Physics-Guided Deep Learning for Volumetric-Modulated Arc Therapy Treatment Planning Quality Assurance

by

Thijs van der Hoeven

to obtain the degree of Master of Science
at the Delft University of Technology,

Date of Defence: 19 April, 2022

Student number:	4553853	
Project duration:	July 2021 - April 2022	
Supervisors:	Dr. Zoltán Perkó	TU Delft
	Dr. Tomas Janssen	NKI
	Dr. Rita Simões	NKI

TU Delft, Faculty of Applied Sciences, MSc. Applied Physics

Abstract

Treatment planning for radiation therapy is a complex process, as there are many machine parameters to determine for a treatment. To decrease the required labour and improve the plan quality, autoplanning systems have been developed, which can automatically generate high-quality treatment plans. These plans still have to be checked to ensure their quality, which means an independent and automatic quality assurance method is needed. This is where deep learning comes in, as recent advancements in computer hardware and artificial intelligence make it possible to quickly train neural networks for fast predictions of dose distributions based on the anatomy of a certain patient. Even though these models have proven capable of generating predictions comparable to the actual delivered dose, there is no guarantee that the distributions are actually deliverable. Therefore, this study is conducted to develop a novel deep learning model that uses physical information of how dose is delivered to constrain the neural networks and force realistic and deliverable dose distributions. The models presented in this study are implemented on prostate cancer patients treated with the Volumetric-Modulated Arc Therapy (VMAT) delivery technique.

First, the conventional Anatomy-based dose prediction model is further developed to improve the quality of the predicted dose distributions. The goal is to assess how accurate the predicted dose is from a deep learning model without any physical constraint. The model with U-net architecture has shown to be capable of predicting dose distributions of very high quality with average DVH differences of less than 1 Gy and errors in the dose coverage statistics on the PTV of less than 1%. Furthermore, the lack of deliverability was confirmed due to the missing ray effects in the dose distributions, showing no clear entry points of the external photon beam. This lack of ray effects was quantified and confirmed using the dip in the Sørensen-Dice Index around the 30% isodose contours, where the ray effects are most apparent.

Second, the Physics-guided prediction model was developed by combining the Anatomy-based dose prediction model with the newly developed Segment prediction model, which predicts Multi-Leaf Collimator (MLC) positions and beam intensity values for all directions from which the dose is delivered. These predictions are based on the prediction from the Anatomy-based dose prediction model and the Beam's Eye View images of the CT and the delineated structures. The segment prediction is used in a stand-alone dose engine, based on the Matrad dose calculation algorithm, to calculate a dose distribution from the predicted treatment plan, which is deliverable by definition. The performance of the Physics-guided prediction model is underwhelming, scoring considerably worse on most evaluation metrics compared to the Anatomy-based dose prediction model. The lack of performance is mainly caused by the poor performance of the Segment prediction model component.

To circumvent the issue of the Segment prediction model, a dose mimicking model has been developed and investigated, where a dose mimicking algorithm is used to mimic a prediction from the Anatomy-based dose prediction model by optimizing the segments on which the dose is delivered. This model has only been tested on a single patient but seems to be performing better than the Physics-guided prediction model, with the downside that an optimisation process is involved. Still, the performance is not on par with the Anatomy-based dose prediction model.

Further investigation on the Segment prediction model will be needed before the Physics-guided prediction model can be considered as a serious quality assessment tool. Once performance of the Segment prediction model has increased, it could prove an interesting model that could even replace an autoplanning system in the far future.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
3D-CRT	Three Dimensional Conformal Radiation Therapy
BCE	Binary Cross-Entropy
BEV	Beam’s Eye View
CNN	Convolutional Neural Network
CT	Computed Tomography
CTV	Clinical Target Volume
DICOM	Digital Imaging and Communications in Medicine
DVH	Dose Volume Histogram
GPU	Graphical Processing Unit
GTV	Gross Tumour Volume
HD U-net	Hierarchically Densely Connected U-net
HPD	Hybrid-Physics-Data
HU	Houndsfield Unit
IMRT	Intensity-Modulated Radiation Therapy
LINAC	Linear Accelerator
MAE	Mean Absolute Error
MALD	Mean Absolute Leaf Difference
MCO	Multi-Criteria Optimization
MLC	Multi-Leaf Collimator
MSE	Mean Squared Error
MU	Monitoring Unit
OAR/OARs	Organ at Risk/Organs at Risk
PGNN	Physics-Guided Neural Network
PSK	Point-Spread Kernel
PTV	Planned Target Volume
QA	Quality Assurance
ReLU	Rectified Linear Unit
SDI	Sørensen-Dice Index
SSD	Source-to-Surface Distance
TERMA	Total Energy Release per unit Mass
TPS	Treatment Planning System
VMAT	Volumetric-Modulated Arc Therapy
WMSE	Weighed Mean Squared Error

Symbols

Symbol	Definition	Unit
b	Artificial node bias	-
c	Leaky ReLU slope coefficient	-
D	Dose array	[Gy]
d	penetration depth	[cm]
$d_{i,j}$	Dose influence i^{th} voxel from pencil beam j	-
d_{in}	Convolutional layer input size	-
d_{out}	Convolutional layer output size	-
E	Photon energy	[MeV]
$g(x)$	Activation function	-
h	Artificial node output	-
I	Convolutional layer input array	-
K	Convolution kernel array	-
k	Kernel size	-
L	Loss	-
m_{pr}	Predicted beam intensity	[MU]
$\mathbf{m}_n$	n^{th} Iteration first moment vector	-
$\widehat{\mathbf{m}}_n$	n^{th} Iteration bias-corrected first moment vector	-
m_{gt}	Ground truth beam intensity	[MU]
N	Number of elements in array	-
P	Point-Spread Kernel	-
p	Zero-padding value	-
p^i	i^{th} DVH point value	-
q	Bixel weight	-
S	Convolutional layer output array	-
s	Stride value	-
T	Total Energy Release per unit Mass	-
$\mathbf{v}_n$	n^{th} Iteration second moment vector	-
$\widehat{\mathbf{v}}_n$	n^{th} Iteration bias-corrected second moment vector	-
w_i	WMSE weight element	-
$\mathbf{w}$	Artificial node weight vector	-
$\mathbf{x}$	Input features	-
x_i	Input element	-
$\widehat{x}_i$	Normalized element	-
y_i	Ground truth element	-
$\widehat{y}_i$	Predicted element	-
α	Step size	-
β	Exponential decay rate	-
γ	Linear activation scale	-
δ	Linear activation bias	-
ϵ	Learning rate	-
θ_j	Model parameter	-
λ	Weight decay	-
μ	Photon attenuation coefficient	[cm^{-1}]
ρ	Tissue density	[kg/m^3]
σ	Standard deviation	-
τ	Numerical stability term	-
ψ	Mean value	-
Ψ	Primary energy fluence	[J/m^2]
Ψ_0	Initial photon fluence	[cm^{-2} s^{-1}]
∇	Gradient operator	-

1
Introduction

Throughout the last century, the world has made enormous advancements in the fields of medicine and health care, meaning that many diseases that were considered fatal in earlier times are now treatable. In the field of cancer treatment, significant progress has been made as well, and a larger percentage of cancer patients is cured than ever before. Still, cancer remains a dangerous disease, as over 18 million people were diagnosed with cancer in 2020 worldwide, and nearly 10 million people died due to the disease [1]. Moreover, it is expected that the number of yearly new cancer cases will grow to almost 29 million in 2040 [2]. These numbers demonstrate that research on cancer treatment is an essential field of science and that there is still enough room for improvement.

One of the primary modalities of treating cancer is radiation therapy, which can be divided into internal and external radiation therapy. With external radiation therapy, the patient is irradiated with an external beam of high energy photons or protons, damaging the cancerous tissue. As cancer cells are less capable of repairing themselves after being exposed to radiation compared to healthy cells [3], this is an effective method of killing the tumour, as long as the dose to the healthy tissue is kept as low as possible. Treatment planning is the process of setting up the dose delivery machine so that the prescribed dose is delivered to the tumour while keeping the dose to healthy tissue and vital organs as low as possible. This is a complex and labour-intensive manual process, consuming a lot of the available time of the dosimetrists. Moreover, the treatment plan quality heavily depends on the available time and experience of the dosimetrist.

Auto-planning systems have been developed to deal with these issues, which can automatically generate high-quality treatment plans, using the patient CT and prescription doses to specific structures within the patient. Even though these systems save a lot of time, the resulting plans are not always optimal or clinically acceptable, and therefore in some cases, replanning is required. This means that all plans still have to be checked manually, which partially negates the advantages of treatment planning systems. This calls for an independent and automatic quality assurance method to evaluate the automatically generated treatment plans.

One such quality assurance method is Knowledge-Based Planning (KBP), where a quick prediction of the Dose Volume Histogram (DVH) can be made based on earlier generated treatment plans. The predicted DVH can be compared to the plan generated by the auto-planning system to assess the quality of this plan. Research has shown that the KBP method works well as a quality assurance method [4].

As a result of the tremendous advancements in computing power and artificial intelligence in the past few decades, deep learning has found its way into almost all fields of science through many different applications. Since the development of the U-net by Ronnenberger et al. in 2015 [5], deep learning models have also been used to predict high-quality and optimal dose distributions. As these predictions are made quickly, they could serve as an alternative quality assurance method for the treatment planning systems. By comparing the automatically planned dose to a predicted dose distribution, an assessment can be made on the planned dose to see if it is clinically acceptable. The advantage of this method over KBP is that there is much more spatial information available, which can also be easily translated into a DVH. This means that a more in-depth assessment can be done on the quality of the generated treatment plan by making a per-voxel comparison[6].

The only issue with the conventional dose prediction models is that there is no way to confirm that a predicted dose is realistic or even deliverable. This is a problem when comparing the predicted dose to the planned dose. Trying to achieve similar qualities to a dose distribution that is not physically achievable could prove impossible. The problem could be resolved by adding physical knowledge on how the dose is delivered to the deep learning models, forcing deliverability .

In this study, two primary investigations have been done. First, a conventional dose prediction neural network has been developed further, which was created by a previous Master student Meerbothe [7]. The goal is to obtain high-quality dose distributions from the model and confirm that the predictions lack deliverability.

Furthermore, a novel approach has been taken to predict dose distributions by considering the physics behind the dose delivery. This is done by deploying the previously developed dose prediction neural network and using a second neural network capable of predicting the treatment plan parameters, which define the machine setup for dose delivery. Combining this secondary neural network with a dose engine algorithm produces a “predicted” dose, which is deliverable by definition as the dose is calculated from the predicted treatment plan. The additional advantage of this method is that the treatment plan parameters needed to deliver the dose are also available after making a prediction. This means that an even more in-depth assessment can be done, and in the far future, this type of model could potentially replace auto-planning systems entirely. The dose prediction models have been trained on and tested with two datasets of already treated prostate cancer patients who have been treated with Volumetric-Modulated Arc Therapy treatment.

To develop the Physics-guided prediction model, a Segment prediction model is first created, capable of predicting the treatment plan parameters of a VMAT treatment. Next, a stand-alone dose engine is made, which can compute a dose distribution from a predicted treatment plan. Lastly, The dose prediction model, segment prediction model and dose engine are combined to create the Physics-guided prediction model, capable of predicting both a dose distribution and a corresponding treatment plan.

In Chapter 2, the background theory required for a detailed understanding of the work done is explained. Then, in Chapter 3 an elaborate explanation is given of how the different models and components have been implemented. Next, the resulting performance of these various components are assessed and evaluated in Chapter 4, and these results are then analysed and discussed in Chapter 5. Lastly, a conclusion and further recommendations on future research are presented in Chapters 6 and 7.

This paper is part of the Master thesis, conducted at the Delft University of Technology, at the faculty of Applied Sciences, for the master of Applied Physics. The work presented here will be publicly defended on 19-02-2022 to obtain the degree of Master of Science. The project is a collaboration between the Reactor Institute Delft (RID) and the Netherlands Cancer Institute (NKI) at the Antoni van Leeuwenhoek hospital in Amsterdam. The patient data and hardware capable of training neural networks have been provided by the NKI.

2 Theory

In this chapter, all the relevant background theory is explained, necessary to understand the details of the project entirely. In Sections 2.1, 2.2 and 2.3, the basics of radiation therapy and treatment planning are discussed. In Sections 2.4, 2.5 and 2.6, the basics of deep learning and the connection of deep learning to treatment planning are explained.

2.1. Radiation Therapy

Approximately 50% of cancer patient treatments involve some form of radiotherapy [8]. This shows how radiotherapy is an essential modality in cancer treatment, besides chemotherapy and surgery. During radiation treatment, the patient is irradiated with high-energy photons or protons, which deposit their energy somewhere along the way. This deposition of energy causes damage in the tissue, specifically to the DNA in the cells, which could potentially induce cell death. Radiation treatment is effective because cancer cells are less capable of repairing themselves after being exposed to radiation compared to healthy cells [3]. Therefore, irradiating a patient at specific locations with an appropriate dose can kill the cancer cells but spare the healthy cells. This type of treatment helps cure patients; it can also be used for palliative purposes when curing the patient is no longer viable [9].

2.1.1. Treatment Device

There are two main types of radiation therapy: internal and external radiation therapy. With internal radiation therapy, a source of radiation is placed inside the patient's body, ideally as close as possible to the tumour. With external radiation therapy, the patient is irradiated with an external photon or proton beam directed at the tumour. In this project, the focus lies on external radiation therapy with a photon beam.

The most common machine used for photon radiotherapy is the Linear Accelerator (LINAC). Figure 2.1 gives an overview of the inner workings of the LINAC. This device employs an electron gun to deliver high energy photons to the patient. The electrons released by the electron gun are accelerated using accelerating waveguides, and they are directed onto an X-ray target using a bending magnet. At the target, the high energy photons used for the treatment are released as Bremsstrahlung radiation, created by the incoming electrons. The high energy photons are then directed towards the tumour through proper alignment of the treatment couch with the photon beam. The entire system is located inside a rotating arm called the gantry, which can move around the patient to deliver the dose from any desired direction. Ideally, the region that is to be irradiated is aligned with the isocentre during treatment, which is the intersection of the gantry rotation axis and the photon beam centre.

To be able to deliver as much dose as possible to the tumour while keeping the dose delivered to healthy cells as low as possible, it is essential to be able to adjust the shape of the photon beam specifically for each patient. This way, the beam can be shaped like the irradiated tumour, and the healthy tissue directly surrounding the tumour can be spared. Three different beam limiting devices are used to shape the photon beam into the desired shape. First, the beam passes through 2 sets of jaws, which are dense metal collimators blocking the radiation. These two sets are combined to create a rectangular beam aperture, of which the dimensions can be modified as desired. In some cases, only one set of jaws is used. After the jaws, the photon beam passes through the Multi-Leaf Collimator (MLC). This is a set of thin tungsten slabs that block the incoming radiation and can be adjusted individually to create an intricate beam shape. This device makes it possible to shape the photon beam tightly around the tumour contour. Figure 2.2 presents an example of these beam limiting devices.

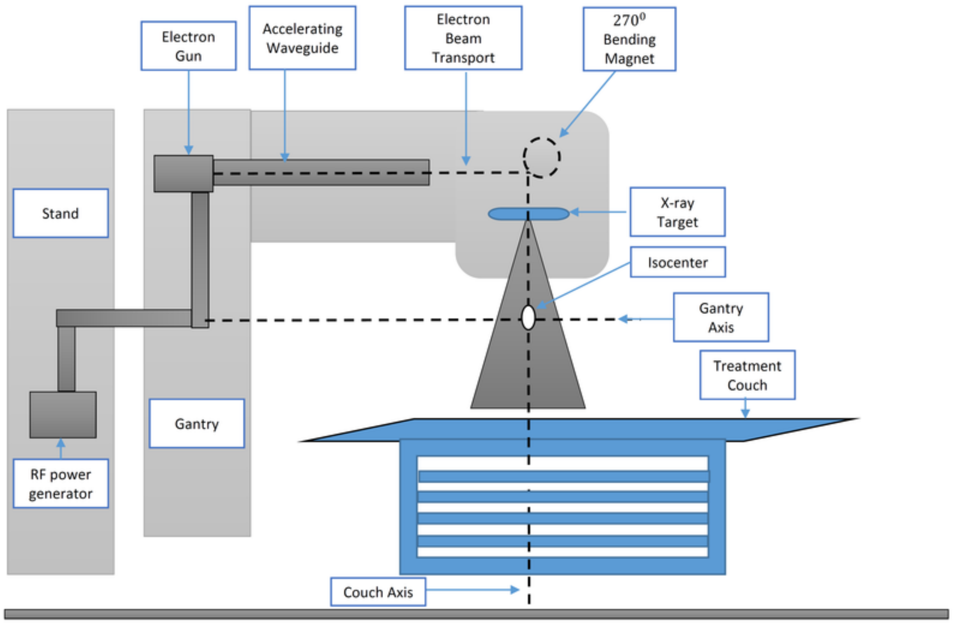


Figure 2.1: Schematic overview of the Linear Accelerator (LINAC) [10].

