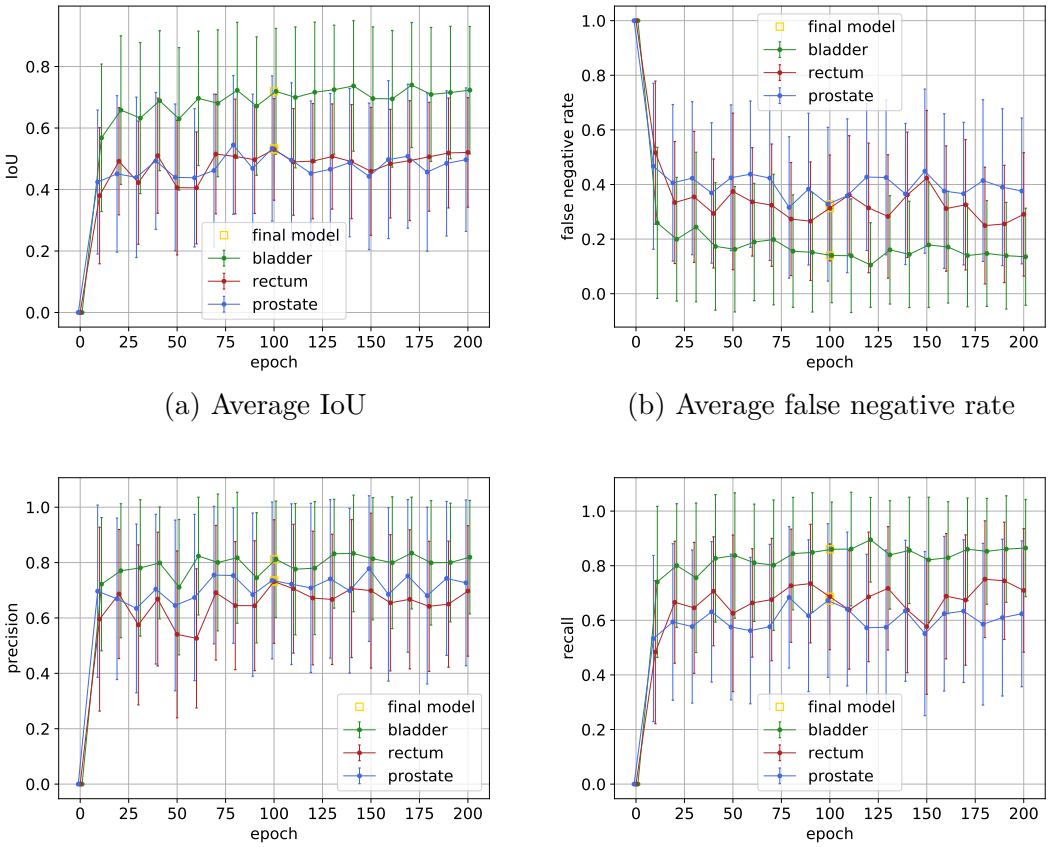
Figure 5.12: Final loss as a function of time (epoch). The model is trained with $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (10, 4, 1, 1)$ and custom anchor boxes.

of all the epochs.

Another interesting fact is that removing the non-max suppression module (section 4.2.6) and replacing it by a module that only keeps the highest confidence score for each organ drastically improves performances. We observe that the non-max suppression module tends to suppress prostate boxes as they overlap with bladder boxes. This is strange because in practice, an image can contain an object in front of another and the system should still detect it. Though, the way non-max suppression is implemented does not seem to allow that. So we remove the non-max suppression module and keep our highest-confidence module instead. We could also have modified the current non-max suppression by separating it for all the different classes or simply experiment on the threshold. We did not do any of those as the highest-confidence module was working just fine.

Every ten epochs, we monitor the performances of the model by evaluating its IoU, precision, recall and false negative rate, for each organ and on the validation set. Figure 5.13 illustrates those results.

We keep the model with the best performance in terms of average IoU of the three organs and we evaluate it on the test set. Table 5.3 sums up the performances of the model in terms of IoU, precision, recall, false negative rate and detection percentages, which is the percentage of images where the system found the organ when it was indeed there. We see that the bladder gets the best results in terms of all metrics but detection percentage. Figures 5.14 and 5.15 illustrate an example



(a) Average IoU

(b) Average false negative rate

(c) Average precision

(d) Average recall

Figure 5.13: Metrics with standard deviations for the three organs as functions of time (epoch). Those metrics were evaluated from the model with $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (10, 4, 1, 1)$ and custom anchor boxes on the validation set. The yellow square shows the results of model that was kept to perform the evaluation on the test set, which is the model that has the best average IoU on the validation set.

	IoU	Precision	Recall	FNR	Found
Bladder	$\mathbf{0.68 \pm 0.23}$	$\mathbf{0.82 \pm 0.22}$	$\mathbf{0.82 \pm 0.22}$	$\mathbf{0.18 \pm 0.22}$	87.5%
Rectum	0.50 ± 0.20	0.62 ± 0.25	0.75 ± 0.25	0.25 ± 0.25	**88.0**%
Prostate	0.54 ± 0.24	0.72 ± 0.25	0.68 ± 0.28	0.32 ± 0.28	86.9%

Table 5.3: IoU, precision, recall, false negative rate (with standard deviations) and detection percentage (percentage of the images where the system found the organ when it was indeed there) for all three organs evaluated on the test set's predictions of the models trained with $(\lambda_{\text{coord}}, \lambda_{\text{obj}}, \lambda_{\text{class}}, \lambda_{\text{noobj}}) = (10, 4, 1, 1)$ and custom anchor boxes in 2 dimensions.

of average and bad predictions. Appendix A contains illustrations for each patient of the test set. The average prediction is defined by a prediction that falls in the average IoU plus or minus standard deviation for all predicted boxes while a bad prediction is defined as being 0.35 under the average IoU for each organ. The average inference time for a single image is 0.02 second on the test set using the NVIDIA GeForce GTX 1080 GPU.

This model can be trusted to accurately identify the bladder but a single shot detection for rectum and prostate can sometimes be inaccurate. The next step is to extract our final box in 3 dimensions and thus see if merging information for several images produces better results. Section 5.3.3 addresses the tests and performances related to this extraction.

5.3.3 Performances on 3-dimensional box extraction

In section 5.3.2, we evaluated the performances of several models to identify and localize each organ independently. In this section, we use the model that performed best to extract a bounding box in 3 dimensions. The 3D images that we evaluate the model on are the full CT scans of the 7 test patients.

We start by running the algorithm described in section 4.2.7 on all 7 patients and collect the predicted bounding boxes (one for each organ and one for all of them together). To extend our results from section 5.3.2, we compute the usual metrics for each organ in 3 dimensions and report the results for the IoU in table 5.4. We see from the table that our model is performing better in 3 dimensions than previously for individual slices. This table also holds the results of the methods that we introduced in section 2.3 which we will discuss in section 5.3.5.

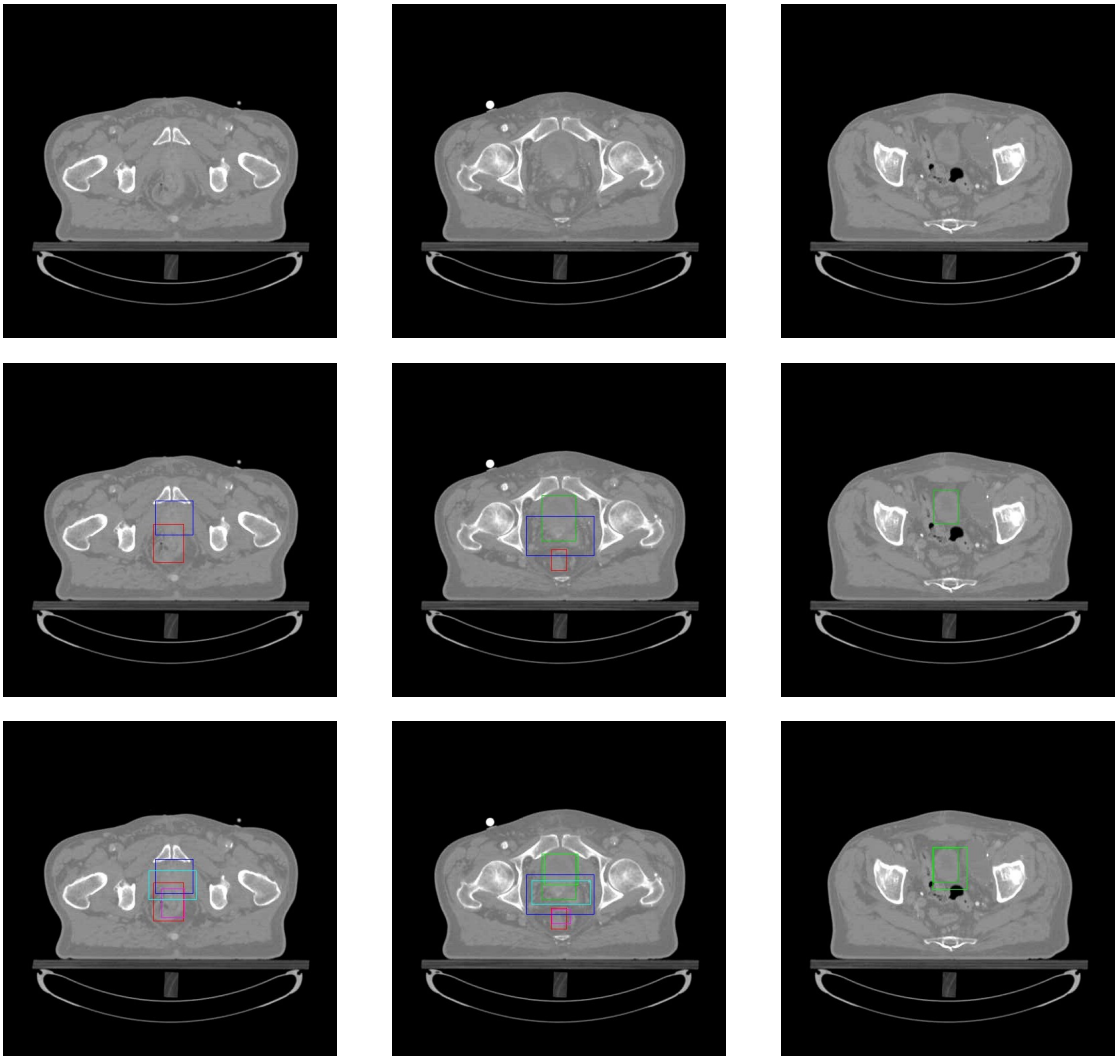


Figure 5.14: Visualization of average predictions on data from the test set for the best performing model. Each column corresponds to a slice of the same CT scan. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The bladder is in green, prostate in blue and rectum in red. The most left slice has IoU scores of 0.553 for rectum and 0.524 for prostate. The middle slice has IoU scores of 0.687 for bladder, 0.546 for rectum and 0.518 for prostate. The most right slice has an IoU score of 0.680 for bladder.

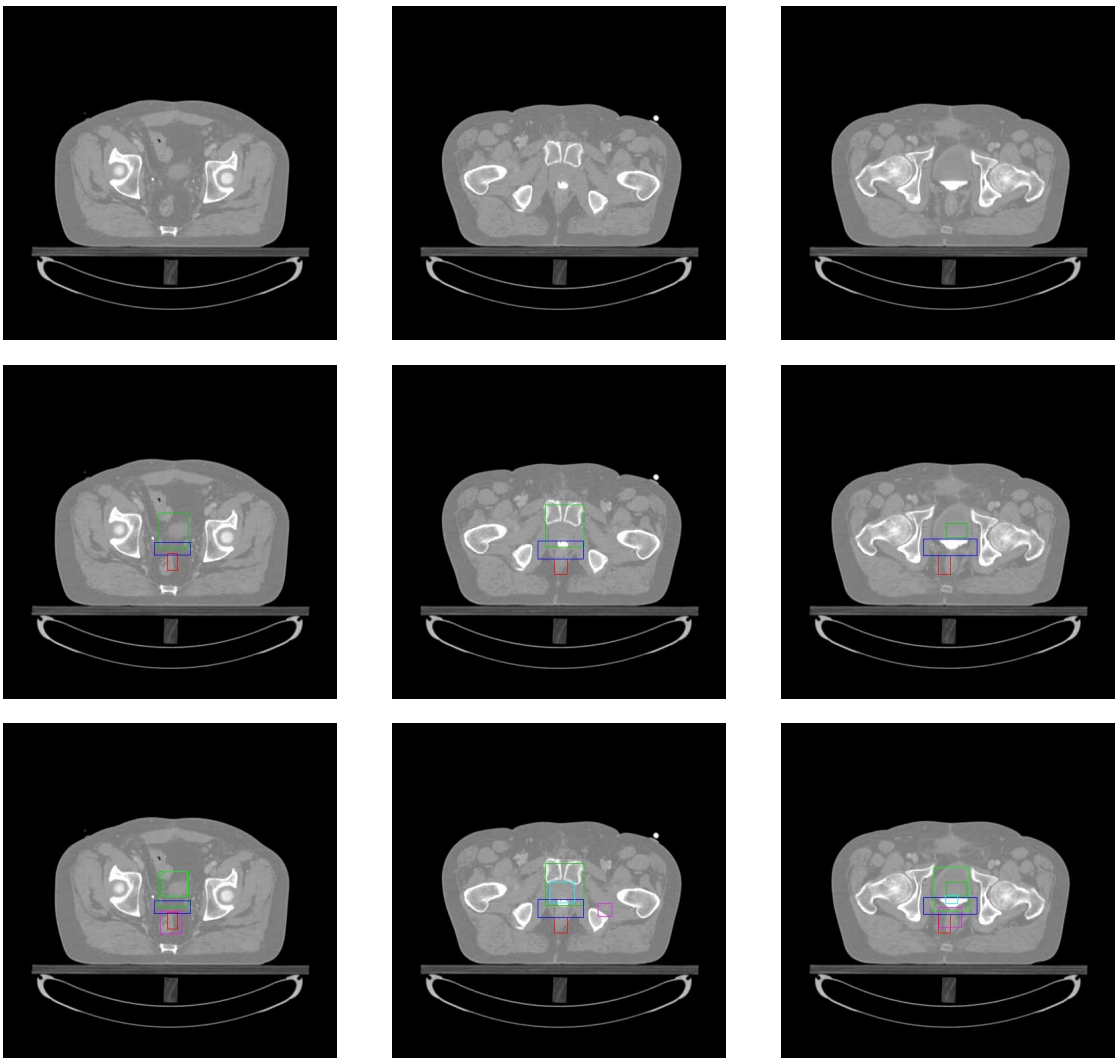


Figure 5.15: Visualization of bad predictions on data from the test set for the best performing model. Each column corresponds to a slice of the same CT scan. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The bladder is in green, prostate in blue and rectum in red. The most left slice has IoU scores of 0.180 for bladder and 0.345 for rectum . The middle slice has IoU scores of 0.0 for rectum, 0.071 for prostate. The most right slice has an IoU score of 0.190 for bladder, 0.277 for rectum and 0.089 for prostate.

	Ours	Humpire et al. [20]	Xuanang et al. [48]	Sekuboyina et al. [41]
Bladder	$\mathbf{0.822 \pm 0.060}$	0.73 ± 0.20	0.58	-
Rectum	$\mathbf{0.688 \pm 0.065}$	-	-	-
Prostate	$\mathbf{0.619 \pm 0.192}$	-	-	-
Left lung	-	$\mathbf{0.93 \pm 0.03}$	0.85	0.77
Right lung	-	$\mathbf{0.94 \pm 0.03}$	0.86	0.73
Left kidney	-	$\mathbf{0.84 \pm 0.23}$	0.75	0.60
Right kidney	-	$\mathbf{0.85 \pm 0.21}$	0.76	0.57
Liver	-	$\mathbf{0.86 \pm 0.09}$	0.78	0.80
Spleen	-	$\mathbf{0.84 \pm 0.19}$	0.70	0.60
Pancreas	-	-	**0.58**	0.32
$\mu + \sigma$	0.710 ± 0.084	$\mathbf{0.83 \pm 0.23}$	0.730 ± 0.088	0.627 ± 0.151
time (s)	1.72	3.7	**0.3**	3.1
slices	86	$42 - 1026$	512	45

Table 5.4: Results in terms of average IoU and standard deviations for different methods (ours and previous methods) and organs in 3 dimensions. The final average and standard deviation ($\mu + \sigma$) is computed on all body organs, some of which do not appear in the table (11 organs for Humpire et al. [20] and Xuanang et al. [48], 7 for Sekuboyina et al. [41]).

In order to have an easy and effective comparison criterion, we define a baseline method for 3D bounding box extraction. The method goes as follows : we gather all boxes from the training set and for each slice we compute the box that contains the three organs as depicted by figure 5.16. Then we take the average center coordinate as well as height and width[3] over all the slices in the training set. We define this box to be our baseline method. In summary, if we were to define a method very quickly to perform extraction of the organs in 3D images, we would use this average box (with constant depth) to do it.

Now that we defined a baseline method, we can compare it to ours. To do so

[3]This is equivalent to taking the average of the top left and bottom right coordinates, which is the method we used in practice but we thought it was not as easy to visualize.

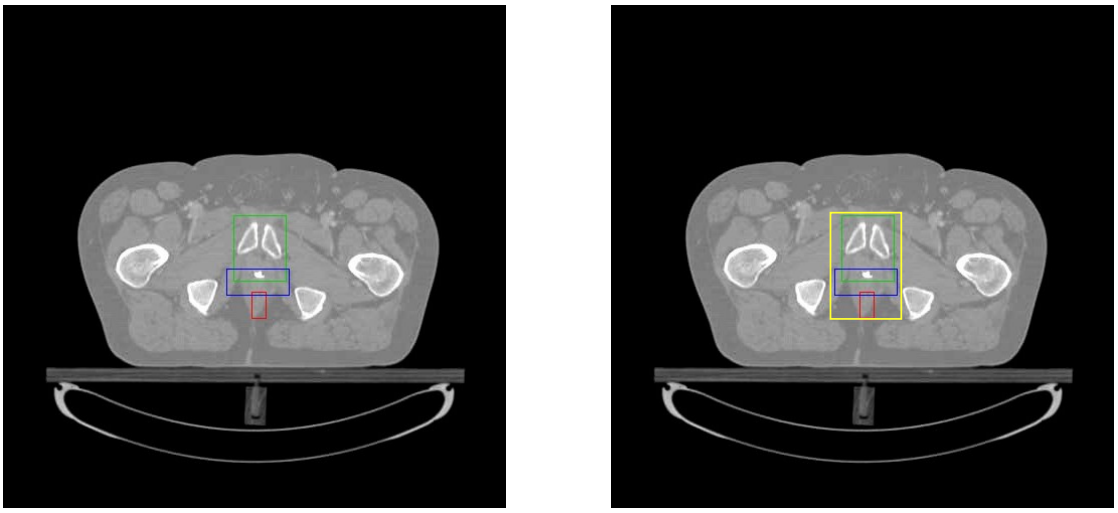


Figure 5.16: Example of the ground truth bounding boxes for each organs (left) and corresponding full box that contains all others in it (right).

we observe for every patients two variables : the IoU between the average box and the patient's ground truth box and the IoU between the box predicted using our method and the patient's ground truth box. We report the results in table 5.5. We see that our method outperforms the baseline method on every single patient of the test set by a relatively large amount.

After this, we collect (still for the 7 patients) the usual metrics plus a new one that we will denote ϕ_{volume}. This new metric represents the ratio between the final bounding box volume and the original image input volume. Mathematically,

$$\phi_{\text{volume}} = \frac{\text{H}_{\text{box}} \times \text{W}_{\text{box}}}{\text{H}_{\text{image}} \times \text{W}_{\text{image}}}$$

where H and W are respectively the height and width of the data. The number of slices stays constant and so does the depth of the original image. This is because all the slices that we work with are annotated for at least one organ, which means we never have to crop the image in depth.