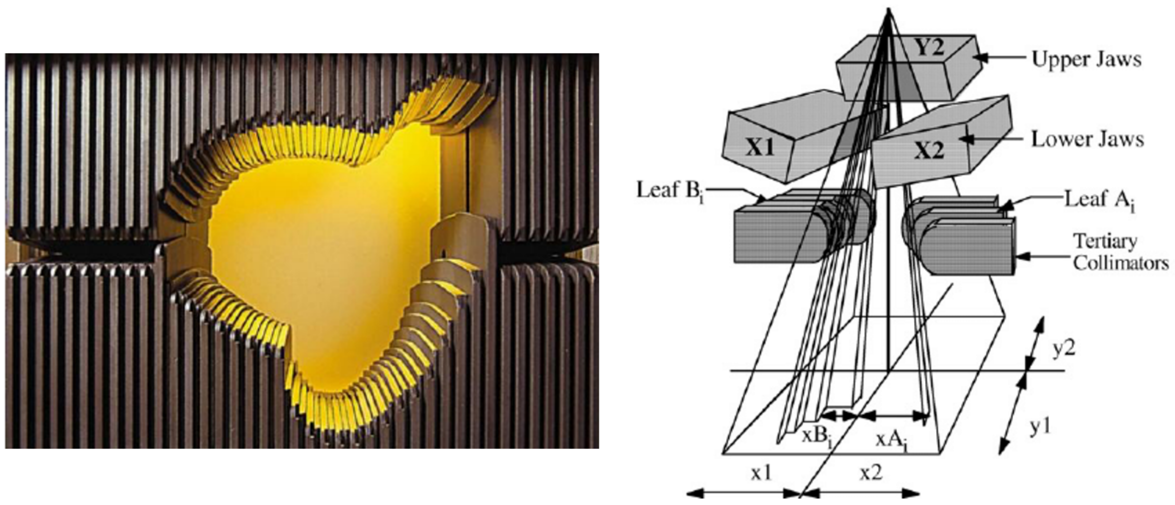


Figure 2.2: Overview of the beam limiting devices used in the LINAC [11], [12]. The image on the left shows an example configuration of the Multi-Leaf Collimator (MLC). The schematic on the right shows how combining two sets of beam limiting jaws, and the MLC can create an intricately shaped beam.

2.1.2. Radiotherapy Techniques

With the LINAC, there are a lot of possibilities on how to deliver the dose due to the many machine parameters to determine, such as the gantry angle, couch positioning, beam aperture specification and beam intensity. This is also why multiple techniques are used to deliver dose with a LINAC.

One of the older techniques emerged in the 1980s as Computed Tomography (CT) information became more widely available and is called Three Dimensional Conformal Radiation Therapy (3D-CRT) [13]. With this technique, the dose is delivered from multiple angles around the patient. The beam aperture could be adjusted so that the beam was shaped according to the tumour's contour when viewed from the gantry head position for each different gantry angle. By using multiple angles, irradiation of essential organs can be avoided as much as possible.

Around the late 1990s, a new and potentially better dose delivery technique was developed called Intensity-Modulated Radiation Therapy (IMRT) [14]. The main difference from 3D-CRT is that the beam intensity can be spatially modulated over the aperture. This is done by adjusting the aperture created by the MLC during the dose delivery from a specific angle. This gives more machine parameters to define during treatment and therefore allows for adjusting the delivered dose even more. This way, the high dose region can be shaped tighter around the tumour.

In 2008, the first Volumetric-Modulated Arc Therapy (VMAT) algorithm was developed by Otto [15]. This technique is similar to IMRT, as the MLC is also continuously moving during dose delivery. The only difference is that with VMAT, the beam intensity changes continuously, and the gantry moves over a specified arc during the delivery. This means that the dose is delivered no longer from a specific set of gantry angles but a continuous arc surrounding the patient. The main advantage of VMAT over IMRT is that it allows for much quicker dose delivery [16].

For the rest of this thesis, the focus will lie on the VMAT technique for dose delivery. In Figure 2.3, a comparison is shown between dose distributions delivered with the three different techniques mentioned. As can be seen from the figure, a more continuous dose distribution is visible for the VMAT delivery technique due to the continuous rotation of the gantry during delivery.

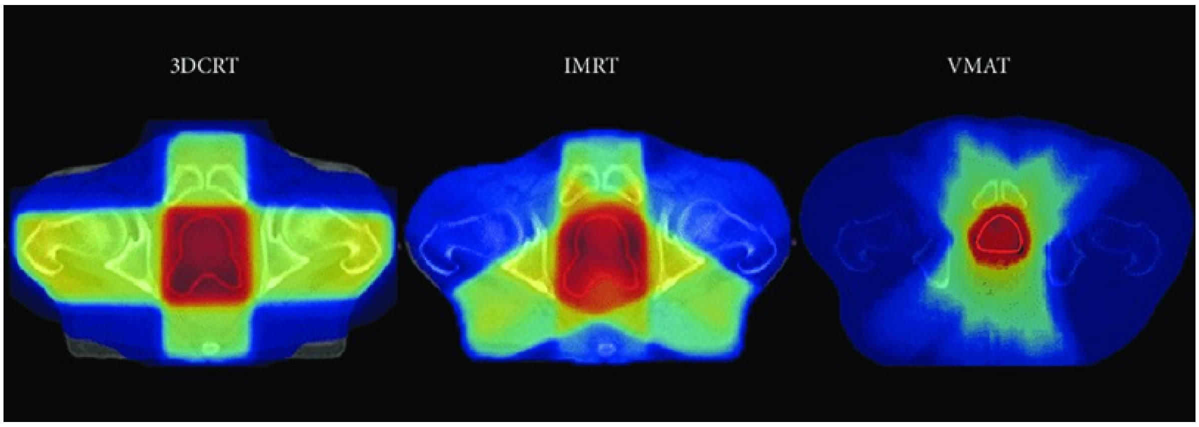


Figure 2.3: Comparison of a dose distribution delivered with 3D-CRT (left), IMRT (centre) and VMAT (right) [17].

2.2. Treatment Planning

Converting a medical prescription of dose to the different regions into a radiation plan is called treatment planning. Treatment planning is a highly complex process, as there are many machine parameters to be determined in such a way that the optimal dose is delivered to the patient. This optimal treatment plan can be very different from patient to patient and therefore requires a lot of work for every individual. With the VMAT dose delivery technique, more optimal dose distributions can be delivered compared to 3D-CRT, for example. Still, the treatment planning process is also much more complex, as the specific positioning of the beam needs to be determined at each point of the continuous arc on which the dose is delivered. Treatment planning is a multi-step process and always starts with patient imaging.

2.2.1. Patient Imaging and Delineation

To generate an optimal treatment plan for a particular patient, the patient's anatomy should be mapped. This is often done by making a CT scan of the patient. A CT scan is made by rotating an X-ray emitting source and receivers around the patient. The attenuation of the X-rays is detected by the receivers, and the measurements from all different angles are then combined to create a three-dimensional tomographic image of the patient using a reconstruction algorithm. This 3D image is made up of 3D pixels, called voxels, and the intensity of each voxel is the measured linear attenuation coefficient. The CT is often depicted as multiple grayscale images, where the grayscale intensity of the pixels defines the attenuation of the tissue in Houndsfield Units (HU). The conversion from the measured linear attenuation coefficient to HU values is done using Equation 2.1:

$$HU = 1000 \times \frac{\mu - \mu_{\text{water}}}{\mu_{\text{water}} - \mu_{\text{air}}}, \tag{2.1}$$

where μ stands for the measured photon attenuation coefficient in a specific voxel, μ_{water} for the photon attenuation coefficient measured in water and μ_{air} for the photon attenuation coefficient measured in air. The Houndsfield Unit is calibrated so that a value of 0 HU corresponds to the photon attenuation of water, while a value of -1000 HU corresponds to the photon attenuation of air [18]. The last step required is to convert the HU values to a corresponding electron density. This is needed as CT uses photons with an energy in the keV range, while photons in the MeV range are used during radiotherapy. In practice, this conversion is often based on a calibration process involving a tissue surrogate phantom [19]. The electron density is then directly used to calculate the attenuation for a specified nominal energy of photons.

Once a CT image has been made from a patient, the next step is to interpret this image. Specifically, the goal is to define contours of individual structures present in the patient, such as vital organs, bones, and tumours. The process of creating these contours is called delineation. It is generally a manual process, where for each voxel needs to be determined whether or not it is part of a vital organ or other structure.

Several delineations are of importance for treatment planning. First, a delineation should be made of the tumour visible on the CT. This delineation is called the Gross Tumour Volume (GTV). As it is often the case that close to the GTV microscopic tumour volumes are present, not visible on the CT, a slight margin is taken around the GTV defining the Clinical Target Volume (CTV) [20]. Lastly, as some uncertainties in dose delivery, patient imaging and patient movement can cause slight misalignment between the patient CT and the delivered dose, another margin is taken around the CTV, defining the Planned Target Volume (PTV). The PTV is then the structure considered to be the volume that has to be irradiated during treatment. Apart from the tumour, vital organs close to the cancer are also delineated and defined as Organs At Risk (OARs), to which the delivered dose should be as little as possible. In Figure 2.4, an example is shown of a delineated CT slice.

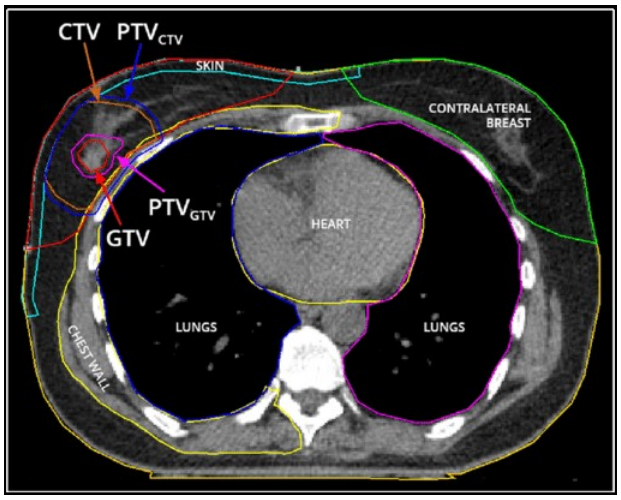


Figure 2.4: Example slice of a delineated CT slice [21]. The GTV, CTV and PTV delineations can be seen here and the different OARs.

2.2.2. Treatment Plan Optimisation

With a CT image made and delineated, the patient's anatomy is defined. The next step is to determine a treatment plan for this specific patient. This is a complex Multi-Criteria Optimisation (MCO) problem, as many constraints and criteria are taken into account. For example, a particular minimum dose should be delivered to all voxels belonging to the tumour, but also the dose to certain OARs should remain below a particular threshold. The fact that there are multiple constraints makes the problem complex.

Optimisable Parameters for VMAT

First, it is essential to know the exact parameters that need to be defined for a treatment plan. For VMAT, this is not as straightforward as for 3D-CRT, as the dose is delivered on a continuous arc, with constant change in beam intensity and MLC shape. This continuity needs to be discretised to define a treatment plan, which is done by defining specific points on the rotation arc, called control points. For each control point, a particular beam intensity and shape of the MLC is defined, which means that when the gantry is at this point of the arc during treatment, the machine should be set to that specific MLC aperture and beam intensity. The combined configuration of the MLC and beam intensity is called a segment. Figure 2.5 gives a visualisation of how these control points are defined.

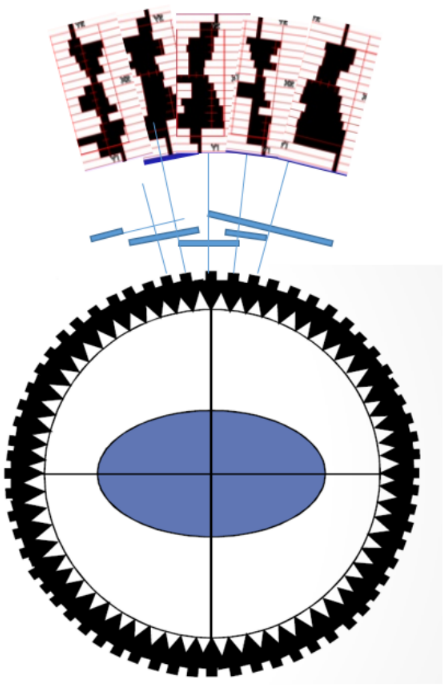


Figure 2.5: Schematic visualisation of the definition of a VMAT plan [22]. The rotation arc is discretised into control points, denoted by the black arrows. For each control point, a beam intensity, denoted by the blue bar, and an MLC aperture, indicated by the figure above the blue bar, should be defined.

Methods of Optimisation

Two general methods are used to obtain such a treatment plan: forward planning and inverse planning. Forward planning is the more classical method of getting a treatment plan, where first the treatment plan parameters are defined, and then the dose in the patient is calculated by simulating the treatment. If the calculated dose is not satisfactory, the treatment plan parameters are altered, and the dose is recalculated. This process continues until a dose distribution is obtained where the dose delivered to the different structures is satisfactory.

In the case of VMAT, the amount of treatment plan parameters is extremely large. The treatment machine considered in this study consists of an 80 leaf MLC (i.e. 40 leaf pairs), and the treatment plan is defined on 140 control points, which means that there are more than 10,000 parameters to define. Because of this, it is not easy to decide, based on the calculated dose, how the treatment plan parameters should be altered to obtain a better dose distribution. To solve this problem, a different method of producing a treatment plan is more often used, which is inverse planning.

With inverse planning, certain constraints are first defined, limiting the allowed dose on specific structures. During the optimisation process, these constraints should be kept at all times. In the first iteration of optimisation, the treatment plan parameters are found for which these constraints are met. Then the optimisation objective is altered by adding so-called criteria, which limit the allowed dose further. If a criterion goal has been completed, the optimisation objective is changed further with a new criterion [23].

A less common kind of optimisation method which can be used for treatment planning is dose mimicking. With dose mimicking, a dose distribution is defined, which is considered the desired dose distribution for the treatment. Then, initialised treatment plan parameters are inserted into a dose calculation algorithm to calculate the dose. A function can then be designed which evaluates the difference between the desired dose and the calculated dose, which should be minimised by optimising the treatment plan parameters. This optimisation is continued until no further improvements can be made and a final dose distribution with corresponding treatment plan parameters is produced.

Pareto Optimality
Ultimately, the goal of MCO is to obtain a Pareto-optimal plan. This means that no improvements can be made on a criterion without deteriorating another. Therefore, a Pareto-optimal plan is typically better than a plan that is not Pareto-optimal. However, not all Pareto-optimal plans are also clinically favourable, which is also shown in Figure 2.6.

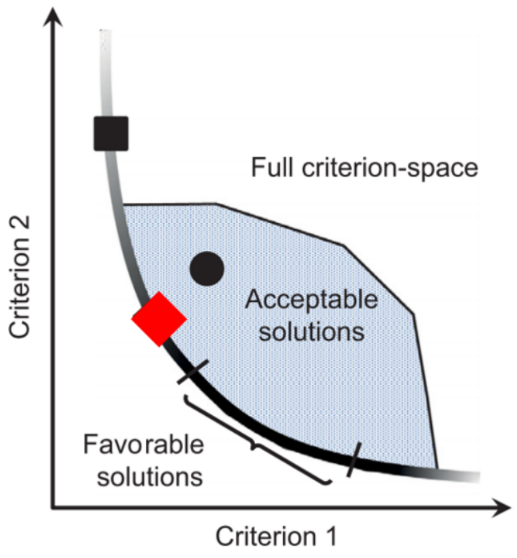


Figure 2.6: Graph denoting a two-dimensional criterion space [24]. The black line represents all Pareto-optimal plans, where improvement on one criterion can not be made without the sacrifice of another.

From Figure 2.6 can be seen that a specific region on the Pareto-optimal line is considered clinically favourable since some criteria are more important than others. For example, the black square plan is a Pareto-optimal plan but scores very poorly on Criterion 2. The red diamond plan has a more balanced score between criteria 1 and 2 and is closer to the favourable solutions. Even the plan denoted by the black circle might be better than the black square plan as it is in the region of acceptable solutions, even though the plan is not Pareto-optimal.

In the end, the inverse planning method should always result in a Pareto-optimal plan. However, a clinical expert still needs to ensure that this plan is also clinically favourable. Note that in reality, there are more than two criteria involved in the optimisation process, which means that the criterion space consists of many more dimensions than two [24]. Furthermore, it can be challenging for some criteria to express them in terms of Pareto-optimality. An example of this is dose conformity, which is often an important requirement for the dose distribution but is challenging to define as a quantifiable criterion.

2.2.3. Dose Calculation Algorithms

Dose calculation algorithms are used to convert a treatment plan into a dose distribution. The issue with dose calculation is that, in reality, photons behave in a non-deterministic manner. At each point in time, the travelling photon has a certain probability of interacting with the tissue differently. To accurately model this, the path of individual photons would have to be modelled with random chances of interaction, which is a very resource-intensive calculation method. For both forward and inverse planning, the dose delivered within the patient has to be calculated many times during the optimisation process. Without the dose calculation, there is no way of knowing how changes in the treatment plan affect the delivered dose. That is why dose calculation algorithms have been developed that are much faster, with the downside that they are only an estimation of the actually delivered dose.

Convolution Algorithm

A type of dose calculation that is commonly used is the convolution method. The idea behind this method is that all photons behave similarly on average. The first step in this method is to define on each voxel within the patient how much energy is released by primary photon interactions, which interact in that specific voxel. This is defined as the Total Energy Release per unit Mass (TERMA). The TERMA has to be calculated for each voxel that lies in the “line of sight” of the photon beam, which is done as:

$$T(\mathbf{r}) = \frac{\mu(\mathbf{r})}{\rho(\mathbf{r})}\Psi(\mathbf{r}), \tag{2.2}$$

where $T(\mathbf{r})$ stands for the TERMA at location **r**, $\mu(\mathbf{r})$ for the photon attenuation coefficient at point **r**, $\rho(\mathbf{r})$ for the local density at point **r** and $\Psi(\mathbf{r})$ for the energy fluence of the primary photons. The primary energy fluence is calculated as

$$\Psi(\mathbf{r}) = \Psi_0(r_\perp)Ee^{-\mu(\mathbf{r})d(\mathbf{r})}, \tag{2.3}$$

where $\Psi_0(\mathbf{r}_\perp)$ denotes the initial photon fluence at a point on the patient surface located $r_\perp$ away from the beam centre, E for the nominal photon energy and $d(\mathbf{r})$ for the distance from the point where the photon ray enters the patient to point **r** [25].

With the TERMA, the energy released in the patient is defined. However, the energy released at a specific voxel is not all deposited in that voxel due to many types of secondary and tertiary interactions. The distribution of energy deposition around the primary interaction point is given on average by a Point-Spread Kernel (PSK). As the TERMA essentially defines how many photons have their primary interaction in all voxels, and the PSK defines how the energy is distributed around this primary interaction point, a convolution of the two leads to a total dose distribution. The convolution can be calculated as

$$D(\mathbf{r}) = \int T(\mathbf{r}')P(\mathbf{r}-\mathbf{r}')d\mathbf{r}', \tag{2.4}$$

where $D(\mathbf{r})$ stands for the calculated dose distribution, T for the TERMA and P for the PSK.