Obviously, we have to carry this test a bit further. Let us recall that the goal of our work is to reduce the input data as much as we can while still making sure that all organs are part of the final output. Or equivalently, until the false negative rate is equal to zero. With this in mind, we define s a scaling factor which is used to stretch the output bounding box as illustrated in figure 5.18. We monitor the IoU, precision, recall and false negative rate and ϕ_{volume} for every integer value of s

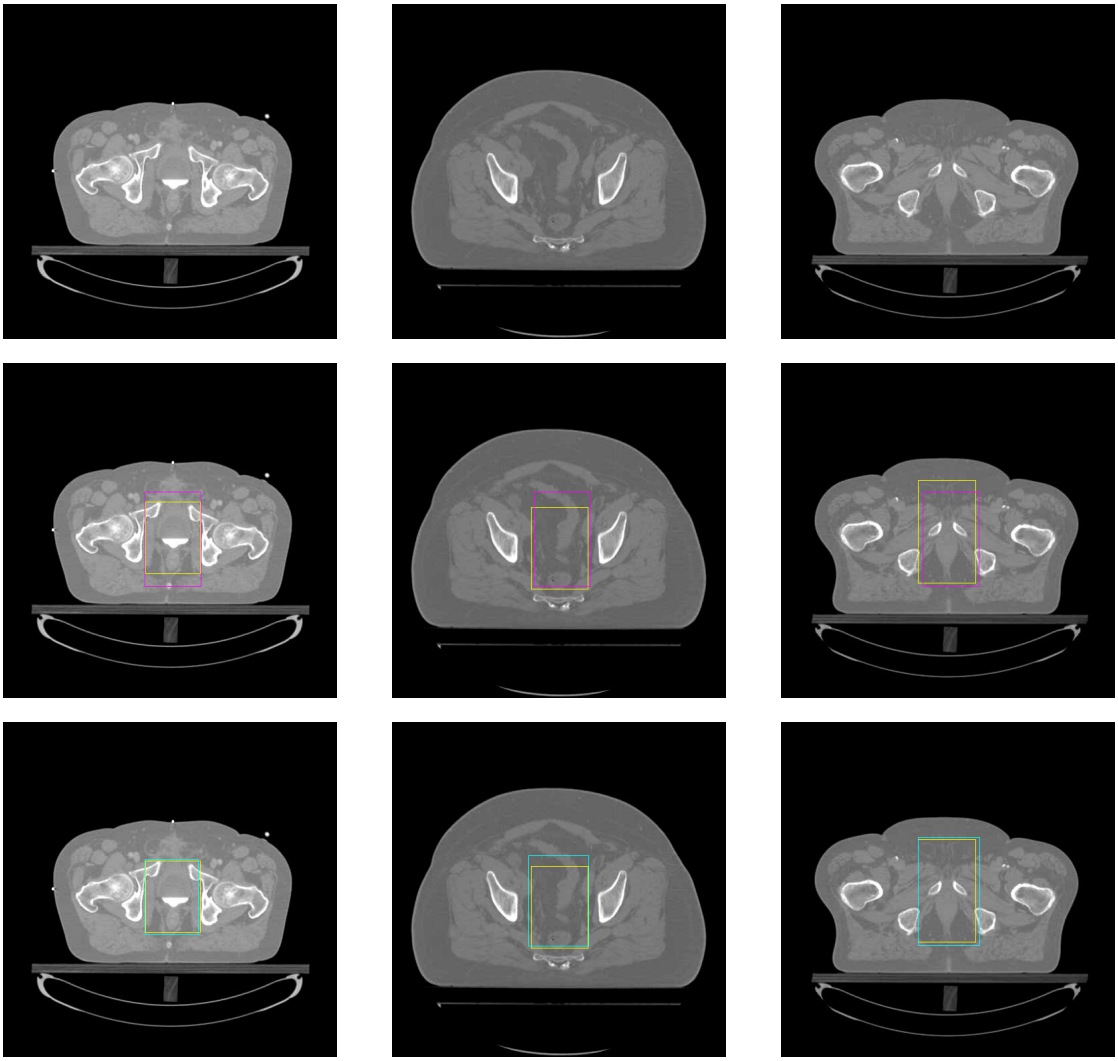


Figure 5.17: Visual comparison of the baseline method and ours (3D extraction) on data from the test set. Each column corresponds to the same slice of a patient, respectively patient 0, 3 and 5 (because they are most representative of the average). There is no need for several slices of a same patient here as the boxes are constant over the slices. The yellow box represents the ground truth bounding box (second and third row). The violet box corresponds the baseline prediction (second row). The blue box corresponds to our method's prediction (third row). From left to right, the IoU scores for the baseline method are 0.684, 0.741 and 0.638 against 0.804, 0.903 and 0.824 respectively for our method.

Patient	Baseline	Ours
Patient 0	0.684	**0.804**
Patient 1	0.755	**0.821**
Patient 2	0.851	**0.859**
Patient 3	0.741	**0.903**
Patient 4	0.754	**0.888**
Patient 5	0.638	**0.854**
Patient 6	0.522	**0.824**
$\mu \pm \sigma$	0.706 ± 0.097	$\mathbf{0.850 \pm 0.033}$

Table 5.5: Results of the IoU for the baseline method (IoU between the average box and the ground truth) and our method (IoU between the output box of our algorithm and the ground truth) evaluated on the test set. We see that our method outperforms the baseline method on every single patient of the test set by a relatively large amount.

until we reach a false negative rate of zero (for all 7 patients). Figure 5.19 shows the evolution of the metrics in terms of s. If we set $s = 0$, which corresponds to the direct output box of the method, we are able to reduce the image down to 4.2% of its original volume (which corresponds to a factor 23.8) while achieving 0.826 of IoU. We also observe that the false negative rate reaches zero for $s \geq 13$. In practice, this means that adding 13 pixels in each direction of the bounding box that is originally computed by our model ensures that all organs are included in the final boxes. Of course, this is only true on our particular data set, but by doing this we can get a good feeling of what scaling factor is needed to have a high probability of reducing the false negative rate down to zero. While ϕ_{volume} inevitably increases with s, we are still able to reach a ratio of 0.070 which corresponds to dividing the original image volume by a factor 14.3 for the maximum value of s.

About inference time, we know that a single image is processed by the network in 0.020 seconds on average. The test set 3D images have a mean number of slices of 86. If we combine these numbers, we get an average inference time of 1.72 seconds for the 3D images from the test set.

We have seen how the model performs when it comes to extracting 3D boxes. We have also studied what volume of the original image could be discarded and the impact of this dimension reduction on the performances. Now that we quantified these values, we can use them to compute how much memory would be saved by

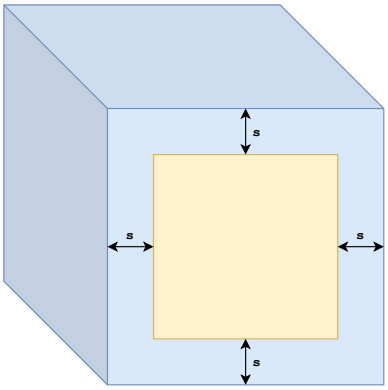


Figure 5.18: Visual representation of the scaling factor. In this example, the yellow box (which is the original output of the algorithm) is scaled by a factor s, resulting in the blue box.

using our method.

5.3.4 Extrapolation on memory usage

In the previous section, we showed that our model can be trusted to accurately extract 3-dimensional boxes from the data. We also determined a scaling factor that makes the final bounding box directly usable for segmentation with high probability. In this section, we use those results to quantify the impact of our method on GPU memory savings.

The first step is to compute the average size of our data files. Before preprocessing, the data is stored in 64 bits per element as they can take values ranging from -1024 to 3071 HU (although this range only requires 16 bits). The preprocessing step, detailed in section 3.3, reduces this range to $[0, 255]$ which allows to store the files in uint8 and save a significant amount of space. Figure 5.20a shows the box plot of that data to help better represent the distribution. When stored in uint8, the average size is 72 MB for an image and 216 MB for a mask (compared to 578 MB and 1729 MB respectively when the data is stored on 64 bits). The segmentation mask is always 3 times as heavy as its corresponding image because it holds information for the three organs. An interesting fact to notice is that the segmentation mask is made of ones and zeros, thus does not need 8 bits to be stored but rather 2. This could bring down the average data size to 145.5 MB for the segmentation masks. We will not focus too much on this as it does not

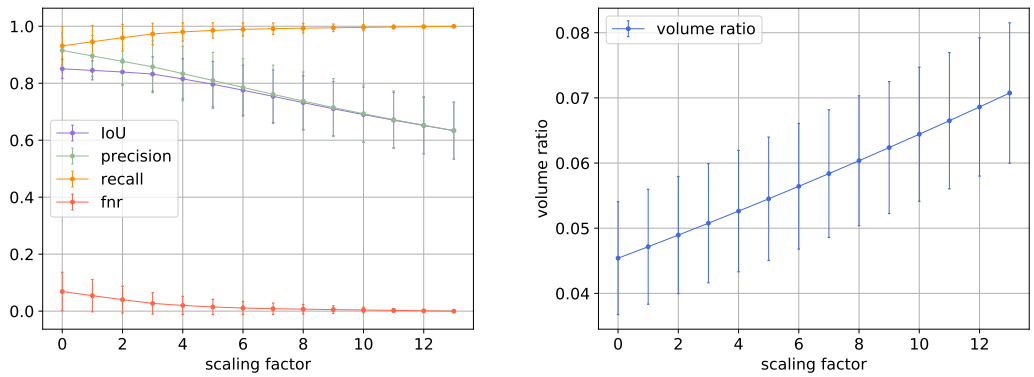


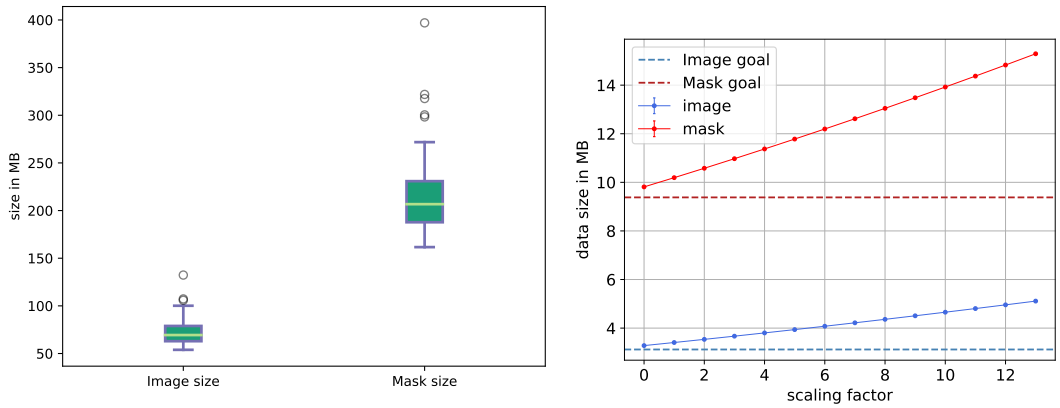
Figure 5.19: Evolution of the IoU, precision, recall, false negative rate (left) and area ratio between the original image and the final bounding box (right) in terms of scaling factor.

impact directly our work, but is rather another tool we could use in order to save even more memory when it comes to the segmentation phase.

Note that, the size of any file from our data set is directly proportional to its shape because they are stored as `.npy` files. Which means that the size does not depend on the data that is held by the array. This can seem obvious but it is relevant enough to mention it. In practice, it is useful because we do not need to know where the image is reduced, only how much we have reduced it. This means we can multiply the coefficient ϕ_{volume} from section 5.3.3 to the original data size to know the size of the output file of the algorithm, or equivalently, the input size to the segmentation algorithm. Figure 5.20b shows the evolution of that size in terms of scaling factor for uint8 enconding.

With these final results, we see that our model is able to reduce the image size down to a small amount above what is used by Léger et al. [28] if we do not scale the bounding box. This of course has limitations because that box potentially misses some parts of the organs. In general, we see that our algorithm is able to divide the original input data size by factors ranging from 13.5 up to 23.8 depending on s. This suggests that there is a trade off between dimension reductions and false negative rate. Another important aspect is that by reducing the input data size, we get rid of the weights that were associated to the remaining pixels. This, naturally has a very significant impact on memory usage.

Let's take a step back from those results and recall some information from section 1.5. The GPU we are using as our example has 11 GB of VRAM, the input



(a) Input data sizes (uint8).

(b) Average output data size.

Figure 5.20: Box plot of the 3D initial data sizes in uint8 encoding (left) and evolution of the average image and mask size after running our algorithm and in terms of scaling ratio (right). The blue and red dashed line represent respectively the current image and mask input sizes used by Léger et al. [28].

data size (image and mask combined) revolves around 10 MB and yet a batch size of 4 can cause the system to go out of memory. This gap between the input data size and the memory usage can seem irrational, but the reason behind this is actually not. The first, and maybe obvious, aspect to think about is that the model does not use the same amount of memory at all time, but is limited to the maximum amount of memory it will use at any point in the training. But behind this, the real reason is that each layer with learnable parameters will need to store its input until the backward pass. This means that every batch norm, convolution, dense layer will store its input until it was able to compute the gradient of its parameters [11]. Knowing that even a simple NN can carry several million of weights, we understand where this need for memory might come from. In addition to that and as we already said before, the larger the input, the more weights you will need to associate with each new element (pixel in our case). So even a slight change in input dimension can have a tremendous impact on memory.

We have discussed the overall performances of the model along with the impact on memory usage. To set a reference, we compare those performances to other papers that have some similarities and differences with our method. Section 5.3.5 addresses this comparison.

5.3.5 Comparison with other authors

We compare our results to three different papers which we introduced in sections 2.3.1, 2.3.2 and 2.3.3. Those papers are respectively from Humpire et al. [20], Xuanang et al. [48] and Sekuboyina et al. [41]. We report the IoU of our method along with theirs for a series of selected organs in table 5.4.

Before going on about the results, we see that none of those papers deal with the rectum or the prostate. Which encourages to think that our method brings something new to the table. When it comes to the results themselves, several observations arise. The first one is that our method is responsible for detecting 3 classes of object while the others account for 11, 23 (11 body organs and 12 head organs) and 7 classes. This can be a first indicator of why our system outperforms the others on bladder detection.

Another aspect, which is probably the most critical, is that all these models perform actual cropping in 3 dimensions. Our model only really performs two dimensional cropping as the depth of the images is always constant (refer to section 5.3.3). This means that there is a complete dimension in which we achieve a perfect accuracy in any case but not for them.

On the other hand, these methods (apart from Sekuboyina et al. [41]) benefit from the large data set they work with. Respectively, 1884 and 201 CT scans for Humpire et al. [20] and Xuanang et al. [48] while we only used 76. Sekuboyina et al. [41] has a set composed of 70 CT scan but the whole point of using reinforcement learning was to show that less data is needed to achieve good results. This difference in data set sizes is likely to play a key role in the results.

Looking at the inference time for each model, we see that our method comes second. But we have to be careful because, for our method, this number depends on the number of slices in an image. This is not true for the other methods as they work directly with 3D images. With this in mind, we specify the number of slices used by each model in table 5.4 to highlight the true differences.

In general we see that our method performs a little bit under the other supervised learning methods ([20] and [48]) in terms of IoU. A limitation of our comparison is specifically that it only compares the results in terms of IoU. An improvement could be to compare them in terms of wall distances and centroid distances, which was a metric that all three papers used. Only Xuanang et al. [48] computed the average precision and recall for their method, which we reported in table 5.6. We see that they reach better precision and recall even though we come

	Precision	Recall	False negative rate
Ours	0.915 ± 0.061	0.931 ± 0.067	0.068 ± 0.067
Xuanang et al. [48]	**0.979**	**0.987**	-

Table 5.6: Average precision, recall and false negative rate (with standard deviations for our method). Xuanang et al. [48] were the only ones to detail these results in their paper which is why we could only compare those metrics with them.

quite close.

5.3.6 Conclusion on deep learning model's performances

From the previous sections, we can conclude the our model overcomes the lack of accuracy that the classical and computer vision methods suffered from. We reach an average IoU of 0.85 (for a single box containing all organs) in 3 dimensions while allowing to reduce the data to 4.2% of its original size. Even though the performances in 2 dimensions could still be improved, the model adapts well in 3 dimensions. However, it could benefit from more data to be trained on, as this is likely to be one of its biggest limitations.

5.4 Limitations and future work

In this work, we started out by evaluating the traditional methods on CBCT scans and switched to CT scans when it came to deep learning. This is because we only had the CBCT data at the beginning and when we started designing the deep learning model, the CT scans became available. The CT scans are more realistic (essentially because they are larger in all three dimensions) than the CBCT we were given. We could improve the statistics and make better comparison of the methods if the experiments were conducted on both data sets.

Carrying out the experiments on a larger set of patients could certainly improve the performances and generalizability of the model. Different techniques could also be used to improve efficiency such as data augmentation. We evaluated the model on a set of metrics which could be extended with others like the wall distance or the Dice Similarity Coefficient.

As mentioned earlier, we only use annotated data to train our model in 2D.

Extending the model in 3D, our algorithm only performs cropping in height and width and keeps the depth constant. This is an aspect that could be worked on, training the model with slices that have no organs in it and perform 3D extraction by also cropping in depth for those slices.

We evaluate all performances based on comparison with ground truth data. As this algorithm aims at suppressing the need for human intervention in the process of organs localization, we could make use of a tier, human observer performing bounding box detection by hand. Comparing his or her results with our method's would set a proper baseline for measuring efficiency.

Finally, we could make a pipeline between our model and a segmentation model (for instance Léger et al. [28]) to make it directly applicable. From there, another set of experiments could be conducted and we could evaluate the practical impact of our method in the clinical workflow.

5.5 Summary

In section 5.1, we introduce four metrics that we think are relevant for evaluating our model : IoU, precision, recall and false negative rate.

In section 5.2, we assess the quality of the classical and computer vision methods. We run several experiments on the split and crop by making the threshold k vary and monitor the precision, recall and false negative rate (section 5.2.1). We repeat this experiment but pre-process the image with split and merge (section 5.2.2). We observe that both methods lack of accuracy and show no trend of improvement in terms of threshold (section 5.2.3).

In section 5.3, we evaluate the performances of our deep learning model. Our training set is made of 4500 images and our validation and test sets are made of 450. We set the learning rate of our model to 10^{-3} and the batch size to 16 (section 5.3.1). After experimenting on custom anchor boxes, we observe that a model with custom boxes achieves better results than without. We try out different combination of λ_{coord} and λ_{obj}, train each configuration for around 4 hours (100 epochs) and pick the best model in terms of IoU on the validation set. We perform detection on the test set and obtain an average IoU score of 0.681 ± 0.233, 0.496 ± 0.198 and 0.538 ± 0.245 for bladder, rectum and prostate respectively with and average inference time of 0.02 second per image (section 5.3.2). We extend the model to perform 3D extraction and compare the results with a baseline method

that we define. Our method outperforms the baseline method for each patient by a significant margin, achieving 0.850 ± 0.033 of IoU (section 5.3.3). We use the previously computed results to analyse the impact that our method could have in terms of memory savings. We see that we are able to get rid of 95.6% of the original data volume if we accept a false negative rate of 0.085 ± 0.099 on the test set. If we bring this rate down to zero, we are still able to reduce the data to 7% of its original size (section 5.3.4). Finally we compare our results with the ones that other authors obtain from their methods, this time detecting organs separately. On average, we obtain an IoU of 0.710 ± 0.084 compared to 0.84 ± 0.23 for current SOTA and 0.73 ± 0.088, 0.627 ± 0.151 for the other methods. Our processing speed for a full 3D image is 1.72 seconds on average but depends on the number of slices.

We conclude by expliciting the limitations of our work as well as the possible improvements that could be brought to it (section 5.4).

Conclusion

Automatic detection and localization of organs is an important step that can improve organ segmentation. In this work, we built an algorithm based on an existing model for object detection. We used this algorithm to perform detection on bladder, rectum and prostate in the specific framework of radiotherapy applications. The application we focus on is to reduce the size of a CT scan and make sure all organs are still in the image, which can then be input to a segmentation algorithm. But in general, the method can be used for any purpose that requires detection on bladder, rectum and prostate, separately or together in a CT image.

We started by trying out methods based on traditional computer vision and showed that they were not accurate enough to fit our problem. This can be due to the poor contrast given in the medical images combined with the lack of ability to extract deep features with traditional techniques.

To overcome the limitations of these techniques, we investigated a deep learning model, YOLO, and extended it to perform 3D detection and localization of the organs. The ability of deep learning to extract embedded features makes it possible to carry out this task. We tried several set of parameters for the model and recommended the best ones. We defined a baseline method to compare with ours and found the correspondence with previous work : Humpire et al. [20], Xuanang et al. [48] and Sekuboyina et al. [41]. We also established and quantified the trade off between accuracy and memory savings.

For future work, we want to extend the model to perform cropping in all 3 dimensions. We would also like to train the model on a broader data set, add techniques like data augmentation to improve performances, evaluate it with more metrics such as DSC or wall distance and finally, close the pipeline architecture by making a bridge between our method and a segmentation algorithm for bladder, prostate and rectum.

Appendices

Appendix A

Bounding box predictions on the test set

This appendix contains three slices and associated predictions for each patient of the test set (7 patients). Theses slices are chosen randomly but spread in the 3D volume (one is a the front, one in the center and one at the back of the overall volume). The IoU score for each prediction is given under the figure.