The advantage of this dose calculation algorithm is that it is relatively accurate and works well for inhomogeneous media. The disadvantage, however, is that it is still quite a resource-intensive computation since, for each voxel in the patient, the dose has to be calculated by taking into account the PSK from all other voxels in the patient [26].

To reduce computation time further, the calculation can be further approximated by using a collapsed cone convolution algorithm. Instead of defining the PSK as a full 3D array, the PSK is discretised into several rays moving outwards from the point of interaction. The dose deposition value is then no longer defined in each voxel but only along these individual rays. The deposited dose in a particular voxel close to the point of interaction is then determined by the value present at the nearest point on the nearest ray. This way, fewer computations are done in the convolution and the calculation time is much shorter [27].

Pencil Beam Algorithm
Another commonly used algorithm to calculate the dose is the pencil beam algorithm. This algorithm is much less complex than convolution algorithms and much faster. The downside, however, is that it is also less accurate and does not work very well for inhomogeneous media, such as the lungs.

First, a kernel is defined of the dose deposition caused by a thin beam, called a pencil beam. This is often done by performing Monte-Carlo calculations of a tiny beam on a water phantom. This narrow beam is defined as the pencil beam kernel, which can then be scaled with a so-called bixel weight to obtain the desired intensity and adjusted for heterogeneities within the patient. Note that only heterogeneities in the longitudinal direction on the central axis are corrected, which is why this model only works well for homogeneous regions [28]. The beam aperture can be divided into individual small beamlets, and for each, a pencil beam dose can be calculated inside the patient. These pencil beam doses can then be combined to obtain the final dose distribution. Figure 2.7 shows a visualisation of the beam aperture discretisation.

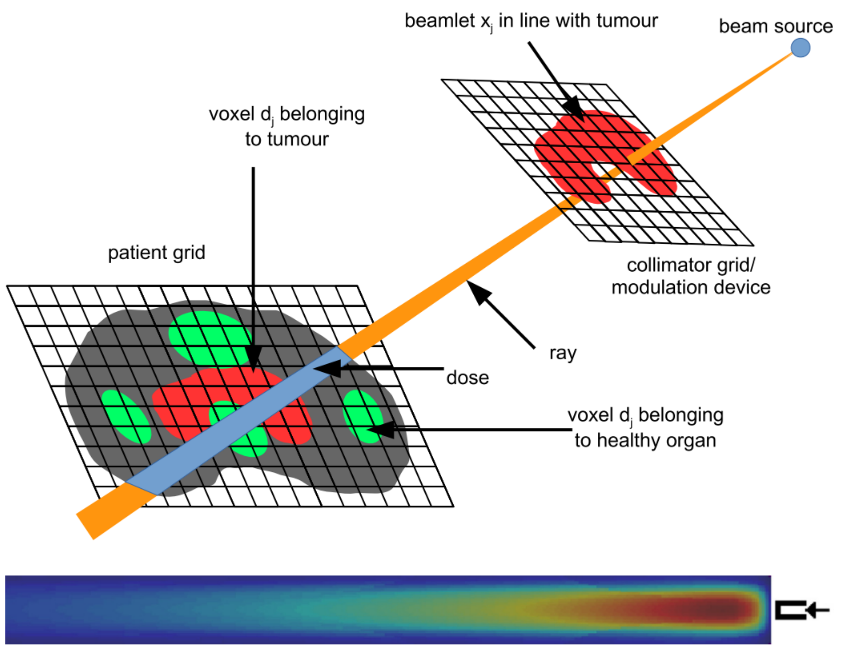

beamlet x_j in line with tumour
beam source
voxel d_j belonging to tumour
patient grid
collimator grid/ modulation device
ray
dose
voxel d_j belonging to healthy organ

Figure 2.7: Visualisation of the discretisation of the beam aperture into individual beamlets [24]. For each beamlet in the aperture, a corresponding pencil beam dose is calculated within the patient. These pencil beam doses are then combined to obtain the final dose.

To increase the computation speed even further, the pencil beam kernel can be decomposed using a singular value decomposition, where the 3D kernel is expressed by three much smaller arrays, speeding up the calculations considerably. The size of these arrays can be chosen arbitrarily and determines the resulting loss of accuracy and speed up of computation time [29].

2.3. Auto-planning and Quality Assurance

From Section 2.2 could be seen how treatment planning is highly complex, which means that it is a labour-intensive process, requiring many iterations of optimisation before an optimal treatment plan is obtained. To reduce the amount of labour needed for treatment planning, Treatment Planning Systems (TPS) have been developed, which automatically generate a treatment plan from a given delineated CT. An example of such a TPS is Pinnacle[3], which has been shown to produce good quality treatment plans compared to manual plans [30].

A TPS generates a treatment plan by following the same steps as in the inverse optimisation method. The first step is defining the constraints and criteria for the optimisation process, which is done by providing a so-called wish-list to the TPS. The constraints of the wish-list are hard constraints, which should never be compromised during the optimisation process. Common examples of these constraints are a minimum and maximum dose to the PTV or a maximum dose to a specific OAR.

During the second step, a first optimisation round is done, where the TPS generates a plan in which the specified constraints are met, but no further optimisation is done. Next, the TPS investigates the criteria defined on the wish-list. These criteria have been defined with a certain priority, showing the more and less critical criteria. An example of a criterion is to minimise the dose to a specific organ below a certain threshold. The TPS will take each criterion one by one, starting from the highest priority, and optimise the plan until the criterion goal is achieved. Note that once a criterion goal has been met, this criterion is added to the constraints list. When moving on to the following criterion, the TPS will still consider all previous criteria and constraints. The optimisation often ends somewhere along the criterion list when it is can no longer meet a criterion goal. If all criteria have been achieved, the TPS will try to minimise the dose to the OARs even further until no improvements can be made without compromising the criteria and constraints.

The advantages of a TPS are numerous. First of all, due to the labour-intensive nature of manual treatment planning, a TPS can save a lot of time for technicians or physicists, allowing more treatment plans to be produced in the same amount of time. Moreover, due to the subjective nature of the decision-making optimisation, the quality of the manual treatment plans heavily depend on the experience and available time of the dosimetrist [31]. With auto-planning, the generated plans are much more consistent. Furthermore, plans generated by a TPS often prove to be better or on par with manually planned treatment plans [30], [32].

The only downside is that in a small number of cases, the plan generated with a TPS is not good enough, and some adjustments are required before the plan is delivered. This means that even with auto-planning, a dosimetrist has to evaluate each plan to assess whether or not the plan is of good quality. As this means that manual labour is still required before a final plan is obtained, the benefits obtained from auto-planning are partially negated. Therefore, the need has arisen for an automatic Quality Assurance (QA) method to assess whether or not a generated treatment plan is the most optimal plan for a particular patient. Currently, a large field of research is dedicated to developing deep learning models for the specific task of QA of automatically generated treatment plans.

2.4. The Basics of Deep Learning

In this thesis, deep learning models are designed and implemented. Therefore, this section is dedicated to a brief description of the basic mechanics behind deep learning. The section is divided into a few subsections. First, a short description of what deep learning is within the field of machine learning is given. Next, the mechanics of neural networks are introduced, which is the most commonly used deep learning algorithm. Then, a more specific discussion on convolutional neural networks is held, after which some standard loss functions are given. Lastly, the mechanics behind backpropagation and optimisation are explained.

2.4.1. Machine Learning Basics

Deep learning is part of a larger scientific field called machine learning, which started to develop in 1957 when Frank Rosenblatt from Cornell University created a machine called the “Perceptron” capable of letter recognition [33]. Since then, the field has grown significantly, and today machine learning is applied in almost every branch of society.

Machine learning is a category of algorithms that can “learn” from a given dataset to improve their performance. This is done by making predictions based on given input data and improving themselves by learning from their mistakes. An example application of machine learning is speech recognition, where an audio input can be translated to text output. In this example, the algorithm will improve itself by “training” on a large dataset of input audio and trying to predict the correct text.

Two main methods are used to train a machine-learning algorithm: supervised and unsupervised learning.

- With supervised learning, the algorithm generates predictions for many different inputs, and its predictions are compared to the corresponding desired output, called the ground truth. The difference between the ground truth and the prediction is then used to improve the machine learning algorithm. Within the category of supervised learning, there are two subcategories: classification and regression. A classification algorithm tries to label certain input data, which means that the input is mapped to a discrete output. An example application of this is trying to decide whether an email should be classified as spam or not. A regression algorithm will try to find a continuous relation between input values and output values and maps the input to a continuous output. An example application of this is determining the housing price based on a certain set of input features.
- With unsupervised learning, no ground truth is used, and the machine learning algorithm is given the task to find structure in the input data. A typical application for this type of learning is defining clustered groups of data [34]. Figure 2.8 shows simple examples of these different types of machine learning.

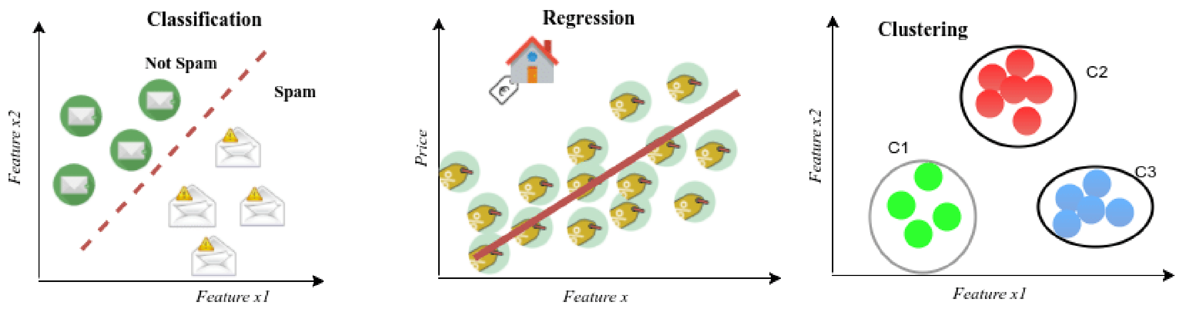


Figure 2.8: Three examples of the different machine learning categories [35]. The circles and icons represent data points. The left figure shows an example of a classification algorithm, where the algorithm needs to decide what email is spam and what is not based on features $x1$ and $x2$. The middle graph shows a regression example, where a relation is found between the housing price and feature $x1$. The right graph show an example of unsupervised learning, where the data is structured into three different clusters, based on features $x1$ and $x2$.

A complication within machine learning is that for these algorithms to work, it needs to be very clear what type of input data is required to make predictions on the output. This means that expert knowledge is required on the subject to decide what features should be provided as input to the algorithm to make accurate predictions.

A sub-field within machine learning called deep learning circumvents this problem by incorporating feature extraction in the learning process. This means that the algorithm does not only learn to find a relation between input and output but also to extract the essential features from a large stack of input data. Therefore, it is no longer necessary to have expert knowledge on a particular topic to create a deep learning algorithm.

2.4.2. Neural Networks

The most commonly used algorithm for deep learning is the neural network. The name is derived from the neuron cells in the human brain, as the behaviour of the human brain is mimicked in these algorithms. Neural networks consist of artificial neurons or nodes, which are connected together to form the network. Figure 2.9 shows the schematic of a simple neural network.

As is shown in Figure 2.9, the nodes are grouped together in vertical groups called layers. The first layer, the input layer, contains all the input data in the form of different nodes. As this is an example of a fully connected network, all input nodes are connected to all nodes from the second layer, which is defined as the first hidden layer. This means that all input data is used by each second layer node to perform computations and generate a single output. It is called a hidden layer as the user of the neural network will never see the values produced by this layer. These values are then sent to all nodes in the second hidden layer, which again produces some output. Lastly, the output of the three third-layer nodes is sent to the output layer, which is, in this case, a single node. This node performs some final computations to generate a single output value. The architecture of a neural network can differ based on the application. Examples of possible alterations are using more hidden layers or a different amount of nodes per layer.

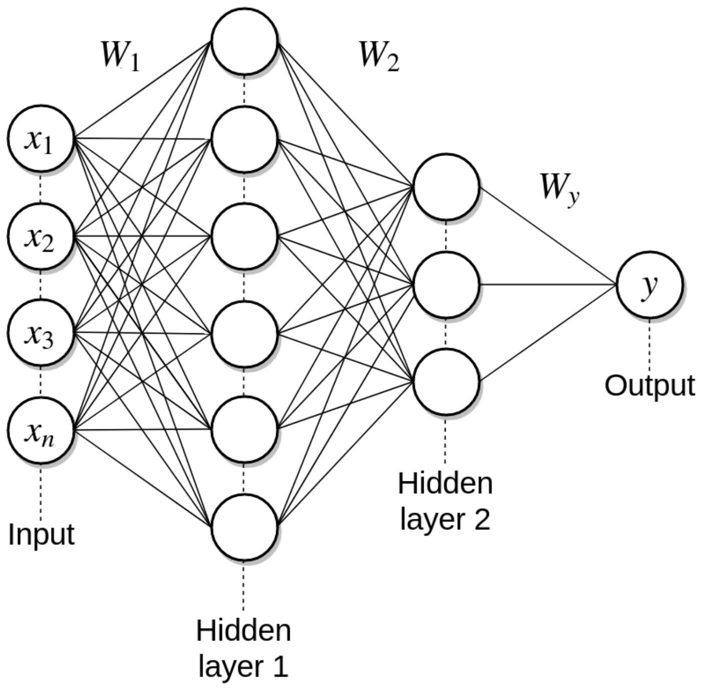


Figure 2.9: Schematic of a simple neural network [36]. This specific type of neural network is called a fully connected network, as all nodes from a previous layer are connected to all nodes from the next layer

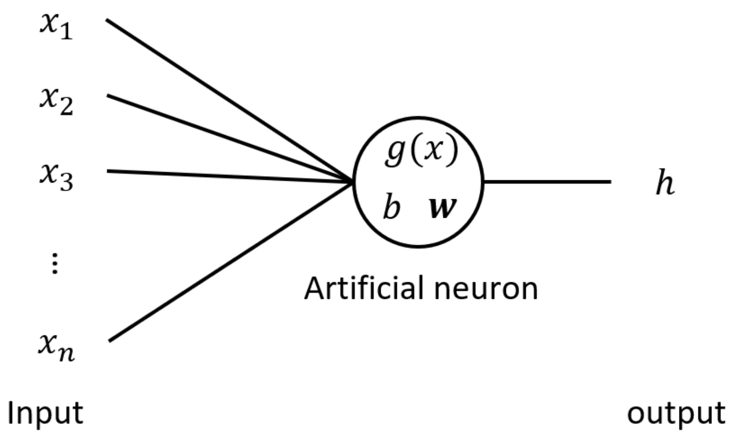


Figure 2.10: Schematic of an individual neural network node. The node receives several input values from the previous layer. Computations on these input values are done to generate an output. An activation function $g(x)$, weight vector **w** and bias b are used for these computations.

Figure 2.10 shows a schematic of an individual node, which is connected to many input nodes and produces a single output h. Each node contains a few attributes, which define how its computation is done. These attributes are a weight vector **w**, a bias b and an activation function $g(x)$. The output of the node is then computed as

$$h = g(\mathbf{w} \cdot \mathbf{x} + b), \tag{2.5}$$

where $\mathbf{w} \cdot \mathbf{x}$ represents the dot product of all input features x_i with the corresponding weights w_i. The activation function of a node is always the same in a single layer and often the same throughout most of the neural network. However, the bias and weights are defined specifically per node. These are also the values that are altered during the learning process to improve the model's output.

Activation Functions
There are many different types of activation functions that can be used, both linear and non-linear. In essence, the activation function decides what the output signal should look like based on a given input signal. Furthermore, activation functions allow for adding non-linearity to the model, which is one of the main advantages of neural networks. Four different types of activation functions are used in this report:

- The sigmoid function. This function is often used in binary classification problems, and it maps any input value to the (0,1) range. The function is defined as

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{2.6}$$

- The Rectified Linear Unit (ReLU). It is defined as

$$g(x) = max(0, x). \tag{2.7}$$

  It is quite a simple function that allows the model to "turn off" nodes by ensuring the inputs, weights, and bias combined give a negative value. When this happens, the ReLU always outputs a 0, meaning that no signal is sent further through the network.
- The leaky ReLU. This function is very similar to the ReLU and is defined as

$$g(x) = max(cx, x), \tag{2.8}$$

  where c is the slope coefficient, which is an arbitrarily chosen small value between 0 and 1. This function is the same as the ReLU for positive values but different for negative values. Instead of mapping all negative values to 0, there is a small slope in the function, and negative values are mapped to negative values closer to zero.
- The linear function. This is a simple linear layer, which is defined as

$$g(x) = \gamma x + \delta, \tag{2.9}$$

  where γ and δ are network parameters, which define the slope and bias of the function. These network parameters can also be updated during the learning process to improve performance.

Figure 2.11 shows a graph of all the different activation functions mentioned. As can be seen from the graph, the activation function chosen heavily affects the output signal of the node.

2.4.3. Convolutional Neural Networks

An important sub-type of neural networks are Convolutional Neural Networks (CNN). These are neural networks where convolutional layers are used to connect the nodes instead of fully connected layers. This means that no longer all nodes of layer j are connected to all nodes of layer $j+1$. This network type is often used to make predictions from multi-dimensional input data. For example, object recognition in images is an application where a CNN functions better than a conventional neural network. This is because the input data used here are images, which can be interpreted as two-dimensional arrays of pixel intensities. To recognise an object in an image, the combination of a group of pixels can say more about whether an object is present there than individual pixels. To problem with fully connected networks is that there is no information passed through the network about which pixels (or input nodes) are close to each other. Adding convolutional layers to the network solves this problem.

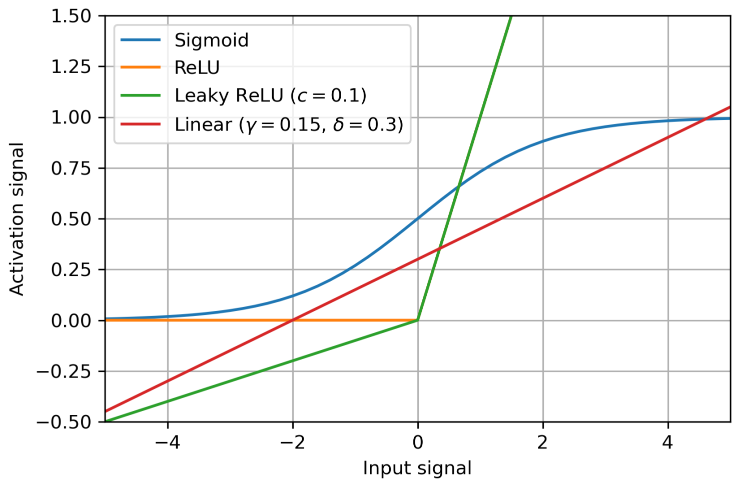


Figure 2.11: Graphical representations of the four different activation functions used in this thesis.

Convolutional Layer
Figure 2.12 depicts the mechanics behind a convolutional layer. A convolutional layer uses a kernel of arbitrary size and the same dimensionality as the input data. As the size of this kernel is smaller than the input data size, it is moved over the input data. In the figure can be seen how a three by three convolutional kernel starts at the top-left corner on the input data. The kernel values are then multiplied with the corresponding data values, and the results are summed together to obtain the output value in the top-left corner. Then, the kernel will be moved to the right, and again the same computation is performed. This is continued until an array of output values is obtained. The kernel values stay the same throughout this process, and these parameters are comparable to the weight parameters **w** in fully connected networks, which are optimised during the learning process. In the case of Figure 2.12, a single output element only depends on a patch of nine elements on the input array. This is the main difference with fully connected networks, where each output node depends on all input nodes. The entire computation can be summarised by the equation

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, i+n)K(m,n), \tag{2.10}$$

where S stands for the output array, I for the input array, K for the kernel array, and m and n are iterated over the kernel size [37].

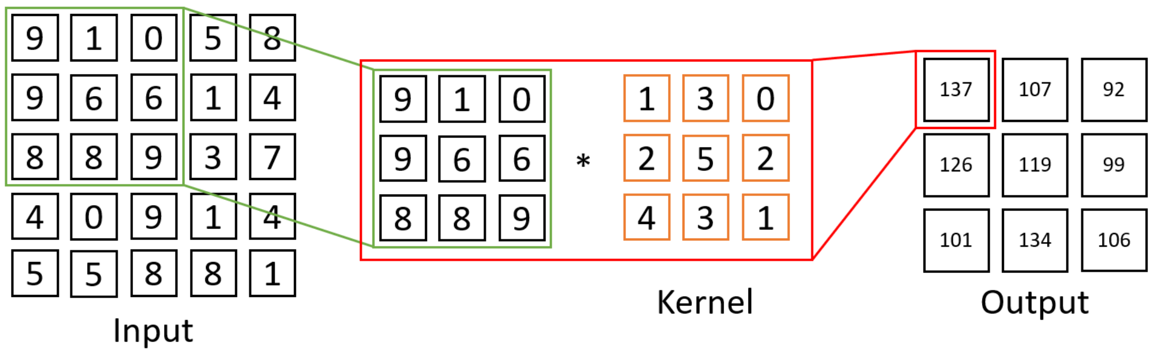


Figure 2.12: The computation done in the convolutional layer. A kernel is moved over the input data, generating output values based on the linear combination of input values with kernel values.

A few things have to be determined when implementing a convolutional layer. First, the kernel size needs to be specified. In Figure 2.12, a three by three kernel is used, which means that patches of nine input elements are mapped to single values in the output. By choosing a different kernel size, larger or smaller patches can be used.

Furthermore, the amount of input and output channels needs to be specified. Each input channel needs to be connected to each output channel, which means that if there are x input channels and y output channels, there should be y kernels with a size of $y \times k \times k$ to map the input to the output. k stands for the kernel size here.

Also, the stride used should be determined. By default, the stride is often set to one, which means that after each computation, the kernel moves by one element. If this value is increased, larger steps can be taken between calculations. A side effect of increasing the stride is that the output size is reduced.

Lastly, the amount of padding done has to be specified. Padding is the process of adding values around the input data, increasing its size. The most common type of padding is zero-padding, where zeros are added around the input data. As seen in Figure 2.12, the convolutional layer operation causes a decrease in data size. If the data is zero-padded, this can be prevented. The specification of stride, kernel size, and zero-padding determine the size of the output with respect to the size of the input as

$$d_{out} = \frac{d_{in} + 2p - k}{s} + 1, \tag{2.11}$$

where d_{out} stands for the output size, d_{in} for the input size, p for the number of zero-padding layers applied, k for the kernel size and s for the stride.

Transpose Convolutional Layer
Apart from regular convolutional layers, transpose convolutional layers are also often used in CNNs. This is because, in general, convolutional layers downsample the data, and in some cases, it is necessary to upsample the data again. In essence, a transpose convolutional layer does precisely the opposite to what a convolutional layer does, as is shown in Figure 2.13. An input value is multiplied with all different values present in the kernel, which creates a patch of output data from the single input value. This is repeated for every input value. If the patches overlap (which is the case when a stride smaller than the kernel size is used), the overlapping outputs are simply added together.

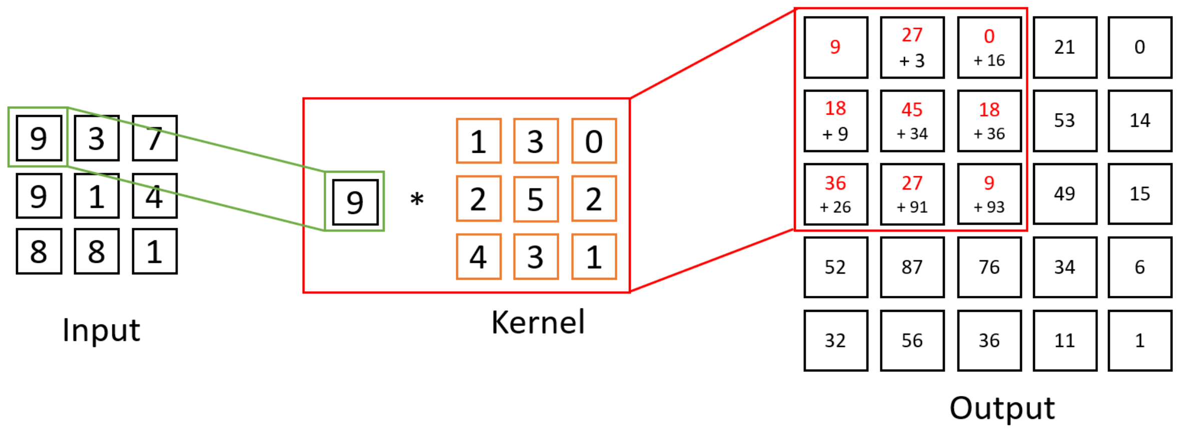


Figure 2.13: The computation done in a transpose convolutional layer. Each input value is multiplied by the different kernel values generating multiple patches. Overlapping parts of the patches are added together in the output. The red numbers indicate the contribution of the selected input element to the designated patch.

Pooling Layer
Another commonly used layer in CNNs is the pooling layer, which downsamples the input data. The most common pooling layer is the max-pooling layer, of which an example is shown in Figure 2.14. To generate the output, a kernel is moved over the input data, and the maximum value of the selected patch is selected as the output value. Again, the stride, kernel size and zero-padding can be adjusted to obtain the desired output dimensions.

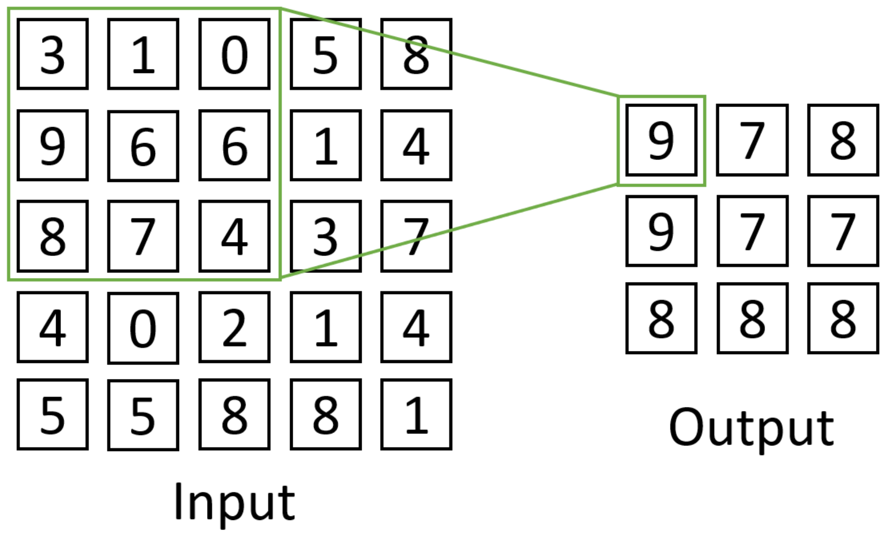


Figure 2.14: The computation done in a max-pooling layer. A kernel is moved over different input patches, and the maximum value within the patch is selected as output.

Normalisation Layers
Lastly, a common type of layer in a CNN is the normalisation layer. As the name suggests, normalisation layers normalise the input data. These layers are often combined with convolutional layers to create a normalised output. The normalisation eases the optimisation and helps very deep networks to converge correctly [38]. This is because applying normalisation to the different layers ensures that the transmitted signals between layers stays within a certain order of magnitude, which means that the calculated gradients will be smoother, allowing for better training.

The normalisation can be done in different ways, but in this research, group normalisation is mainly used. With group normalisation, the data that is present in the different channels of the input data is grouped into smaller sets of channels. For example, if the input data consists of 40 channels, then four groups of data can be made, each consisting of 10 channels. For each group, the normalisation is then performed as

$$\widehat{x_i} = \frac{1}{\sigma}\left(x_i - \psi\right), \tag{2.12}$$

where $\widehat{x_i}$ is the normalized output, x_i is the input of a specific element and σ and ψ are the standard deviation and average, respectively, of the all data within a certain group of input channels.

2.4.4. The Loss Function

A vital part of any machine learning application is the algorithm's training. In the case of supervised learning, the general idea is that the adjustable parameters of the algorithm are optimised based on the differences between the prediction and the ground truth. A loss function is used to assess this difference, quantifying how close or far off the prediction is from where it should be. This loss can then be minimised in an optimisation algorithm to get the predictions as close as possible to where they should be. Therefore, choosing the correct loss function is essential for proper model training.

An example of a simple loss function that works in many cases is the Mean Squared Error (MSE), which is defined as

$$L_{MSE}(y, \widehat{y}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{y}_i\right)^2, \tag{2.13}$$

where L_{MSE} stands for the MSE loss, y for the ground truth output, $\widehat{y}$ for the prediction, y_i for the i^{th} element of the ground truth, $\widehat{y}_i$ for the i^{th} element of the prediction and N for the number of elements in the output.

Another commonly used loss function is the Binary Cross-Entropy (BCE) loss. This loss is most frequently used in combination with a sigmoid activation function in the final layer of the neural network. This combination is useful when the output of the model consists of one or more binary values. The loss is defined as

$$L_{BCE}(y, \widehat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\widehat{y}_i) + (1 - y_i) \cdot \log(1 - \widehat{y}_i). \tag{2.14}$$

As the sigmoid ensures the prediction is between zero and one (but never reaches those values), the two logarithms in Equation 2.14 always have some value less than zero. The loss is closest to zero when the prediction and the ground truth are both either one or zero [37].

2.4.5. Back-propagation

The main power of neural networks lies in the fact that through training, the loss function can be minimised, and the prediction can be moved as close as possible to the ground truth values. This is done by updating the parameters of the deep learning model. To optimise the parameters, an optimisation algorithm has to be implemented during the training. For an optimisation algorithm to function, the gradient of the loss function has to be known with respect to the optimisable parameters, as this allows for the optimisation algorithm to know in which direction to "move" the parameters to reduce the loss value. As most CNNs consist of many sequential layers, each with its own parameters to optimise, this is not as straightforward as it might seem. The method used for neural networks to obtain these gradients is called back-propagation.

The mechanics behind backpropagation are summarised in Figure 2.15 for a simple, fully connected network example. Here the input data consists of three input nodes, and the information is propagated forward through the network. At each layer, the operations performed can be summarised by a single function. For example, in the first hidden layer, the output values y_i are determined by the vector function $\mathbf{f}(\mathbf{x})$, which contains the linear operations done by the weights and biases, and the activation functions for all nodes in that layer. Eventually, the output **k** is obtained, and a loss L is computed after comparison of **k** with the ground truth **gt**.

To compute the gradients of all intermediate variables with respect to this loss, the computation is started at the output layer. To compute the gradient of the loss with respect to the output nodes, the derivative of the $Loss$ function can simply be used to obtain $\nabla_{\mathbf{k}}L$. To compute gradients further back in the network, the backpropagation algorithm uses the chain rule. In one dimension, this rule states that if $y = f(x)$ and $z = g(y)$, z can be directly computed as $z = g(f(x))$ and the derivative of z with respect to x can be computed as

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}. \tag{2.15}$$

When extending the chain rule to multiple dimensions, it can be written as

$$\nabla_{\mathbf{x}} z = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^T \cdot \nabla_{\mathbf{y}} z, \tag{2.16}$$

where ∇ stands for the gradient operator and $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ denotes the Jacobian matrix of the function mapping **x** to **y**. Applying the chain rule to the example network in Figure 2.15 and the gradient can be backpropagated all the way to the input variables as

$$\nabla_{\mathbf{x}} L = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^T \cdot \left(\left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^T \cdot \left(\left(\frac{\partial \mathbf{k}}{\partial \mathbf{z}}\right)^T \cdot \nabla_{\mathbf{k}} L\right)\right). \tag{2.17}$$

The only things required for this gradient are the gradients of the output of each layer with respect to the input of each layer. Back-propagating a layer further back then simply means computing the gradient of the output of that layer with respect to the input and multiplying that with the already calculated gradient in the previous layer [37].

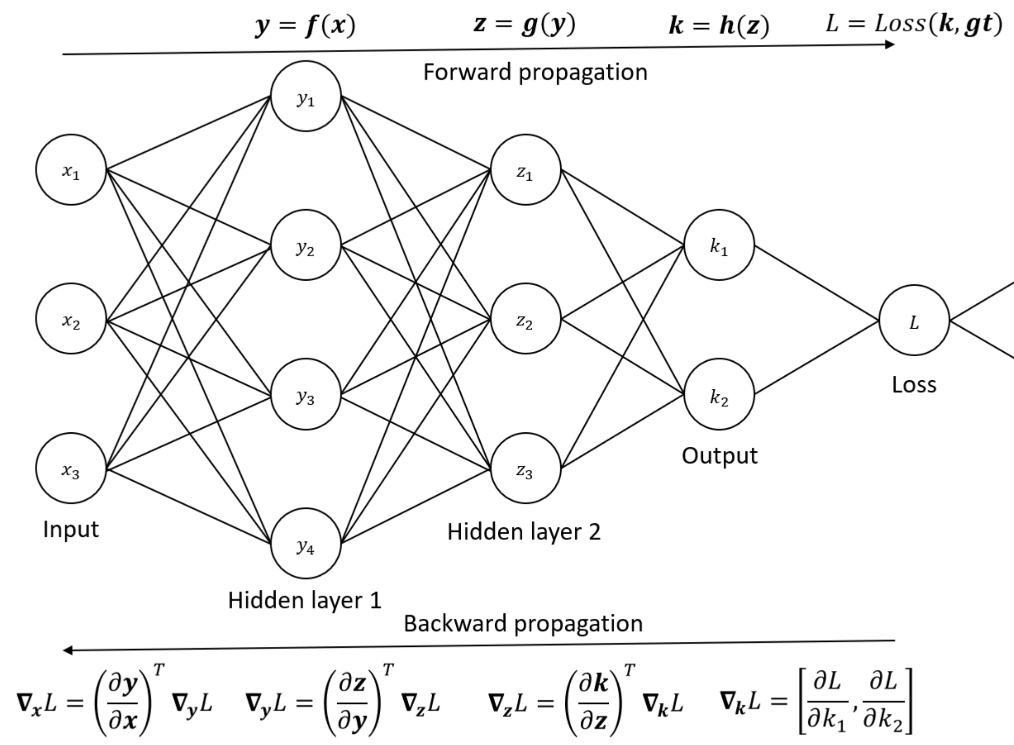

$y = f(x)$
$z = g(y)$
$k = h(z)$
$L = Loss(k, gt)$
Forward propagation
x_1
x_2
x_3
Input
y_1
y_2
y_3
y_4
Hidden layer 1
z_1
z_2
z_3
Hidden layer 2
k_1
k_2
Output
L
Loss
gt_1
gt_2
Ground truth
Backward propagation
$\nabla_x L = \left(\frac{\partial y}{\partial x}\right)^T \nabla_y L$
$\nabla_y L = \left(\frac{\partial z}{\partial y}\right)^T \nabla_z L$
$\nabla_z L = \left(\frac{\partial k}{\partial z}\right)^T \nabla_k L$
$\nabla_k L = \left[\frac{\partial L}{\partial k_1}, \frac{\partial L}{\partial k_2}\right]$

Figure 2.15: Overview of the mechanics behind back-propagation. First, input data is propagated forward through the network, where sequential functions apply operations on the data until a final single loss value is computed. The gradients of the loss with respect to all (intermediate) variables are then calculated by using the chain rule and computing gradients of the output of a specific layer with respect to its input.

With the method described above, gradients can be calculated and stored for each intermediate variable in the network. The last step still required is to translate this gradient to the gradient on the weights and biases of the nodes. This is done similar to back-propagation by calculating the gradient of the output of a node with respect to a specific parameter and multiplying that with the previously computed gradients.

2.4.6. Optimisation Algorithms

After the gradients have been computed, the last step is to optimise the neural network parameters based on the calculated loss. There are many different algorithms available for the optimisation, but in this subsection, two algorithms are discussed: gradient descent and Adam optimisation.