Patient 0

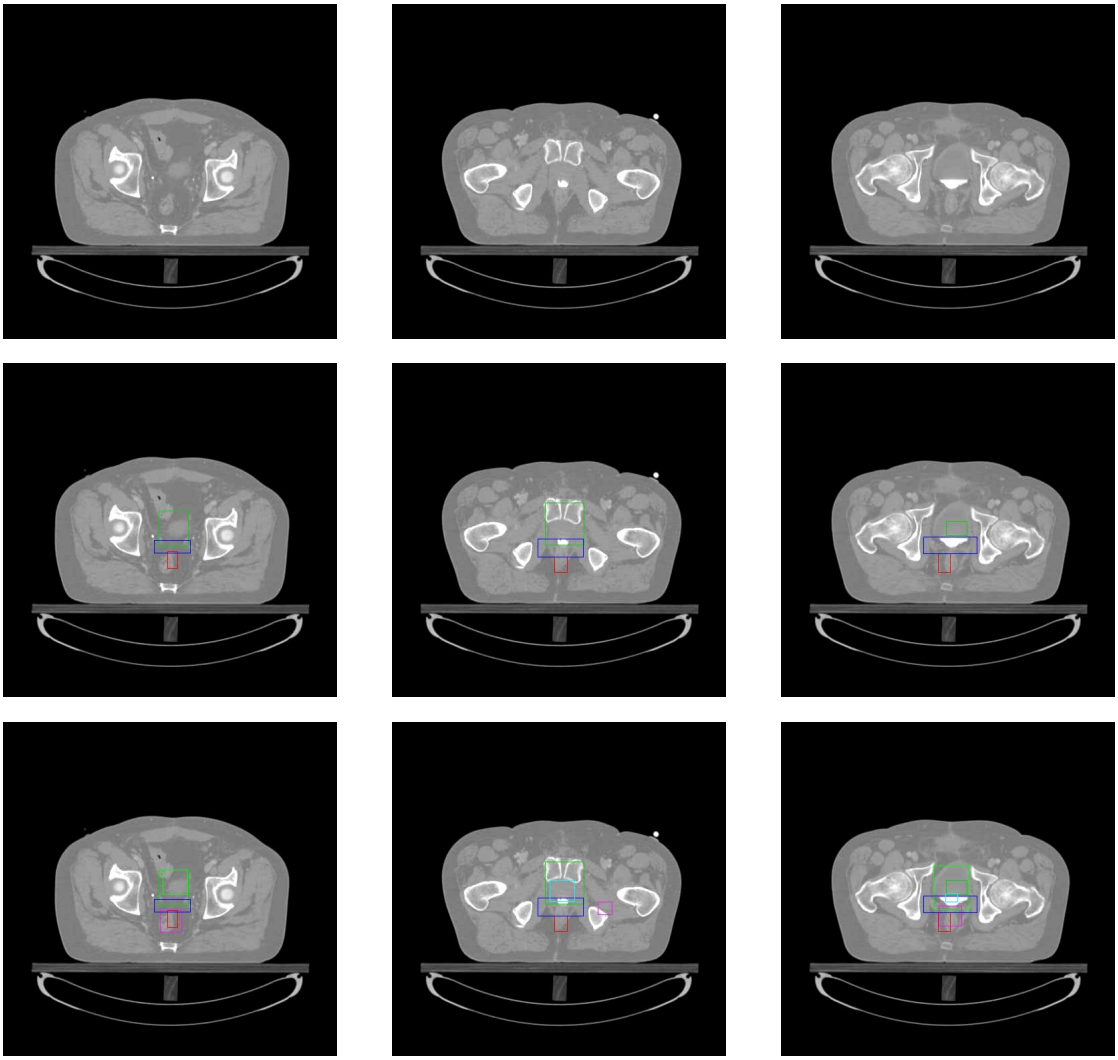


Figure A.1: The IoU score is given from left to right. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The predicted bladder, in green, has an IoU of 0.180 (0), 0.0 (1) and 0.190 (2). The predicted rectum, in red, has an IoU of 0.345 (0), 0.0 (1) and 0.277 (2). The predicted prostate, in blue, has an IoU of 0.0 (0), 0.190 (1) and 0.089 (2).

Patient 1

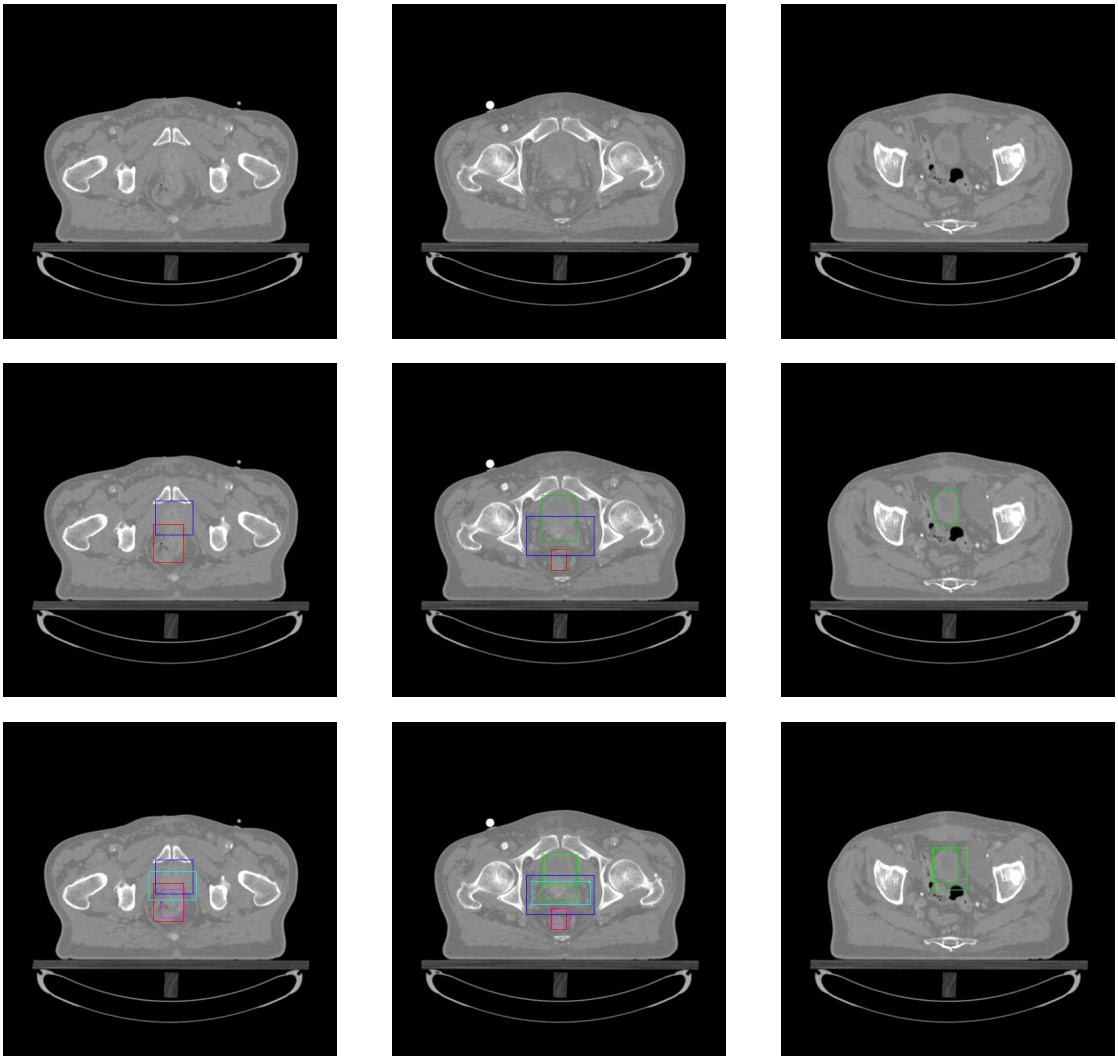


Figure A.2: The IoU score is given from left to right. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The predicted bladder, in green, has an IoU of 0.687 (1) and 0.680 (2). The predicted rectum, in red, has an IoU of 0.553 (0) and 0.546 (1). The predicted prostate, in blue, has an IoU of 0.524 (0) and 0.518 (1).

Patient 2

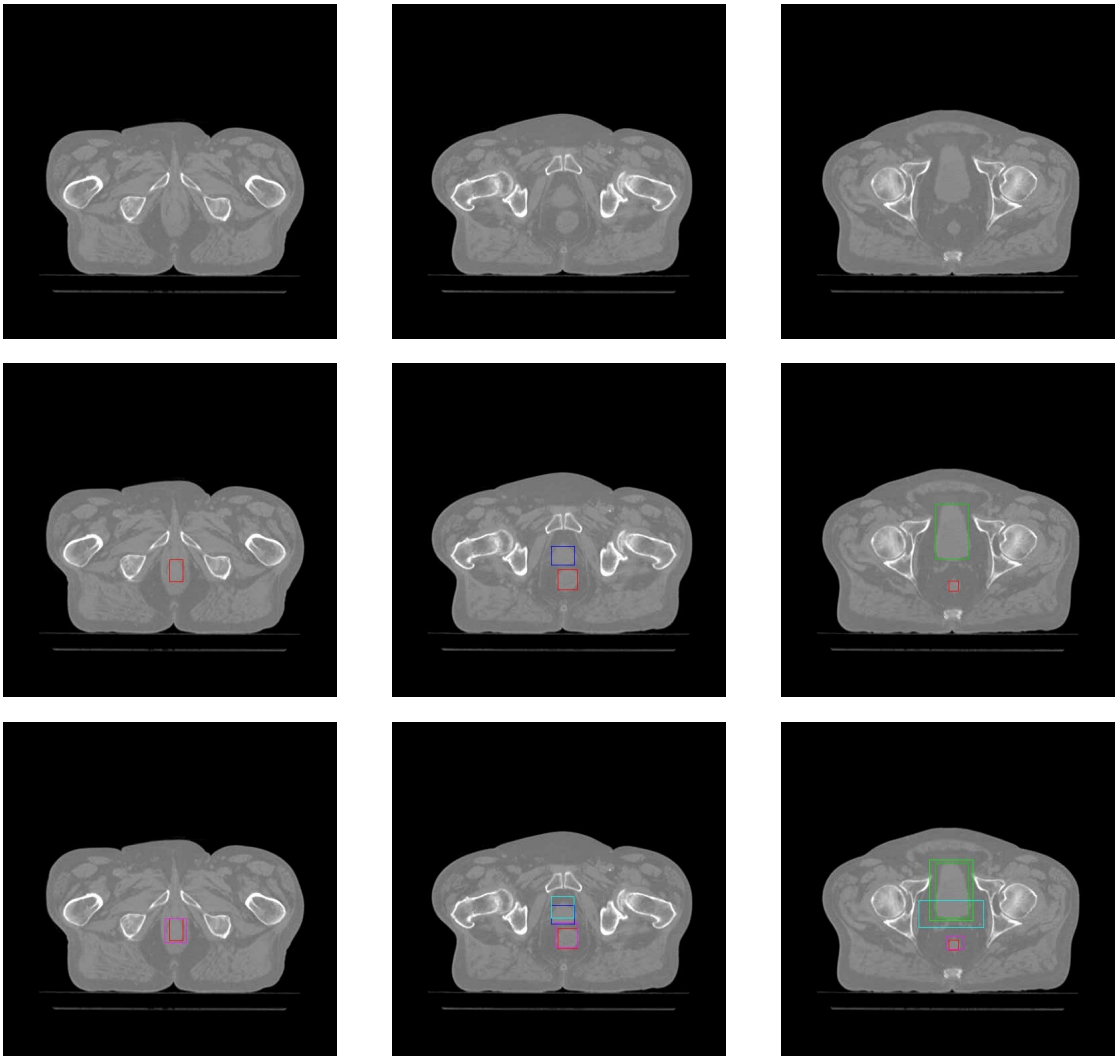


Figure A.3: The IoU score is given from left to right. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The predicted bladder, in green, has an IoU of 0.729 (2). The predicted rectum, in red, has an IoU of 0.191 (0), 0.638 (1) and 0.399 (2). The predicted prostate, in blue, has an IoU of 0.460 (1) and 0.0 (2).