Gradient Descent

Gradient descent is a robust, intuitive and easy-to-implement algorithm used to find the local minimum or maximum of a function. In the case of deep learning, it can be used to find the minimum in the loss as a function of the neural network parameters. The algorithm works as its name suggests: it uses the gradient of the loss function and moves in the direction in which this function is descending.

This is easy to interpret in one dimension: consider the function $f(x)$, which must be minimised based on the single parameter x. First, an initial value x_0 is taken and the function value $f(x_0)$ can be computed. Also, the derivative at this point $f'(x_0)$ is computed, and the next position on the x-axis is determined according to Equation 2.18.

$$x_{n+1} = x_n - \epsilon \frac{df}{dx}|_{x=x_n}. \tag{2.18}$$

In this equation, ϵ denotes the learning rate and regulates step size taken when updating the parameters. The descent is repeated several times until the function reaches its minimum, which means that the gradient value starts converging towards zero, and smaller and smaller steps are taken until no significant improvements are made.

In the case of neural networks, gradient descent is applied in a multi-dimensional space as there are many optimisable parameters, each with its own derivative with respect to the loss (as calculated with the back-propagation algorithm). In this case, Equation 2.18 is extended to multiple dimensions as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \epsilon \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}_n), \tag{2.19}$$

where $\mathbf{x}_n$ denotes a vector of all optimisable parameters after the n^{th} optimisation iteration and $\mathbf{f}(\mathbf{x})$ the function containing all operations performed in the forward propagation of the neural network.

After each iteration, the parameters (weights and biases) of the neural network will have moved closer to a minimum, with a lower loss function as a result. To do a new round of optimisation, first, the data needs to be repropagated forward through the model to compute the new loss function and again back-propagated to obtain the new gradients. The repeating process of forward propagation, back-propagation, and parameter optimisation is what is done during the training of a network.

Adam Optimisation
The Adam optimiser is a more advanced optimisation algorithm used in this thesis to optimise the neural networks. This algorithm is an extension of the gradient descent optimisation, where the learning rate is adjusted during the training, specifically for each parameter. This allows for faster and more precise optimisation by using larger learning rates when the parameters are still far away from the minimum and smaller learning rates when they are closer. Since the learning rate is adjusted per parameter, some parameters can be kept constant when their optimal value has already been reached, while other parameters are still descending quickly to their optimal value. This is also why the Adam optimiser works well when many parameters are involved.

The parameters are updated as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \cdot \frac{\widehat{\mathbf{m}}_{n+1}}{\sqrt{\widehat{\mathbf{v}}_{n+1} + \tau}}, \tag{2.20}$$

where α is defined as the stepsize, $\widehat{\mathbf{m}}_n$ as the bias-corrected first moment vector in iteration n, $\widehat{\mathbf{v}}_n$ as the bias-corrected second moment vector and τ a numerical stability term often set to 10^{-8}. The bias-corrected first moment vector $\widehat{\mathbf{m}}_n$ is updated each iteration and is computed as

$$\widehat{\mathbf{m}}_n = \frac{\mathbf{m}_n}{1 - \beta_1^n}, \tag{2.21}$$

where β_1 denotes the first exponential decay rate for the moment estimates, which is typically set to 0.9 and $\mathbf{m}_n$ denotes the first moment vector, which is also recomputed each iteration as

$$\mathbf{m}_{n+1} = \beta_1 \cdot \mathbf{m}_n + (1 - \beta_1) \cdot \nabla_{\mathbf{x}_n} \mathbf{f}(\mathbf{x}_n). \tag{2.22}$$

Similarly, the bias-corrected second-moment vector is computed each iteration as

$$\widehat{\mathbf{v}}_n = \frac{\mathbf{v}_n}{1 - \beta_2^n}, \tag{2.23}$$

where β_2 denotes the second exponential decay rate, which is typically set to 0.999, and $\mathbf{v}_n$ the second moment vector, computed as [39]

$$\mathbf{v}_{n+1} = \beta_2 \cdot \mathbf{v}_n + (1 - \beta_2) \cdot (\nabla_{\mathbf{x}_n} \mathbf{f}(\mathbf{x}_n))^2. \tag{2.24}$$

2.5. Dose Prediction with Deep Learning

To predict dose distribution with a deep learning algorithm, information on the patient's anatomy must be used as input to the model. This is typically done by inserting a delineated CT into the deep learning algorithm. This kind of input data can be considered a set of multiple 3D images. The output of the deep learning model, a dose distribution, is then also a 3D image, with the voxel intensities defining the deposited dose in Gy. Therefore, commonly used deep learning architectures for such a task are 3D CNNs. For 3D CNNs, convolutional layers work very similar as defined in Subsection 2.4.3. The main difference is that a 3D kernel is used, which is moved over the 3D image in all three directions.

To train a dose prediction neural network, the predicted dose distribution is compared to a ground truth dose. This ground truth can be obtained using a dose calculation algorithm (see Subsection 2.2.3) and a manually or automatically generated treatment plan. By using a large set of patients, for which the delineated CT data and the delivered dose distribution is known, a CNN can be trained to predict accurate dose distributions.

As can be imagined, predicting dose distributions is a very complex task, even for a deep learning model. The important information in the delineated CT data needs to be extracted and based on the extracted features, an entire 3D image needs to be generated as prediction. Therefore, the neural network architecture used for such a task consists of many hidden layers with many optimisable parameters.

A breakthrough in the field of dose prediction but also in image segmentation occurred when Ronnenberger et al. [5] developed the "U-net". This is a CNN architecture, initially developed for the segmentation of 2D images into patches, which has proven capable of predicting dose distributions [6] when extending the model to three dimensions. Figure 2.16 shows a schematic overview of the architecture of the U-net for the two-dimensional application.

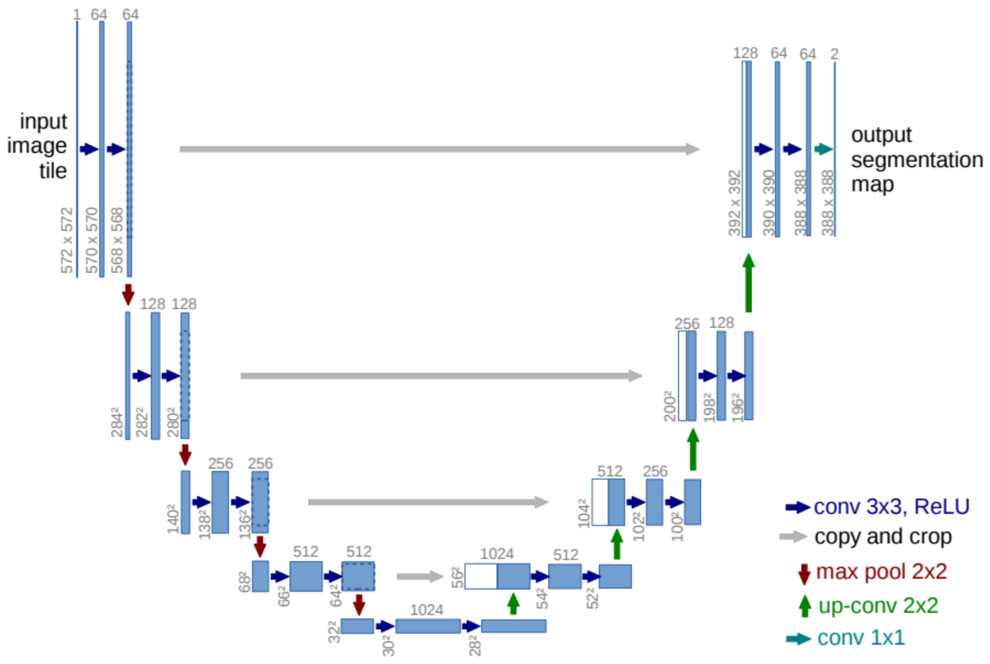


Figure 2.16: Schematic overview of the U-net developed by Ronneberger et al. [5]. The rectangles represent the information propagating through the network and the arrows represent that different operations performed on the data. In essence, the input image is downsampled to extract the important features necessary to obtain a segmentation map. Then the features are upsampled into an output image, which presents the segmented regions.

The rectangles in Figure 2.16 represent the information propagating through the model. The height of the rectangle represents the size of the 2D data at some point in the model. The rectangle's width represents the number of channels containing the information at a certain point in the forward propagation. In Figure 2.16, it can be seen that a single input image with a size of 572 by 572 pixels is used, and that the prediction contains two images with a size of 388 by 388 pixels. The arrows denote the operations performed on the data.

As can be seen, the name of the U-net comes from the shape of the architecture; the “depth” in the schematic represents how far the data has been downsampled. The architecture of the U-net can be broken up into two main parts. In the first half, the input data is downsampled through max-pooling layers, while convolutional layers with a three by three kernel are used to extract meaningful information from the individual patches of the input data. By utilising a zero-padding of one and a stride of one, the output of these convolutional layers can be kept equal to the input (see Equation 2.11). After each max pooling layer, the data is downsampled further, and features on a more abstract level can be extracted. The amount of channels is also increased each time to allow for more space to store the extracted features of the input image. This first half of the U-net is often called the encoding arm, as the input data is encoded into features deemed important by the neural network.

The second half of the U-net, called the decoding arm, is used for upsampling the data to the preferred size. In this part, regular and transpose convolutional layers combine and translate the extracted features into a dose distribution specifically for the patient given as input. To retain high-resolution information on the patient, skip connections are used (denoted by the grey arrows in Figure 2.16) to copy the high-resolution data from the encoding arm to the decoding arm.

Since the development of the U-net, alternative or slightly modified architectures have been developed with varying results. Recently, research has shown that the performance of the commonly used U-net can be improved for head and neck cancer patients by combining it with another widely used architecture, called the DenseNet [40], and form the Hierarchically Densely Connected U-net (HD U-net) [6]. In the HD U-net, the regular convolutional layers are replaced by dense convolutional layers. It has been shown that the HD U-net can outperform the U-net for lung and head-and-neck cancer patients [6],[41]. See Subsection 3.3.1 for more details on the mechanics behind the dense convolutional layer.

2.6. Physics-Guided Neural Networks

The downside of applying a deep learning model to a certain task is that there is no control over how a particular prediction is created. For the case of predicting dose distributions, this problem manifests itself in the fact that a predicted dose distribution can be very close to the ground truth without being very realistic. For example, for a dose distribution to be realistic, high dose regions in the shape of beams should be visible at entry points, from which a lot of dose is delivered. There is no guarantee that these details are being taken into account.

To solve this issue, a Physics-Guided Neural Network (PGNN) can be used. A PGNN is a neural network where additional constraints are implemented to force physical behaviour by the neural networks. In general, there are two ways this can be implemented. The most common method is by adding a secondary loss function to the training of the neural network. This secondary loss is then used to penalise nonphysical behaviour of the neural network. By combining a standard loss with this physical loss, the neural network will learn that nonphysical predictions are not preferred, and it will omit these to minimise the physical loss. The physical loss can be applied on the output of the deep learning model but also on intermediate variables if desired.

A second method of implementing physics guidance is by creating a Hybrid-Physics-Data (HPD) model. In this case, two models are combined, where one produces an output based on deep learning and the other an output based on physical laws. The physical output might only be an approximation and not very accurate, but it does follow the correct physical behaviour. By adding the physical output to the deep learning model as an extra input channel, an example is already given for the model to base its prediction on. PGNNs have not yet been used in research on dose prediction, but there are already many different applications where successful implementation of PGNNs has been achieved[42], [43], [44], [45].

3 Method & Materials

In this chapter, the method is discussed that is used to obtain the results of this study. First, an overview is given of the input data that has been used for the different models. Next, in Section 3.2, the framework used for the deep learning models is quickly introduced. After that, Section 3.3 until Section 3.7 discuss the various models developed and implemented throughout this project.

3.1. Input Data

For deep learning models to function, data is required to train the neural networks. As the goal of this project is to more accurately predict dose distributions and treatment plan parameters, ground truth data of these dose distributions and treatment plans is needed to allow the model to learn what a correct dose distribution and treatment plan looks like. Furthermore, the models require information on the patient's anatomy as input to base their prediction on.

3.1.1. Patient and Plan Data

Two different datasets of prostate cancer patients have been used throughout this project, both provided by the Netherlands Cancer Institute (NKI): Dataset I contains 89 patients and is a very homogeneous set of patients. Dataset II consists of 114 patients and is less homogeneous than dataset I. The difference in homogeneity comes from the fact that Dataset I consists of more carefully selected patients who have similar tumours and anatomy. Furthermore, treatment plans have been generated with the Pinnacle auto-planning system, without any manual adjustments.

Treatment Protocol

The delineations of the CT are made using the Mirada auto-contouring software, where contours of the body volume, rectum, rectal wall and anal sphincter are automatically generated. The rectum is then adjusted if necessary, and the CTV is manually delineated according to the defined protocol. Irregularities within the patient, such as a low-positioned bowel loop, are also manually delineated. These delineations are made by defining several contour points, which make up the edge of a particular structure. After the delineation is completed, the structures are sent to the Pinnacle auto-planning system, where the VMAT table is added to the structures as well.

After generation of the plan in Pinnacle, a few things are manually checked:

- If the VMAT table structure is correctly placed.
- If structures are correctly defined.
- If the isocentre is placed at the correct location.
- The generated treatment plan.

If the plan is not satisfactory, the treatment is replanned, possibly with different objective functions and with the use of a warm start (starting from an already decent plan). If still the plan is not deemed satisfactory, manual planning will be required.

The treatment is delivered with an Elekta machine using an 80 leaf MLC with a photon beam of 10 MV for dataset I and 6 MV for dataset II. MV denotes the maximum photon energy in MeV present in the spectrum of the beam. The prescribed dose is 60 Gy in 20 fractions of 3 Gy, and the dose is delivered over two rotation arcs, which move between a gantry angle of 140° and 220° in both the clockwise and counter-clockwise direction, as shown in Figure 3.1. The plan is defined on 71 control points per rotation arc, which have an equidistance spacing of 4°.

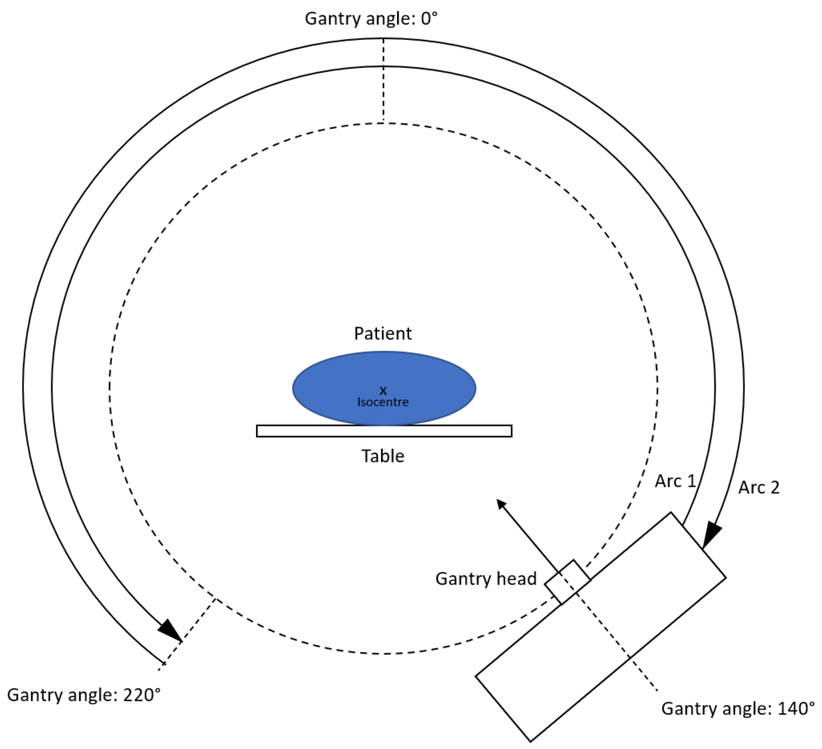


Figure 3.1: Schematic overview of how the VMAT treatment is delivered to the patients from the datasets. Two arcs are used, moving in opposite directions over the same gantry angles.

DICOM data

The data is stored in DICOM format, which stands for Digital Imaging and Communications in Medicine. It is the internationally standardised format for transferring and storing medical images and related patient data [46]. Four different types of DICOM files have been provided for each patient: CT Image, RT Structure set, RT Plan and RT Dose. Note that in dataset I, the CT data is not available.

- CT Image consists of multiple files, where each contains the raw CT data of an individual slice of the CT scan. The image size of all CT images is 512 by 512 pixels, and the physical distance between two slices is 3 mm. The physical representation of 1 pixel inside the patient varies slightly for each patient but stays close to a 1 by 1 mm square. Also, the number of CT slices is different for each patient, mainly due to the difference in size between patients.
- RT Structure contains the delineated structures visible on the CT that are important for the treatment. Six structures are important for this study: the PTV, rectum, rectal wall, anal sphincter, body volume and the treatment table. The treatment table is only used for a dose calculation algorithm and is not included in any deep learning input.
- RT Plan contains all necessary information for executing a treatment plan. Important machine parameters found inside this DICOM file are the leaf positions of the MLC and the beam intensity of the photon beam during treatment. Other important information like isocentre position, gantry angle and Source-to-Surface Distance (SSD) can be found in this file as well.
- RT Dose contains all information on the delivered dose. The delivered dose is defined on a three-dimensional voxel grid, where each voxel represents a 4 by 4 by 4 mm cube inside the patient. The dose delivered to each voxel is given in Gray (Gy). The image size of the dose data is different for each patient.

To ensure a good performance of the deep learning models, the datasets must not contain any outliers. Therefore the structure sets and CT images of all patients were revised to see if anything odd could be found. A previous master student, Meerbothe, has already done this for dataset I in earlier work [7]. In dataset II, five patients were found with a hip implant, which have been removed. This means that ultimately, dataset II consists of 109 patients.

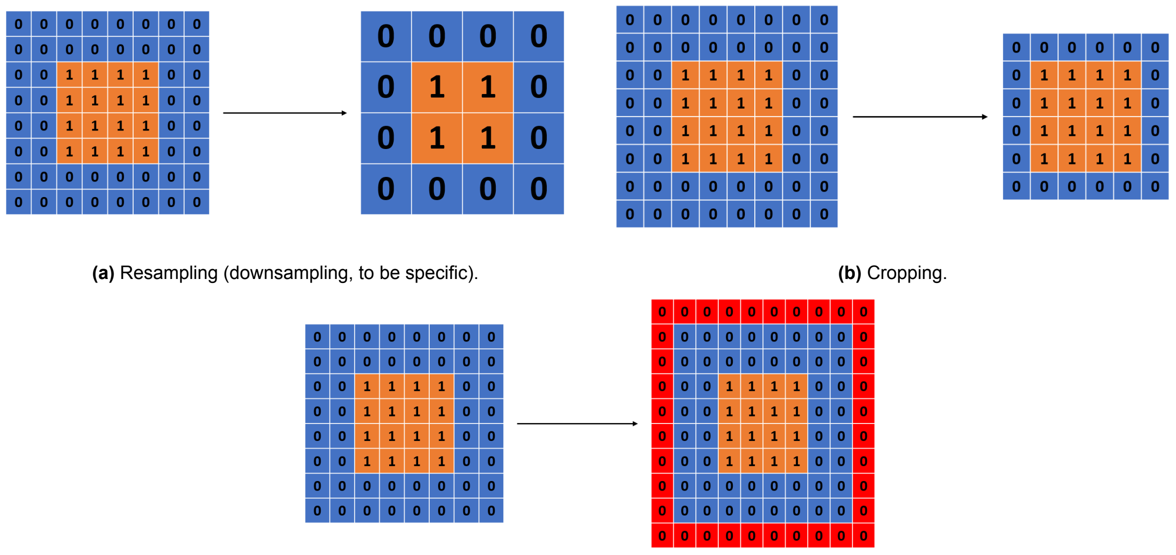
3.1.2. Data Pre-processing

To work with the given input data, all data has been loaded and stored into Numpy arrays [47]. This is because Numpy arrays can be loaded from storage much quicker than the DICOM data, and they are much more flexible to work with. This allows for quick real-time loading and manipulation of data whenever necessary. The only disadvantage is that these arrays take up more storage as their format is not as compressed as the DICOM format.

To convert the DICOM data into Numpy data, two different modules have been used: Pydicom and Pyavsrad. Pydicom is an open-source Python module that can read DICOM data in Python scripts [48]. This has mainly been used to read specific settings from the RT Plan DICOM data like isocentre position and MLC leaf positions. Pyavsrad is an in-house Python module from the NKI, which has been used to extract dose, CT and structure data and store it into 3D Numpy arrays. For each of the delineated structures used for deep learning, a 3D Boolean mask is burned onto the dose grid, where ones represent a voxel inside the structure and zeros a voxel outside the structure.

Data Resampling, Cropping and Zero-padding

For the proper functioning of the deep learning models, it is essential that the input and output data (CT, structures and dose) are correctly aligned and cropped. This means that a specific voxel in all three arrays represents the same volume in physical space. Therefore, also the image size of the 3D arrays should be the same. This can be done through resampling, cropping and zero-padding the dose, structure and CT data arrays. Figure 3.2 shows a representation of these three operations on a simple 2D array.



(a) Resampling (downsampling, to be specific).

(b) Cropping.

(c) Zero-padding.

Figure 3.2: A visualisation of the three operations used for data pre-processing on a simple 2D array of data.

As the structure masks have been "burned" onto the dose array, the dose and structure arrays already have the same resolution and are correctly aligned. The issue is that the CT Image data is made on an entirely different grid. To solve this problem, the CT Image data is translated onto the dose grid. The first step in this process is resampling the CT data to ensure that the CT voxels also represent a 4 by 4 by 4 mm volume in the patient. This is done using the Scikit-image module: an image processing tool in Python [49]. Specifically, the function *transform.rescale* is used to increase the voxel size to 4 by 4 by 4 mm. Because the initial voxel size in the axial plane is different for each patient, the number of voxels in the resulting 3D image will be different for each patient after the rescaling.

After downsampling, the CT voxels are equal in size to the dose voxels, and the next step is to align the matching voxels by cropping and zero-padding the CT array. This is done using the image position defined in both the RT Dose and the CT Image DICOM files. The image position gives the coordinates of the centre of the first voxel in the array in the patient-based coordinate system (see Figure 3.3). The difference between these coordinates can be used to find the amount of misaligned data and overlapping data (see Figure 3.4). The misaligned data in the CT array is then cropped out of the array, while the misaligned data in the dose array is zero-padded into the CT array. The new CT array is then correctly aligned to the dose and structure data. In practice, the CT array is generally much more extensive than the dose array, meaning that most of the pre-processing consists of cropping out CT data.

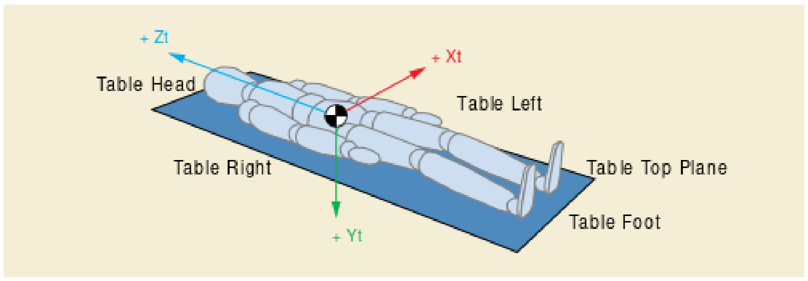


Figure 3.3: Definition of the patient-based coordinate system [50]. The x-coordinate increases when moving from the patient's right arm to the left arm. The y-coordinate increases when moving from the patient's front to the patient's back. The z-coordinate increases when moving from the patient's feet to the head.

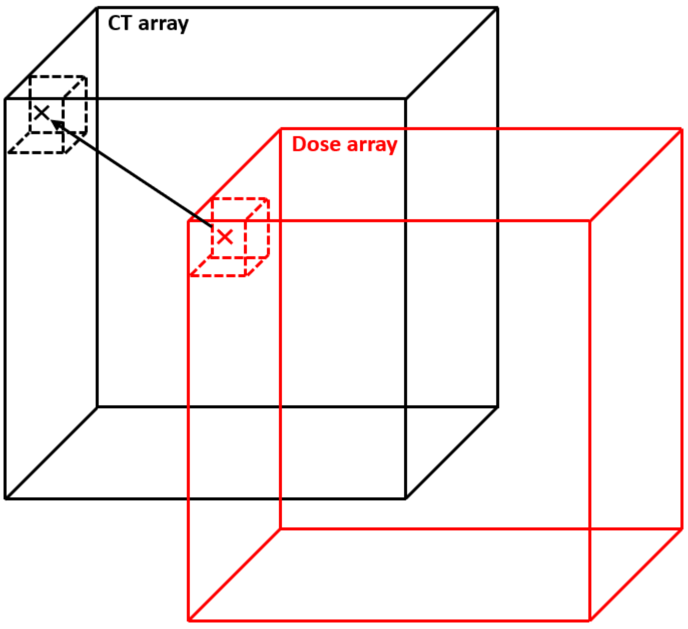


Figure 3.4: A visualisation of the misaligned CT array and the dose array. The black cube represents the CT data, and the red cube the dose data. The dashed cubes represent the first voxels of these arrays. The patient image position for each of the two arrays is then defined as the coordinates of the two crosses located at the centres of the first voxel. The black arrow represents the difference between these coordinates.

The last step is cropping and zero-padding the data into a uniform shape for all patients. This is necessary as the input data size must be consistent for neural networks to function. To this end, a standard size of 144 x 96 x 64 voxels is defined in the x, y and z direction of the patient-based coordinate system, respectively. This size is chosen as it allows for minimal data loss while also being able to halve the resolution multiple times in each dimension, which is important for the data downsampling taking place in the deep learning models. For each patient, the data is again cropped and zero-padded until the standard size is obtained. Figures 3.5 and 3.6 show the resulting pre=processed input and ground truth data that can be used for the deep learning models.

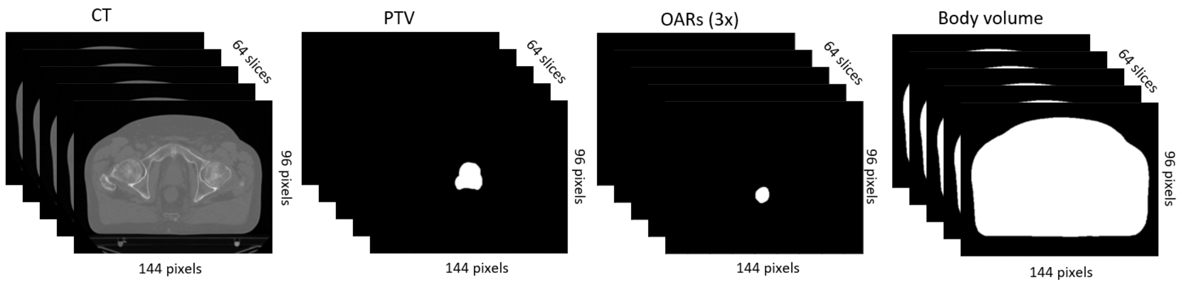


Figure 3.5: Example of the pre-processed input data. It consists of six 3D arrays. From left to right, these are one CT array, one PTV structure mask, three OAR structure masks (rectum, rectal wall and anal sphincter) and one body volume structure mask.

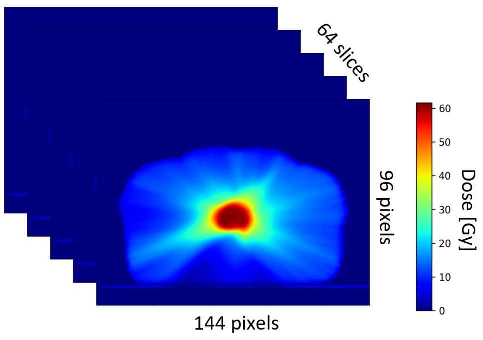


Figure 3.6: Example of pre-processed ground truth dose data. It consists of one 3D array that is aligned to the input data. The colour gradient represents the dose deposition value in Gy.

3.1.3. Data Augmentation

Using data augmentations means that, effectively, a more extensive dataset is available for the training of deep learning models. It helps to prevent overfitting and keep the model more generalised [51]. However, not every type of data augmentation is realistic. It is important to keep this in mind as unrealistic input and ground truth data will also lead to incorrect predictions. To ensure that no unrealistic data is created, the only augmentations used in this project are rigid translations and small rotations. This does not change the patient's anatomy, only the positioning of the patient with respect to the CT scanner or the LINAC.

To create the augmented data, a list of 87 different rotations and translations is made, defining the degree of rotation and the rotation axis, or the length of translation and the translation direction. This list consists of 75 translations, with different combinations of sagittal, axial and coronal translations between the zero and three voxels. The remaining 12 augmentations are small rotations, where a rotation on one of the three main axes is performed with either one or two degrees in clockwise or counter-clockwise direction. The translations are performed through cropping and zero-padding part of the edges, effectively translating the structures of interest. The rotations are performed using the *rotate* function from the Scipy module [52].

During training, a random selection can be made of which augmentation to use. The augmentated structures can the be created during the training as this does not significantly impact the training speed and allows for randomising the used augmentations. The data augmentation scripts used for augmenting the data have been created by Meerbothe [7].

3.2. PyTorch

The programming language in which the deep learning models have been implemented is Python [53]. Specifically, the module PyTorch [54] has been used for the creation and training of neural networks, which is an open-source machine learning framework that allows the use of a Graphical Processing Unit to speed up the training process.

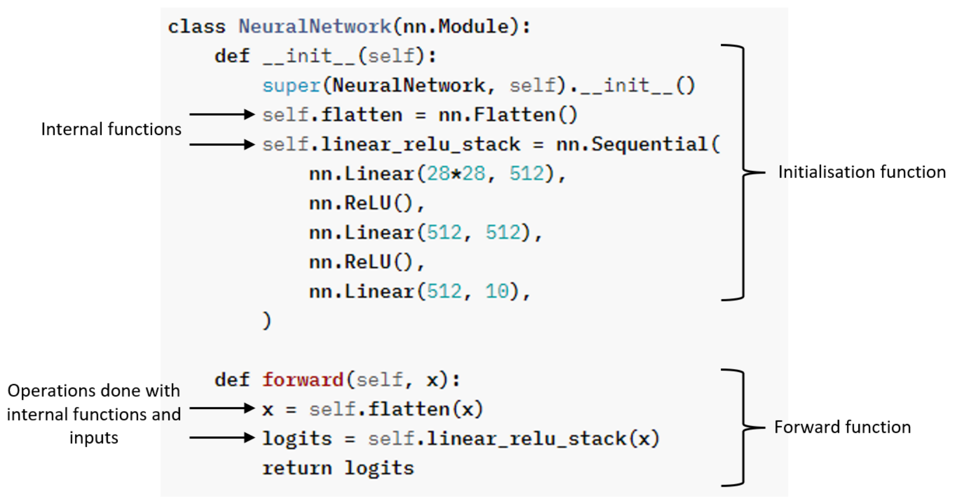
3.2.1. PyTorch Tensors

The PyTorch Tensor is the primary variable type used for holding data and model weights. The Tensor is very similar to the Numpy arrays, as they can store multidimensional matrices in any desired shape, and its elements can be represented with different data types (double, float, integer, etc.). The PyTorch module also contains many functionalities allowing for manipulating the Tensor datatype. Most of the manipulations possible with Numpy arrays are also possible with Tensors. In addition, there are also easy-to-use bridging functions available in PyTorch, allowing for quick casting to and from Numpy arrays.

For neural networks to be trained, backpropagation needs to be possible through the network. This is why Tensors can also have two additional attributes: the *.grad_fn* attribute and the *.grad* attribute. The *.grad_fn* attribute holds the function that can be used to compute the gradient of the Tensor with respect to the variables on which it depends. The *.grad* attribute can store the gradient of a specific final value with respect to the Tensor where this gradient is stored. These two attributes allow for performing backpropagation through a deep learning model.