Patient 3

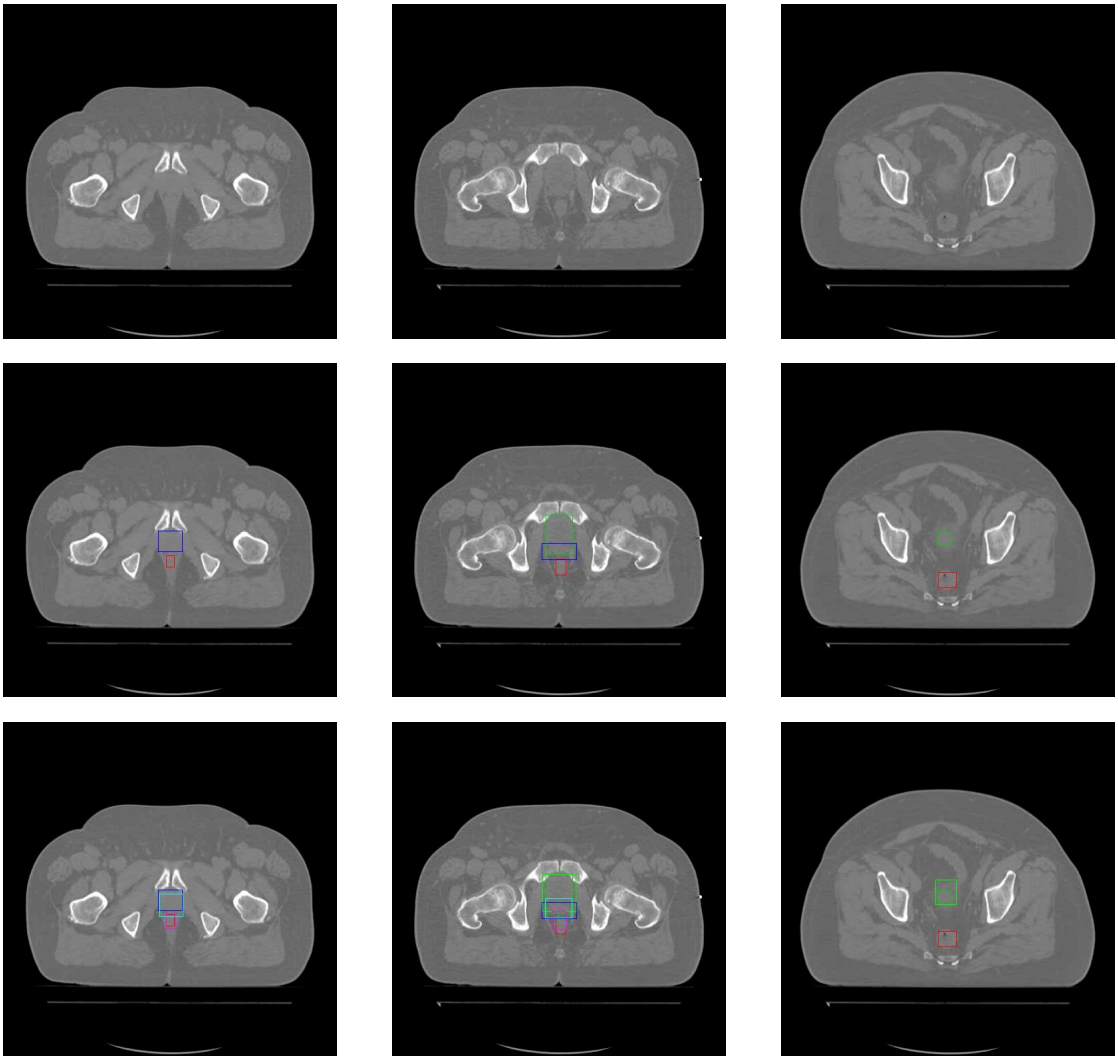


Figure A.4: The IoU score is given from left to right. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The predicted bladder, in green, has an IoU of 0.680 (1) and 0.351 (2). The predicted rectum, in red, has an IoU of 0.619 (0), 0.366 (1) and 0.0 (2). The predicted prostate, in blue, has an IoU of 0.853 (0) and 0.655 (1).

Patient 4

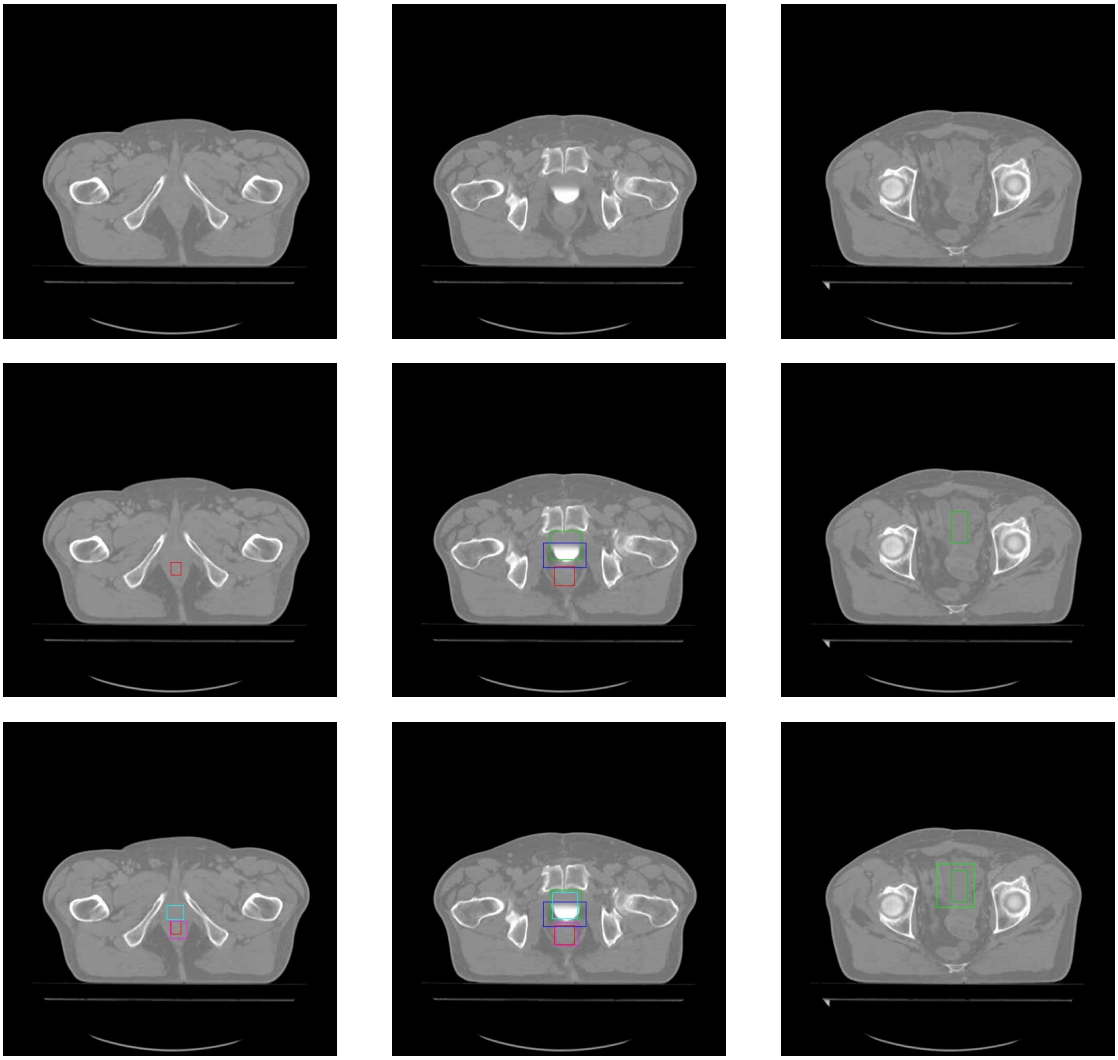


Figure A.5: The IoU score is given from left to right. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The predicted bladder, in green, has an IoU of 0.0 (1) and 0.311 (2). The predicted rectum, in red, has an IoU of 0.429 (0) and 0.692 (1). The predicted prostate, in blue, has an IoU of 0.0 (0) and 0.326 (1).

Patient 5

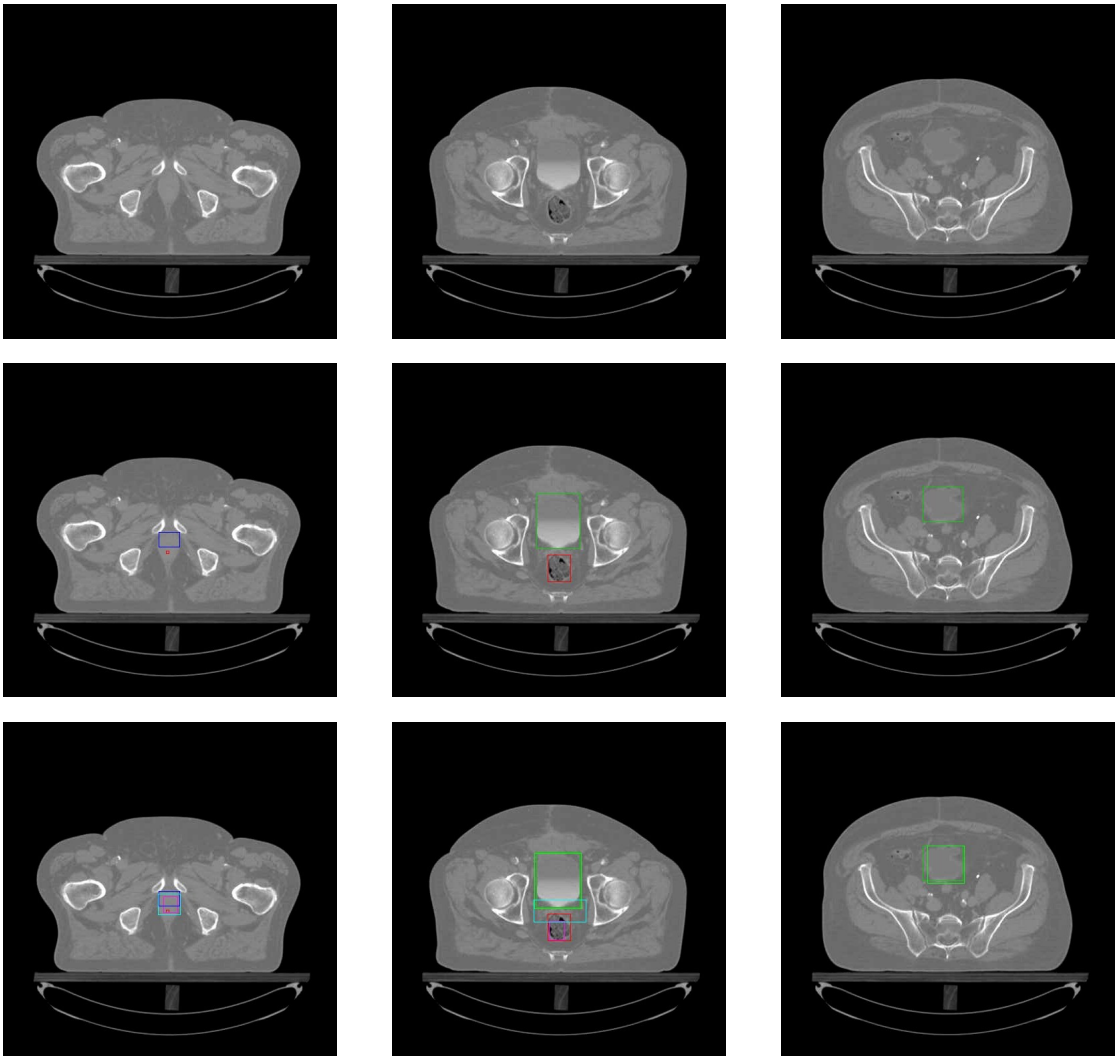


Figure A.6: The IoU score is given from left to right. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The predicted bladder, in green, has an IoU of 0.845 (1) and 0.563 (2). The predicted rectum, in red, has an IoU of 0.030 (0) and 0.621 (1). The predicted prostate, in blue, has an IoU of 0.549 (0) and 0.0 (1).

Patient 6

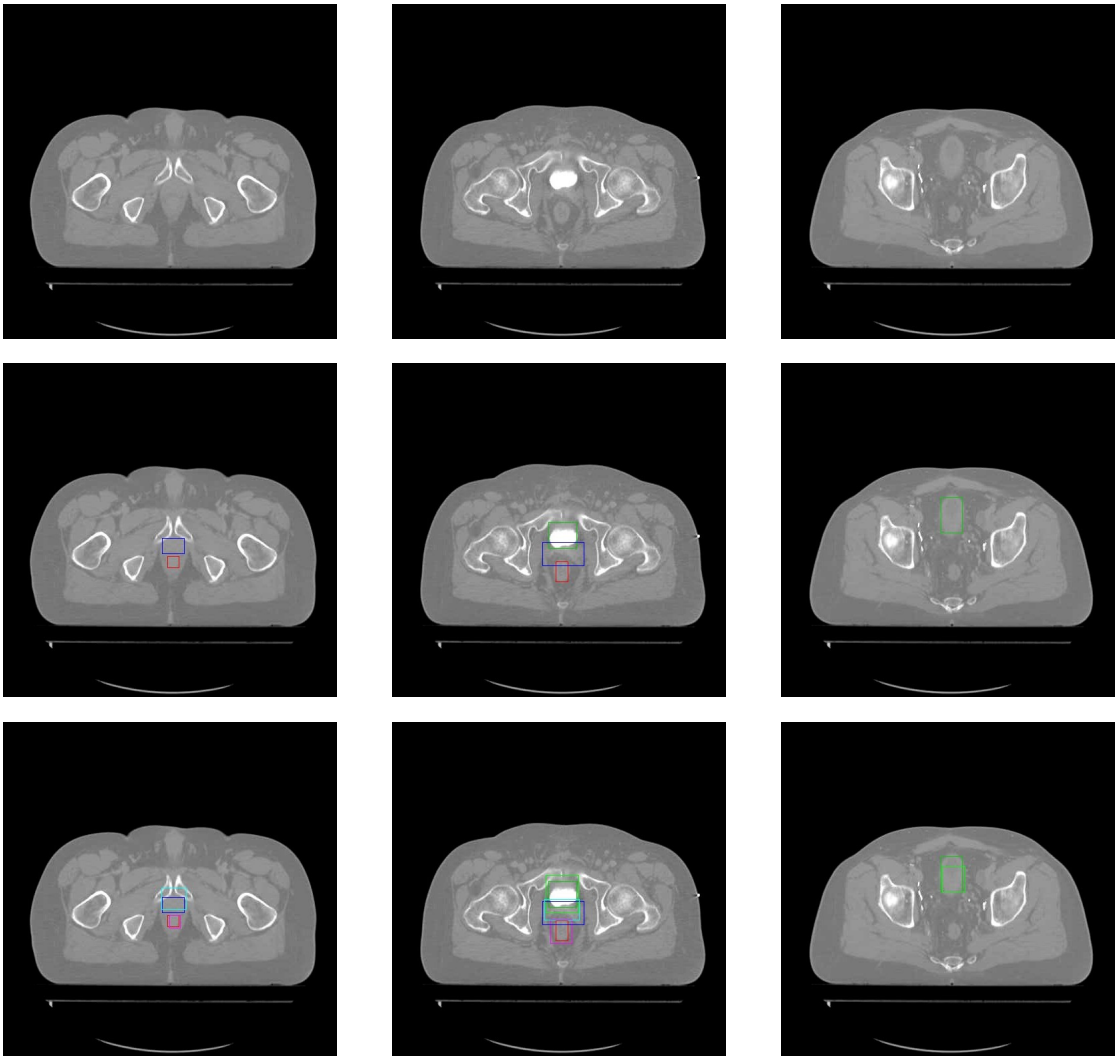


Figure A.7: The IoU score is given from left to right. Dark colours represent ground truth bounding boxes (second and third row) while light colours are the bounding boxes that our system predicted (third row). The predicted bladder, in green, has an IoU of 0.856 (1) and 0.623 (2). The predicted rectum, in red, has an IoU of 0.649 (0) and 0.556 (1). The predicted prostate, in blue, has an IoU of 0.648 (0) and 0.748 (1).

Appendix B

Neural network structure

B.1 Model summary

Table B.1 represents the NN structure as it is returned by keras' function `model.summary()`.

Layer name	**Output shape**	**Connected to**
input_1	(416, 416, 3)	
conv2d_1	(416, 416, 32)	input_1
batch_normalization_1	(416, 416, 32)	conv2d_1
leaky_re_lu_1	(416, 416, 32)	batch_normalization_1
max_pooling2d_1	(208, 208, 32)	leaky_re_lu_1
conv2d_2	(208, 208, 64)	max_pooling2d_1
batch_normalization_2	(208, 208, 64)	conv2d_2
leaky_re_lu_2	(208, 208, 64)	batch_normalization_2
max_pooling2d_2	(104, 104, 64)	leaky_re_lu_2
conv2d_3	(104, 104, 128)	max_pooling2d_2
batch_normalization_3	(104, 104, 128)	conv2d_3
leaky_re_lu_3	(104, 104, 128)	batch_normalization_3
conv2d_4	(104, 104, 64)	leaky_re_lu_3
batch_normalization_4	(104, 104, 64)	conv2d_4
leaky_re_lu_4	(104, 104, 64)	batch_normalization_4
conv2d_5	(104, 104, 128)	leaky_re_lu_4
batch_normalization_5	(104, 104, 128)	conv2d_5
leaky_re_lu_5	(104, 104, 128)	batch_normalization_5
max_pooling2d_3	(52, 52, 128)	leaky_re_lu_5
conv2d_6	(52, 52, 256)	max_pooling2d_3
batch_normalization_6	(52, 52, 256)	conv2d_6

leaky_re_lu_6	(52, 52, 256)	batch_normalization_6
conv2d_7	(52, 52, 128)	leaky_re_lu_6
batch_normalization_7	(52, 52, 128)	conv2d_7
leaky_re_lu_7	(52, 52, 128)	batch_normalization_7
conv2d_8	(52, 52, 256)	leaky_re_lu_7
batch_normalization_8	(52, 52, 256)	conv2d_8
leaky_re_lu_8	(52, 52, 256)	batch_normalization_8
max_pooling2d_4	(26, 26, 256)	leaky_re_lu_8
conv2d_9	(26, 26, 512)	max_pooling2d_4
batch_normalization_9	(26, 26, 512)	conv2d_9
leaky_re_lu_9	(26, 26, 512)	batch_normalization_9
conv2d_10	(26, 26, 256)	leaky_re_lu_9
batch_normalization_10	(26, 26, 256)	conv2d_10
leaky_re_lu_10	(26, 26, 256)	batch_normalization_10
conv2d_11	(26, 26, 512)	leaky_re_lu_10
batch_normalization_11	(26, 26, 512)	conv2d_11
leaky_re_lu_11	(26, 26, 512)	batch_normalization_11
conv2d_12	(26, 26, 256)	leaky_re_lu_11
batch_normalization_12	(26, 26, 256)	conv2d_12
leaky_re_lu_12	(26, 26, 256)	batch_normalization_12
conv2d_13	(26, 26, 512)	leaky_re_lu_12
batch_normalization_13	(26, 26, 512)	conv2d_13
leaky_re_lu_13	(26, 26, 512)	batch_normalization_13
max_pooling2d_5	(13, 13, 512)	leaky_re_lu_13
conv2d_14	(13, 13, 1024)	max_pooling2d_5
batch_normalization_14	(13, 13, 1024)	conv2d_14
leaky_re_lu_14	(13, 13, 1024)	batch_normalization_14
conv2d_15	(13, 13, 512)	leaky_re_lu_14
batch_normalization_15	(13, 13, 512)	conv2d_15
leaky_re_lu_15	(13, 13, 512)	batch_normalization_15
conv2d_16	(13, 13, 1024)	leaky_re_lu_15
batch_normalization_16	(13, 13, 1024)	conv2d_16
leaky_re_lu_16	(13, 13, 1024)	batch_normalization_16
conv2d_17	(13, 13, 512)	leaky_re_lu_16
batch_normalization_17	(13, 13, 512)	conv2d_17
leaky_re_lu_17	(13, 13, 512)	batch_normalization_17
conv2d_18	(13, 13, 1024)	leaky_re_lu_17
batch_normalization_18	(13, 13, 1024)	conv2d_18

leaky_re_lu_18	(13, 13, 1024)	batch_normalization_18
conv2d_19	(13, 13, 1024)	leaky_re_lu_18
batch_normalization_19	(13, 13, 1024)	conv2d_19
conv2d_21	(26, 26, 64)	leaky_re_lu_13
leaky_re_lu_19	(13, 13, 1024)	batch_normalization_19
batch_normalization_21	(26, 26, 64)	conv2d_21
conv2d_20	(13, 13, 1024)	leaky_re_lu_19
leaky_re_lu_21	(26, 26, 64)	batch_normalization_21
batch_normalization_20	(13, 13, 1024)	conv2d_20
space_to_depth	(13, 13, 256)	leaky_re_lu_21
leaky_re_lu_20	(13, 13, 1024)	batch_normalization_20
concatenate_1	(13, 13, 1280)	space_to_depth leaky_re_lu_20
conv2d_22	(13, 13, 1024)	concatenate_1
batch_normalization_22	(13, 13, 1024)	conv2d_22
leaky_re_lu_22	(13, 13, 1024)	batch_normalization_22
conv2d_24	(13, 13, 40)	leaky_re_lu_22
input_2	(,5)	
input_3	(13,13,5,1)	
input_4	(13,13,5,1)	
yolo_loss	(1)	conv2d_24 input_2 input_3 input_4