3.2.2. Building a Neural Network in PyTorch

Neural networks are made with PyTorch as a class object. A neural network class needs an initialisation function and a forward function. The initialisation function defines what internal functions are created within the neural network class on the initialisation of the neural network. This function effectively builds the neural network, including layers and weights. The forward function then decides what happens during a feedforward of input data through the network. This function uses the initialised layers and parameters to calculate an output to the neural network. Figure 3.7 shows a simple example code of a neural network built in PyTorch.



```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

Figure 3.7: Example of how a neural network is built as a class object in PyTorch [55]. In this example, two internal functions are created: a flatten function, which flattens the input tensor and a function containing a few sequential linear layers and activation functions. In the forward, these functions are used to process the input data.

3.2.3. The Computational Graph and Back-propagation

In the PyTorch framework, the gradient calculation is done using the computational graph. The computational graph can be visually interpreted as a network of operations and variables, connected through the dependencies of the variables on each other. In essence, the computational graph stores all the operations performed on the input variables. At the same time, it defines and stores the backward functions of each operation included in the computational graph. These backwards functions are the derivative functions of the operation's output with respect to the input.

These operations and backward functions make up the computational graph, and Figure 3.8 shows an example visualisation of such a graph. The idea is that once the input data has been fed forward through all the required operations, the computational graph has stored all the operations done and the corresponding backward functions, which can be used together with the chain rule to compute the gradient of the output with respect to the input variables (see Subsection 2.4.5). This gradient calculation is then done during the backpropagation through the model, and the calculated gradients are then stored in the *.grad* attribute of the model parameter tensors. Lastly, these gradients can be used in an optimisation algorithm to update the model parameters.

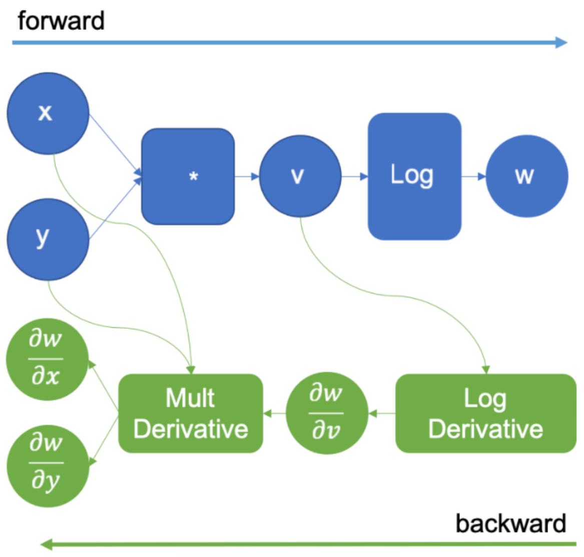

forward
x
y
*
v
Log
w
$\frac{\partial w}{\partial x}$
$\frac{\partial w}{\partial y}$
Mult Derivative
$\frac{\partial w}{\partial v}$
Log Derivative
backward

Figure 3.8: Example of a simple computational graph [56]. In this example, the two input variables x and y are multiplied, and a logarithm is taken of its outcome. When each operation is performed, the backward function of the operation is also stored: in this case, the multiplication derivative and the logarithm derivative. During the backpropagation, these derivative functions are combined using the chain rule to calculate the gradients of the output with respect to both x and y.

An essential functionality of the backpropagation in PyTorch is that it can be done multiple times before performing an optimisation step. Usually, the computational graph is cleared after the backpropagation, as the gradients have already been computed and the graph is no longer needed. By specifying *retain_graph* to be true during the backpropagation, the computational graph is retained such that the gradient can be recalculated at some point. The multiple gradients that have been computed are then accumulated in the corresponding variables. This can be useful when using multiple loss functions, which are applied to different points in the neural network. A first loss can then be calculated from some output variable and be back-propagated. Then, the output variable can be used to continue computations with and obtain a new variable to compute the loss with.

3.3. The Anatomy-based Dose Prediction Model

In this section, the first model is considered to predict dose distributions in prostate cancer patients, which is the anatomy-based dose prediction neural network. This neural network takes the patient's anatomy as input and provides a dose distribution as output. It is trained on already treated patients, where the output dose distribution is compared to the actual delivered dose. Much research has already been done on models like these, which has shown that they are effective in predicting accurate dose distributions when given the proper training data.

3.3.1. Neural Network Architectures

Two different types of neural network architectures are considered in this thesis, both of which have proven successful in earlier research. The first deep learning model investigated is the U-net (see Section 2.5), for which the specific architecture of the network is shown in Figure 3.9. This particular design is a slight alteration of the U-net architecture used by Thierry Meerbothe [7], which has proven successful in predicting dose distributions of prostate cancer patients.

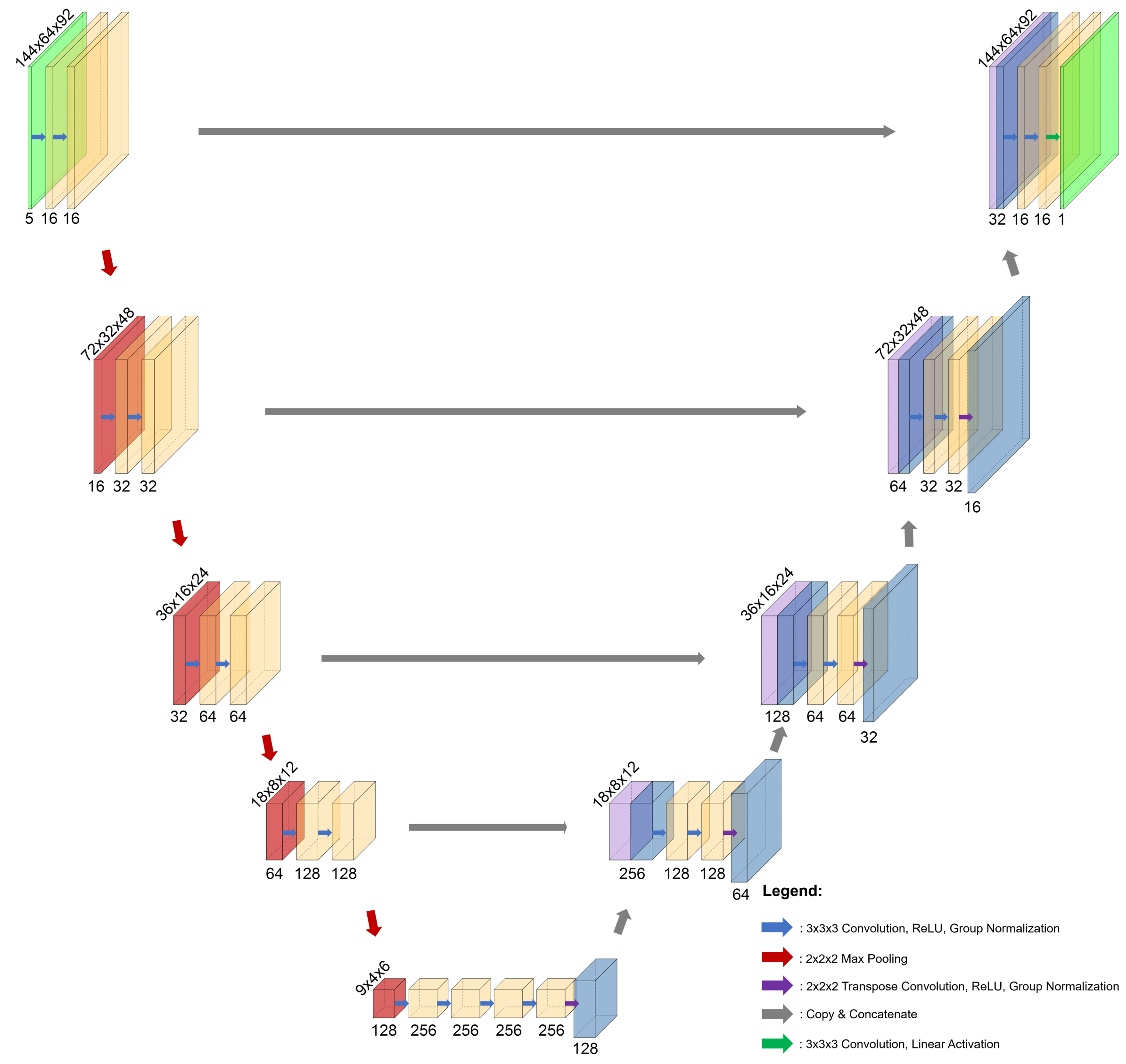


Figure 3.9: Architecture of the U-net that is used for predicting dose distributions based on the patient's anatomy. Note that the 3D volumes propagating through the network are represented in this figure by a 2D square due to the limitations in physical dimensionality. The numbers under the boxes represent the number of channels, while the diagonal numbers in the $x \times y \times z$ format define the size of the data at that point in the network. The different kinds of operations performed on the data are represented by the coloured arrows.

Each convolution is followed by a ReLU activation and a group normalisation layer. After two convolutions, the data is downsampled using a max-pooling operation. This sequence is repeated four times until the lowest level of the U-net is reached, where the learned features can be stored into the 256 different channels present. After a sequence of four convolutions, a transpose convolution is done with a stride of two to upsample the data again. The upsampled data is then concatenated with the data of that level just before the pooling operation through a skip connection. Then, a sequence of two 3D convolutions followed by a transpose convolution and concatenation is repeated three times. When reaching the upper level, the two 3D convolutions are finally followed by a convolution with a linear activation, producing a single output channel of the 3D image, holding the predicted dose distribution.

The second architecture that is investigated for the anatomy-based dose prediction model is the HD U-net. Figure 3.10 shows the specifics of the HD U-net network architecture used in this project. The overall structure is quite similar to the standard U-net, with an encoding arm and a decoding arm, five different levels of depth and skip connections. However, the traditional convolutional layers have been replaced by dense convolutional layers, which concatenate the input of the convolution to the output, essentially increasing the number of channels after each layer. Furthermore, the pooling layers are replaced by dense downsampling layers, where the data is downsampled using both a max-pooling layer and a strided convolutional layer. The two downsampled outputs are then concatenated. Lastly, the amount of output channels after each convolutional layer is kept constant at 16, in contrast to the standard U-net, where the amount of output channels depends on the depth in the network. This means that the number of channels present in the network increases at a constant rate (defined as the growth rate) of 16.

3.3.2. Loss Function

For dose prediction networks, the most common loss function to use is the Mean Squared Error (MSE), which is also the loss function used in other research [6], [41], [57]. The effectiveness of different loss functions for predicting dose distributions has already been extensively investigated by Meerbothe [7], and the result of this investigation is that the best predictions were obtained by making use of a Weighted Mean Squared Error (WMSE) as a loss function. In this loss function, the MSE loss in some voxels is weighed heavier than in others. Specifically, Table 3.1 shows the best-performing weights that have been found. These weights are then applied to the loss function as described in Equation 3.1.

$$L_{WMSE}(D_{pr}, D_{gt}) = \frac{1}{N}\sum_{i=0}^{N} w_i \left(D_{pr}^i - D_{gt}^i\right)^2 . \tag{3.1}$$

In Equation 3.1, L_{WMSE} denotes the WMSE, D_{pr} the predicted dose of all voxels, D_{gt} the ground truth dose of all voxels (as found in the RT Dose file), N the total number of voxels, D_{pr}^i the predicted dose in the specific voxel i, D_{gt}^i the ground truth dose in the specific voxel i and w_i the assigned weight for the specific voxel i.

Table 3.1: The weights that are used for the WMSE loss function. the value under each voxel type denotes the value of w_i for a voxel that belongs to that specific structure.

Voxel type	PTV	OAR	Body
Weight w_i	8	4	1

The reason for using the weights described in Table 3.1, is that it forces the network to perform better on the PTV and the OARs. This is helpful as clinically, the dose distribution in these structures is more important than the dose in the body. Therefore, these weights can improve the clinical quality of the predicted dose distribution.

3.3.3. Preventing Overfitting

The neural network architectures proposed in this section are very complex and contain many learnable parameters, while the dataset is relatively small. This is a perfect combination for creating overfitted models, where the model becomes very good at making predictions from the training set, but once a new example is tested, the model performs poorly. This is unwanted behaviour as the ultimate goal is to train the model to make predictions for patients it has not seen before. Therefore, a few tools are used to prevent the model from overfitting.

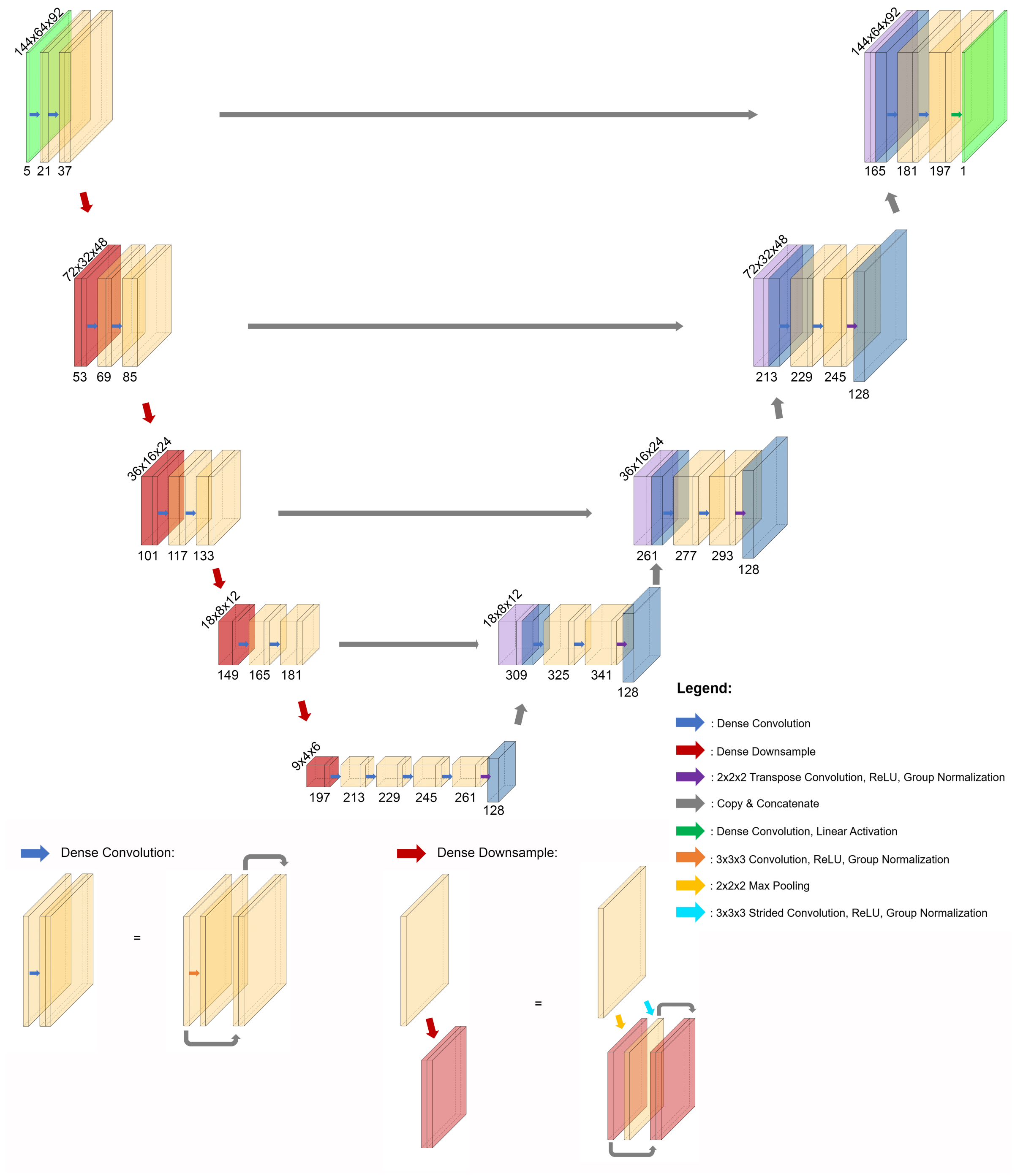


Figure 3.10: Architecture of the HD U-net that is used for the prediction. The architecture is similar to the standard U-net; only the convolutional and pooling layers are replaced by a dense convolutional and dense downsampling layer. The idea is that both layers concatenate the input to the output, increasing the number of channels after every layer.

Early Stopping

One helpful tool is early stopping, which is used during training and requires a secondary dataset, called the validation set, to be kept aside. During the training, the model is trained on the training set over multiple epochs, but during each epoch, the WMSE is also calculated for the validation set patients. Note that it is imperative that the neural network is not allowed to train on the validation set, so parameters must not be updated based on the loss function output from a validation patient. The idea is then that for each epoch, an average loss for the training set and an average loss for the validation set is computed, which should both be monitored during the training. Once the validation loss stops decreasing during training, while the training loss keeps falling, the model training should be stopped early, as this is a signal that the network has started to overfit. Figure 3.11 shows an example of a training/validation loss curve. Note that this tool helps in stopping the training before overfitting occurs but does not do anything to postpone the moment of overfitting.

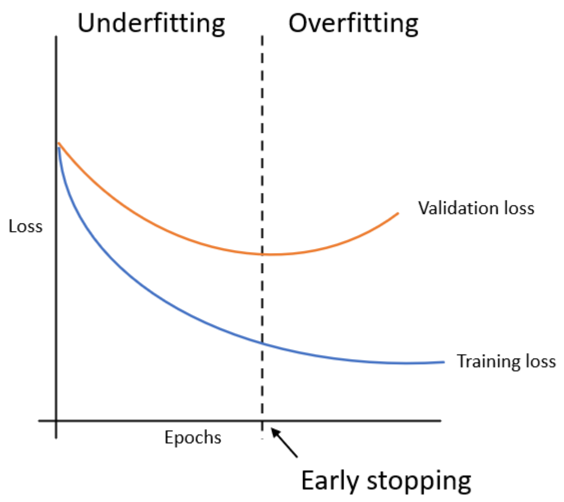


Figure 3.11: A visualisation of the mechanics behind early stopping. During training, both the validation and training loss should initially decrease. Once the validation loss stops decreasing but the training loss continues to decline, overfitting occurs. With early stopping, the training is stopped once this happens.

Dropout Regularisation
A second tool that is implemented to prevent overfitting is dropout regularisation. The idea behind dropout is that randomly some learnable parameters are temporarily set to zero during a forward through the network. Then the parameters are updated based on this forward, which means that the randomly zeroed parameters are not updated. The reason that this helps in preventing overfitting is that neural networks can show co-adaptive behaviour. This means that the model updates its parameters during training based on the value of some other parameter, making the parameter reliant on the presence of some other parameter. These co-adaptations do not generalise very well, which is why preventing them helps in preventing overfitting. By randomly zeroing out some parameters, a parameter can no longer rely on the presence of other parameters and, therefore, will learn not to be co-adaptive [58].

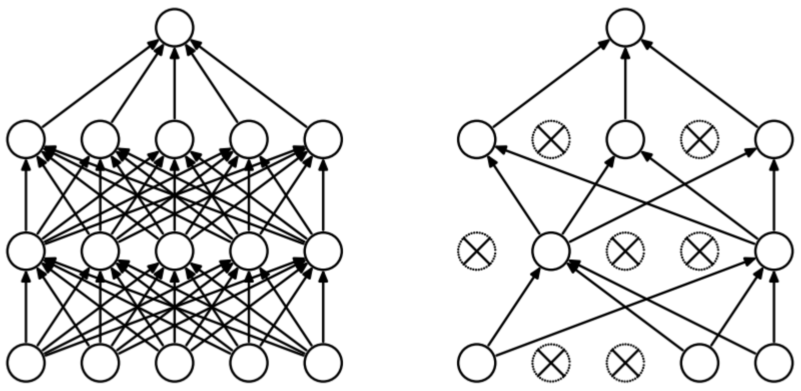


Figure 3.12: A fully connected network vs one with dropout applied [58]. Some weights are set to zero, effectively removing some of the nodes present. This forces the parameters not to be co-adaptive, preventing overfitting.

Weight Decay
The last tool used is weight decay. With weight decay, an extra penalty is added to the loss based on the parameter values of the model. The higher the weights, the larger the penalty, so effectively, the weights are forced to be not too high by this penalty. The reason that this helps in overfitting is that very large weights mean that the model can react heavily to a slight change of the input data.Typically, an overfitted model often has many parameters close to zero and a few extremely high values. A well-generalised model has more evenly spread low values across all the weights. Weight decay is added to the loss function as

$$L_f = L_i + \lambda \sum_{j=1}^{N} \theta_j^2, \tag{3.2}$$

where L_f stands for the final loss including the regularisation penalty, L_i for the initial loss, λ for the weight decay, N for the total number of parameters and θ_j denotes the j^{th} parameter of the model. The weight decay can be altered to achieve the desired amount of regularisation.

3.3.4. Training and Evaluation

As mentioned in Section 3.1, the provided datasets consist of 89 and 109 patients, which have been split into a training, validation and test subset. The training subset, which is used for the training itself, consists of 64 patients in dataset I and 77 patients in dataset II. The validation subset contains 13 patients for dataset I and 16 for dataset II. It is used to monitor the model's performance during the training and for the implementation of early stopping. The test subset consists of 12 patients in dataset I and 16 patients in dataset II. It is kept aside during the training and used to evaluate the model after training.