Table B.1: NN structure as defined by `keras.model.summary()`

B.2 Convolutional neural network architecture

Table B.2 represents the CNN structure in terms of convolution and max pooling layers with associated number of filters, size, strides and output size.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	$(416, 416)$
MaxPool		$2 \times 2/2$	$(208, 208)$
Convolutional	64	3×3	$(208, 208)$
MaxPool		$2 \times 2/2$	$(104, 104)$
Convolutional	128	3×3	$(104, 104)$
Convolutional	64	1×1	$(104, 104)$
Convolutional	128	3×3	$(104, 104)$
MaxPool		$2 \times 2/2$	$(52, 52)$
Convolutional	256	3×3	$(52, 52)$
Convolutional	128	1×1	$(52, 52)$
Convolutional	256	3×3	$(52, 52)$
MaxPool		$2 \times 2/2$	$(26, 26)$
Convolutional	512	3×3	$(26, 26)$
Convolutional	256	1×1	$(26, 26)$
Convolutional	512	3×3	$(26, 26)$
Convolutional	256	1×1	$(26, 26)$
Convolutional	512	3×3	$(26, 26)$
MaxPool		$2 \times 2/2$	$(13, 13)$
Convolutional	1024	3×3	$(13, 13)$
Convolutional	256	1×1	$(13, 13)$
Convolutional	1024	3×3	$(13, 13)$
Convolutional	256	1×1	$(13, 13)$
Convolutional	1024	3×3	$(13, 13)$
Convolutional	1024	3×3	$(13, 13)$
Convolutional	1024	3×3	$(13, 13)$
Convolutional	64	1×1	$(26, 26)$
Space to depth		2×2	$(13, 13)$
Concatenation			$(13, 13)$
Convolutional	1024	3×3	$(13, 13)$
Convolutional	$B \times (C + 5) = 40$	3×3	$(13, 13)$
Softmax			

Table B.2: NN structure in terms of convolution and maximum pooling layers. Each convolution layer is followed by a batch normalization layer and a leaky ReLU, except for the very last one. The convolution layer that comes right after the single line is the *passthrough layer*. The concatenation layer takes the two previous layers' output and concatenates them.

Appendix C

Implementation

C.1 Modifications made to the original implementation

The base code that we used was retrieved from this repository : `https://github.com/allanzelener/YAD2K`. Obviously, we had to bring considerable modifications to it in order to solve our problem. In the end, ten python files were added to the original project along with folders and files related to the data sets or for storing results. The final repository, which contains the code for both traditional and deep learning methods, is available at `https://github.com/lucaderumier/pelvis_detection`.

The original neural network architecture was left almost untouched. We only added four global variables corresponding to the λ coefficients of the model. This allows us to experiment on different λ values more easily. The file `retrain_yolo.py` that was originally designed to retrain the model on custom data was renamed to `training.py` and modified to make more features available. Among those features we find, data normalization, possibility of training with data generators, saving model's weights and running prediction every 10 epochs[1] and some other minor features. In addition to those modifications the following files were created :

1. `anchors_clustering.py` : file that contains a k-means clustering algorithm and results visualisation function to be used on our data set.

2. `config.py` : stores the `Config` class that encapsulates all the useful parameters of the model such as learning rate, batch size etc.

[1] We set it to 10 but this can be changed in the configurations file

3. `data_generator.py` : stores the `DataGenerator` class, directly usable by keras, that enables to generate custom batches.
4. `dataset_utils.py` : utility file that we use to transform our `.jpg` images and `.p` annotation file into a `.npz` file that the model can use.
5. `evaluation.py` : script that is used to evaluate the model's performances on our data.
6. `extract.py` : file that holds all the functions related to the 3-dimensional boxes extraction from the 2-dimensional model's results.
7. `graphs.py` : file for plotting graphs.
8. `prediction.py` : file that computes the predictions on the data using a previously saved model.
9. `utils.py` : file that holds all the utility functions.
10. `visualize.py` : file to visualize the results on images from our data set.
11. `Makefile` : makefile that makes the project easier to use.

Bibliography

[1] Enlarged prostate. https://www.philahomeopathy.com/enlarged-prostate/. Accessed: March 30, 2020.

[2] Prostate cancer : Statistics. Accessed: April 27, 2020.

[3] What are the benefits of ct scans? https://www.radiologyinfo.org/en/info.cfm?pg=safety-hiw_04. Accessed: March 25, 2020.

[4] *Accuracy Requirements and Uncertainties in Radiotherapy*. Number 31 in Human Health Series. International Atomic Energy Agency, Vienna, 2016.

[5] Josep Borras, Yolande Lievens, Michael Barton, Julieta Corral, Jacques Ferlay, Freddie Bray, and Cai Grau. How many new cancer patients in europe will require radiotherapy by 2025? an estro-hero analysis. *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology*, 119, 02 2016.

[6] Neil Burnet, Simon Thomas, Kate Burton, and Sarah Jefferies. Defining the tumour and target volumes for radiotherapy. *Cancer imaging : the official publication of the International Cancer Imaging Society*, 4:153–61, 02 2004.

[7] Juan C. Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[8] Wikipedia contributors. Ct scan. https://en.wikipedia.org/wiki/CT_scan. Accessed March 25, 2020.

[9] Wikipedia contributors. Radiodensity. https://en.wikipedia.org/wiki/Radiodensity. Accessed March 26, 2020.

[10] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature Medicine*, 25, 01 2019.

[11] Quentin Febvre. Deep learning memory usage and pytorch optimization tricks. https://www.sicara.ai/blog/2019-28-10-deep-learning-memory-usage-and-pytorch-optimization-tricks. Accessed: June 13, 2020.

[12] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[13] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *CoRR*, abs/1811.12560, 2018.

[14] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[15] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[16] John Lee Guillaume Janssens and Edmond Sterpin. Engineering challenges in protontherapy, 2017.

[17] Raz Haleva. How to break gpu memory boundaries even with large batch sizes. https://towardsdatascience.com/how-to-break-gpu-memory-boundaries-even-with-large-batch-sizes-7a9c27a400ce. Accessed May 31, 2020.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[19] Steven L. Horowitz and Theodosios Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the Association for the Computing Machinery*, 1976.

[20] Gabriel Humpire Mamani, Arnaud Setio, Bram Ginneken, and Colin Jacobs. Efficient organ localization using multi-label convolutional neural networks in thorax-abdomen ct scans. *Physics in Medicine and Biology*, 63, 03 2018.

[21] National Cancer Institute. About cancer. https://www.cancer.gov/about-cancer. Accessed: March 28, 2020.

[22] National Cancer Institute. Prostate cancer. January 1980.

[23] Jacqueline Johnson, Russell Leonard, A. Lubinsky, and Stefan Schweizer. Opportunities for fluorochlorozirconate and other glass-ceramic detectors in medical imaging devices. *Journal of Biomedical Technology and Research*, 02, 09 2015.

[24] Mohan Kumar, Muhammad Shanavas, Ashwin Sidappa, and Madhu Kiran. Cone beam computed tomography - know its secrets. *Journal of international oral health : JIOH*, 7:64–8, 02 2015.

[25] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[26] William Levin, Hanne Kooy, J.S. Loeffler, and Thomas Delaney. Proton beam therapy. *British journal of cancer*, 93:849–54, 11 2005.

[27] Geert J. S. Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *CoRR*, abs/1702.05747, 2017.

[28] Jean Léger, Eliott Brion, Paul Desbordes, Christophe De Vleeschouwer, John A. Lee, and Benoit Macq. Cross-domain data augmentation for deep-learning-based male pelvic organ segmentation in cone beam ct. *Applied Sciences*, 10(3):1154, 2020.

[29] J Marsden and Haitham Hamoda. European cancer mortality predictions for the year 2019 with focus on breast cancer, by malvezzi m et al. (ann oncol 2019; 30: doi.org/10.1093/annonc/mdz051). *Annals of oncology : official journal of the European Society for Medical Oncology*, 05 2019.

[30] National Institute of Biomedical Imaging and Bioengineering, Maryland, USA. *Computed Tomography (CT)*, 2019.

[31] World Health Organization. Cancer. `https://www.who.int/news-room/fact-sheets/detail/cancer`. Accessed: March 30, 2020.

[32] Henryk Palus. Homogeneity critera for region-growing image segmentation in ihs coulour space. 06 1998.

[33] Di Perri and Geets. Ucl-iba umri seminar. unpublished, 2015.

[34] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[35] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[36] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[37] Dominic Eggbeer Richard Bibb and Abby Paterson. *Medical Modelling: The Application of Advanced Design and Rapid Prototyping Techniques in Medicine*. Cambridge, United-Kingdom, 2015.

[38] Hannah Ritchie and Max Roser. Causes of death. *Our World in Data*, 2020.

[39] Haralick Robert and Shapiro Linda. Image segmentation techniques. *Computer Vision Graphics and Image Processing*, 29(1):100–133, 1985.

[40] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[41] Anjany Sekuboyina, Diana Waldmannstetter, Jan Peeken, Univ.-Prof. Dr. Stephanie Combs, and Bjoern Menze. Deep reinforcement learning for organ localization in ct. 05 2020.

[42] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468, 1997.

[43] Carina Storrs. How much do ct scans increase the risk of cancer? *Scientific American*, 2013.

[44] OncoLink Team. Pictorial overview of the proton therapy treatment process. `https://www.oncolink.org/cancer-treatment/proton-therapy/overviews-of-proton-therapy/pictorial-overview-of-the-proton-therapy-treatment-process`. Accessed: April 1, 2020.

[45] J.R.R Uijlings, K.E.A. van de Sande, and T. et al. Gevers. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 2013.

[46] Vivek Verma, Chirag Shah, Jean-Claude M. Rwigema, Timothy Solberg, Xiaofeng Zhu, and Charles B. Simone II. Cost-comparativeness of proton versus photon therapy. *Chinese Clinical Oncology*, 5(4), 2016.

[47] Chuan Wu, Robert Jeraj, Gustavo H Olivera, and Thomas R Mackie. Re-optimization in adaptive radiotherapy. *Physics in Medicine and Biology*, 47(17):3181–3195, 08 2002.

[48] Xuanang Xu, Fugen Zhou, Bo Liu, Dongshan Fu, and Xiangzhi Bai. Efficient multiple organ localization in ct image using 3d region proposal network. *IEEE Transactions on Medical Imaging*, 38:1885–1898, 01 2019.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**