Two different methods are investigated to implement data augmentation. The first method is based on a multi-phased training implementation conducted by Kearney et al. [57], where the model is trained on different training sets in multiple phases. In the case of this project, two training phases are used:

- In the first training phase, a single training patient is selected, and all different augmentations as mentioned in Subsection 3.1.3 are performed on this specific patient, generating a dataset of 87 different artificial patients. This training phase is used to make the model recognise the consequence of small rotations and translations of the body on the dose distribution.
- In the second training phase, all augmentations are turned off, and the model is trained on the complete training set, which is where the most important deep learning occurs.

The second method of implementing data augmentation is more conventional, where five augmentations are randomly selected from the list of augmentations as described in Subsection 3.1.3 for each training patient. These randomly chosen augmentations are applied to the training patient, increasing the training set by five-fold. This training method consists of only a single training phase.

The training of the model is continued until the validation loss no longer decreases. At that point, the model is saved and allowed to continue training for ten epochs to see if it can still find a new loss minimum. If this is not the case, the training is ended. For the optimisation of the model weights, the Adam optimiser is used with an initial learning rate of 10^{-3}. All training is done on a Linux virtual machine with an Nvidia GRID T4-16Q GPU, containing 16 GB of VRAM.

Qualitative Model Assessment
After the model has been trained, some form of evaluation is needed to assess the model's performance and for comparison with literature and other models. A good way to start the assessment is to take a single test patient and visually inspect the difference between a predicted dose slice and a ground truth dose slice.

Furthermore, a (cumulative) Dose Volume Histogram (DVH) can give many insights on the distribution of dose within a particular structure. The inspection of the DVH of both the prediction and the ground truth also provides a good insight into the prediction quality. To be consistent in the choice of the example test patients for the qualitative assessment, the best and the worst-performing patient are chosen based on the loss value.

Quantitative DVH Assessment
Ideally, a quantitative assessment of the model performance should also be done to make objective statements about a potentially better performing model. This assessment should include all test patients. The first quantitative assessment is based on the DVH curves of all patients, where the Mean Absolute Error (MAE) on all points of the DVH curve is computed and averaged over all test patients. This computation is done according to Equation 3.3.

$$MAE = \frac{1}{N} \sum_{i=0} \left| p_{gt}^{i} - p_{pred}^{i} \right| . \tag{3.3}$$

In Equation 3.3, MAE stands for the mean absolute error in Gy of a patient DVH, p_{gt}^{i} for the i^{th} point on the DVH curve of the ground truth structure, p_{pred}^{i} for the i^{th} point on the DVH curve of the predicted structure and N for the total number of DVH points.

Assessment of Dose Coverage Statistics
Apart from using the DVH and performing a visual inspection, another commonly used quality assessment method for dose distributions is the evaluation of dose coverage statistics. By looking at the difference between the prediction and ground truth values of these statistics, another assessment can be done on the clinical quality of the prediction. An example of a dose coverage statistic is the D_{95}, which is defined as the minimum dose 95% of a specific structure receives. An overview of all the used dose coverage statistics can be found in Table 3.2.

Table 3.2: Overview of used dose coverage statistics. The difference between the ground truth and the predicted value of these statistics can be used to assess the performance of a model.

Variable	Definition
D_{95}	Minimum dose that 95% of a certain structure receives
D_{98}	Minimum dose that 98% of a certain structure receives
D_{max}	Maximum dose within a certain structure
D_{mean}	Mean dose received by a certain structure
V_{45}	the volume of a certain structure that receives at least 45 Gy of dose

Table 3.3: Overview of all evaluation methods used to assess the predicted dose distribution from the Anatomy-based dose prediction model.

Assessed object	Method	Purpose
Best and worst-performing dose distribution slice	Visual inspection	Quick initial assessment on rough quality
Best and worst-performing DVH	Visual inspection	Quick initial assessment on rough quality
DVH of all test patients	MAE (see Equation 3.3)	Complete quantitative assessment of all test patients
Dose coverage statistics	Various metrics (see Table 3.2)	Assessment of clinical quality
Isodose contours	SDI (See Equation 3.4)	Assessment of isodose contour overlap

Sørensen-Dice Index
Another way to assess the model's performance is to look at the overlap between the internal volumes of the isodose contours of the ground truth and predicted dose distribution. In the best case, these volumes overlap entirely, and in the worst case, there is no overlap at all. A well-known method to quantify the amount of overlap between two volumes is with the Sørensen-Dice index (SDI) [59], which is defined as

$$SDI = \frac{2\,|X \cap Y|}{|X| + |Y|}, \tag{3.4}$$

where X and Y represent the two isodose internal volumes. The SDI always has a value somewhere between zero and one, where zero means no overlap and one means complete overlap.

3.4. The Segment Prediction Model

One of the main goals of this project is to improve the dose prediction by including physical knowledge about the way the dose is delivered to the model. For this, the Physics-guided prediction model has been created, which consists of multiple components, each in need of its own description. For a quick overview of the Physics-guided dose prediction model, please see Subsection 3.6.1 and Figure 3.22. In this section, one of the components of the Physics-guided prediction model is discussed, which is the Segment prediction model. This model is used to predict the MLC shapes and beam intensities needed to define a treatment plan and calculate a dose distribution.

3.4.1. Segment Data Extraction

To train the Segment prediction model, the ground truth of the segments is required. The machine parameter data is of a different format than the 3D images of the dose, structure and CT arrays. The VMAT plan is defined in 142 segments on the 71 control points of the rotation arc. For each control point, the MLC aperture and the beam intensity is given in the RT Plan DICOM file for both the clockwise and counter-clockwise rotating beam.

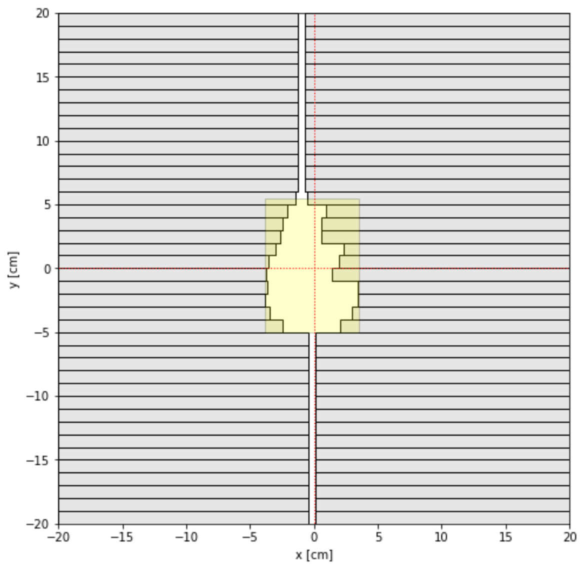
The MLC position data is stored per control point, and for each control point, the position of all 80 leaves and the four jaws is defined in mm. For each jaw and leaf, only a single coordinate is needed to determine its position, as the orientation of the leaves and jaws is also given in the RT Plan file. With the leaf width known, which is 1 cm, the MLC positioning can be reconstructed. This information is then used to create binary images of the MLC apertures, where a one represents an open pixel, meaning that the MLC and the four jaws are not blocking the photon beam, while a zero represents a closed pixel.

The MLC image is projected on a 160 by 160 pixel grid, where each pixel represents a 1 by 1 mm area. Figure 3.13 shows an example of a specific MLC shape and how the MLC positions are translated to a binary map. As seen in the figure, the binary map is much more zoomed in on the beam centre. This is done as all MLC leaf positions of the entire dataset are within this field of view. By zooming in on the region of interest, the deep learning model can focus better on the important part. To be able to treat the treatment plan of one patient as a single data point, all MLC maps are stacked on top of each other as if they are slices, forming a 3D image that defines the MLC aperture for all control points.

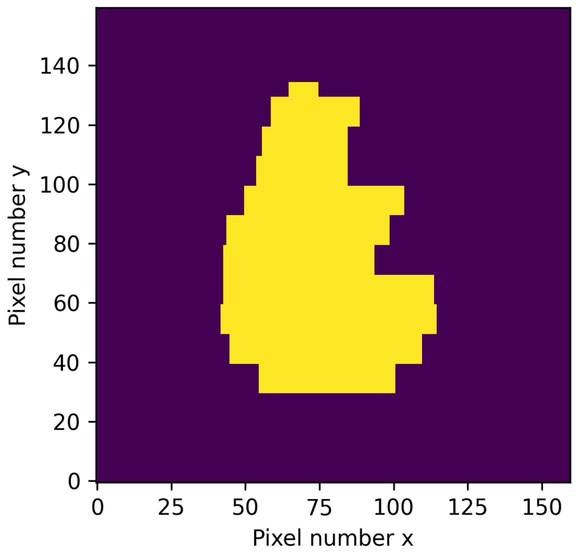
Apart from the beam aperture, a beam intensity value is also necessary to deliver the dose. In the DICOM file, Monitoring Units (MU) are used to denote the intensity of the photon beam. In general, a single MU is equal to 1 cGy of absorbed dose in water under precisely calibrated conditions of the LINAC [60]. This unit is therefore explicitly defined for a specific LINAC machine. Two MU values are given in the RT Plan file, one for each beam. The total MU value for a particular beam is then distributed over the segments, according to the cumulative weight values assigned to the control points. As the first weight always starts at zero, the cumulative weights only determine the amount of dose delivered between two control points. Therefore 140 new control points are defined precisely in the middle of the old control points, where the MU distribution is determined by the difference of the cumulative weight value between the two old control points surrounding it. This difference between two cumulative weights is then multiplied by the corresponding beam MU value to obtain a per control point MU value, as is shown in Figure 3.14. Because the new control points are taken at positions that are at different locations, the MLC maps are also interpolated to obtain an approximation of their position on the 140 new control points.

3.4.2. Beam's Eye View Data Pre-processing

The challenge in predicting machine parameters is that information on the gantry head position must be included in the data input. Furthermore, the model needs to understand somehow the relation between the MLC shape and the shape of different organs at a specific point on the arc. To combine anatomical information and the gantry angle position in the input data, Beam's Eye View (BEV) images are created of the CT, dose and structure data, which are used as input to the Segment prediction model.

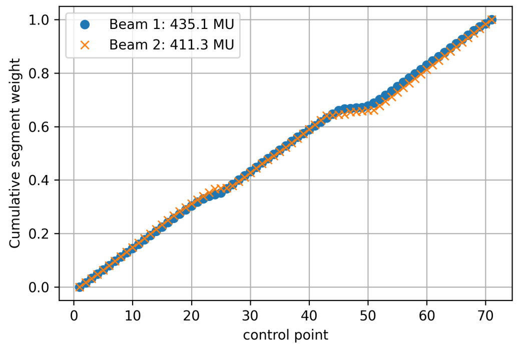


(a) MLC data extracted and visualised

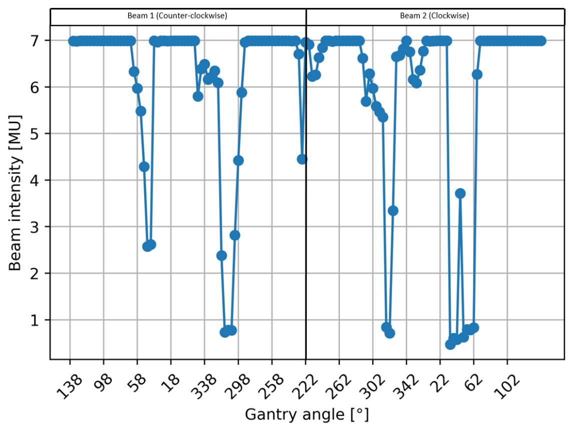


(b) MLC aperture projected on a 160 by 160 pixel grid.

Figure 3.13: An example of a single extracted MLC configuration. The left figure shows how the raw positioning data can be reconstructed into the beam limiting device position. The grey rectangles represent individual leaves. The yellow rectangle represents the aperture of the four jaw positions. The intersection of the two red, dotted lines denotes the origin, which is aligned with the isocentre. The right figure shows a binary map in which the opening of the jaws and leaves are combined into a single aperture. A yellow pixel represents an open location, while a purple pixel represents a blocked location.



(a) Raw beam intensity data present in the RT Plan file.



(b) Conversion to per control point MU values

Figure 3.14: An example of the extracted beam intensity data of a single patient. The left figure shows the raw data as it is stored in the DICOM file, where the two beams each have 71 segments with cumulative weights defined and a total amount of Monitoring Units. The right figure shows the resulting 140 beam intensities, with the MUs distributed over the control points.

In essence, the BEV image of a particular 3D image is a 2D image representing a projection of the 3D image from a certain point of view. For example, imagine a certain 3D shape, such as a pyramid. A BEV image from the top of the pyramid looks like a square, while an image from another point could look like a triangle or some different shape. The same idea can be applied to create an image of, for example, the PTV structure from the point of view of the gantry head. To make these images, each 3D array is rotated opposite to the gantry angle for each control point. The voxel values are then summed together in the direction of the gantry point of view, creating a 2D image. Figure 3.15 shows an overview of the BEV pre-processing workflow. Figure 3.16 shows an example of some single 2D BEV PTV images for a specific gantry angle and patient. Similar to the MLC maps, the BEV images of all segments are stacked on top of each other like slices to obtain 3D images that contain all BEV information for a specific structure.

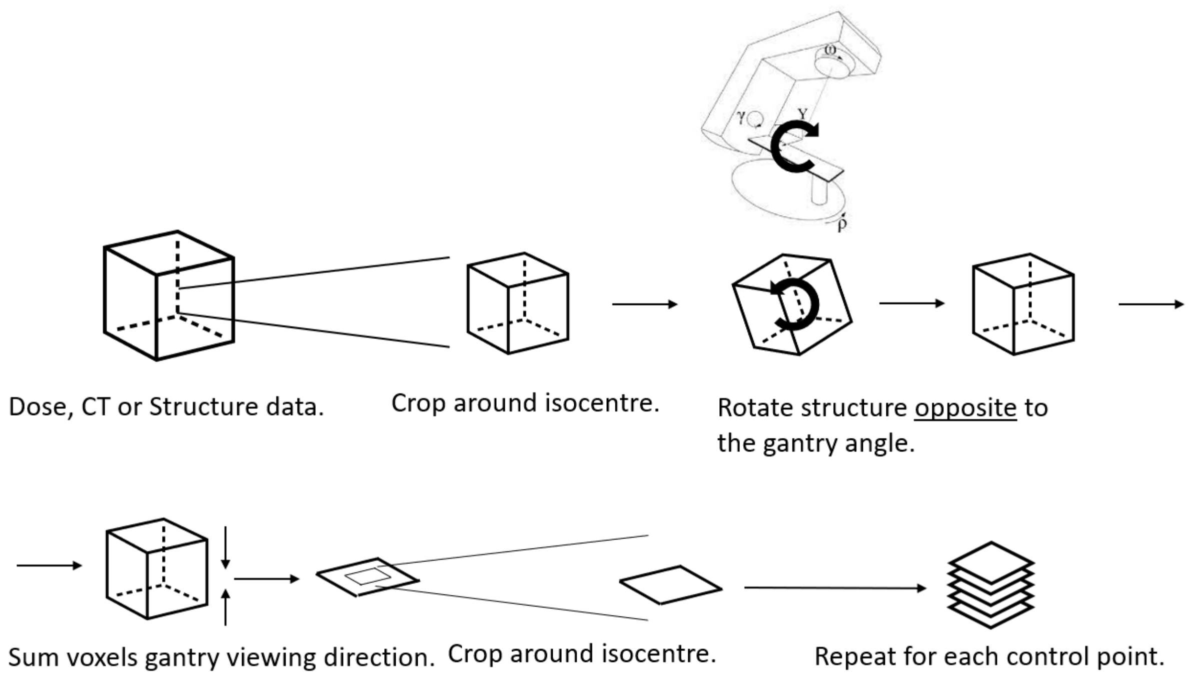

Dose, CT or Structure data.
Crop around isocentre.
Rotate structure opposite to the gantry angle.
Sum voxels gantry viewing direction.
Crop around isocentre.
Repeat for each control point.

Figure 3.15: Overview of the BEV pre-processing workflow. The different cropping operations are done to obtain a field of view that is the same as in the MLC binary maps.

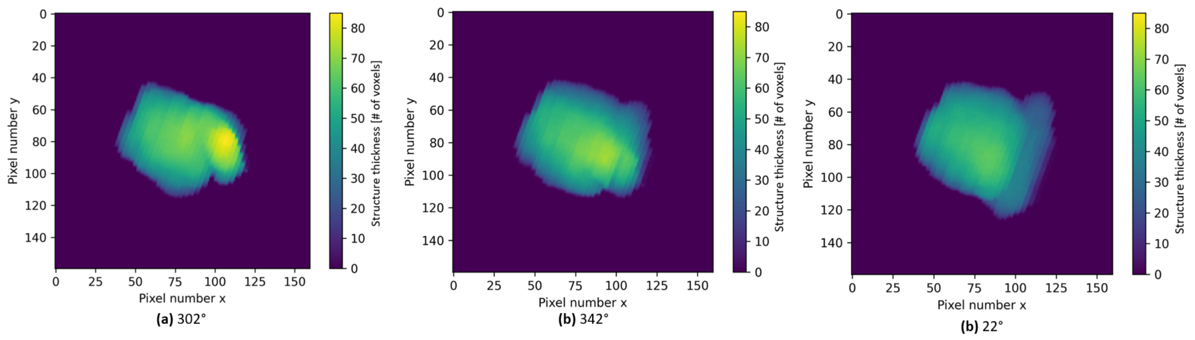


Figure 3.16: Three BEV images of the PTV structure of an example patient, ten control points apart. The images are projected on the same grid as the MLC binary maps.

3.4.3. Network Architecture

To predict the MLC maps and beam intensities from the BEV images, different network architectures have been tested with varying success. The best results have been obtained with a single-encoder, dual-decoder pix2pix-inspired U-net [61]. The pix2pix network has proven successful in many applications for image-to-image translation and has also been implemented by Hibbard to serve as a segment predictor [62]. In our case, the segment prediction can be seen as a 3D image-to-image translation, where BEV images need to be translated to MLC binary images. The only difference is that in this case, also the amount of MUs need to be predicted for each segment, which is why a second decoder is required to decode the learned features to beam intensities as well. Figure 3.17 shows the specifics of the network architecture that is used.

The main difference with the networks used for the Anatomy-based dose prediction model is that the data is downsampled by using a stride of two in the convolutional layers. Furthermore, leaky ReLUs are used as activation functions in the encoder arm. Moreover, there are now two decoder arms, where the first one decodes the data to MLC maps, making use of transpose convolutions and skip connections.

In the last layer of the first decoder, a sigmoid activation function is used to translate the output to a range between zero and one, allowing the model to make an estimation on the MLC binary map. The second arm is used to decode the data to beam intensities. As this is data of a different format, no skip connections are used here but instead Transpose convolutional layers that upsample the data in only one dimension. Note that in the input and output data, the first dimension is of size 160, even though there are only 140 segments. This is done to create a dataset that can be downsampled to deeper levels, and the missing segments are zero-padded in.

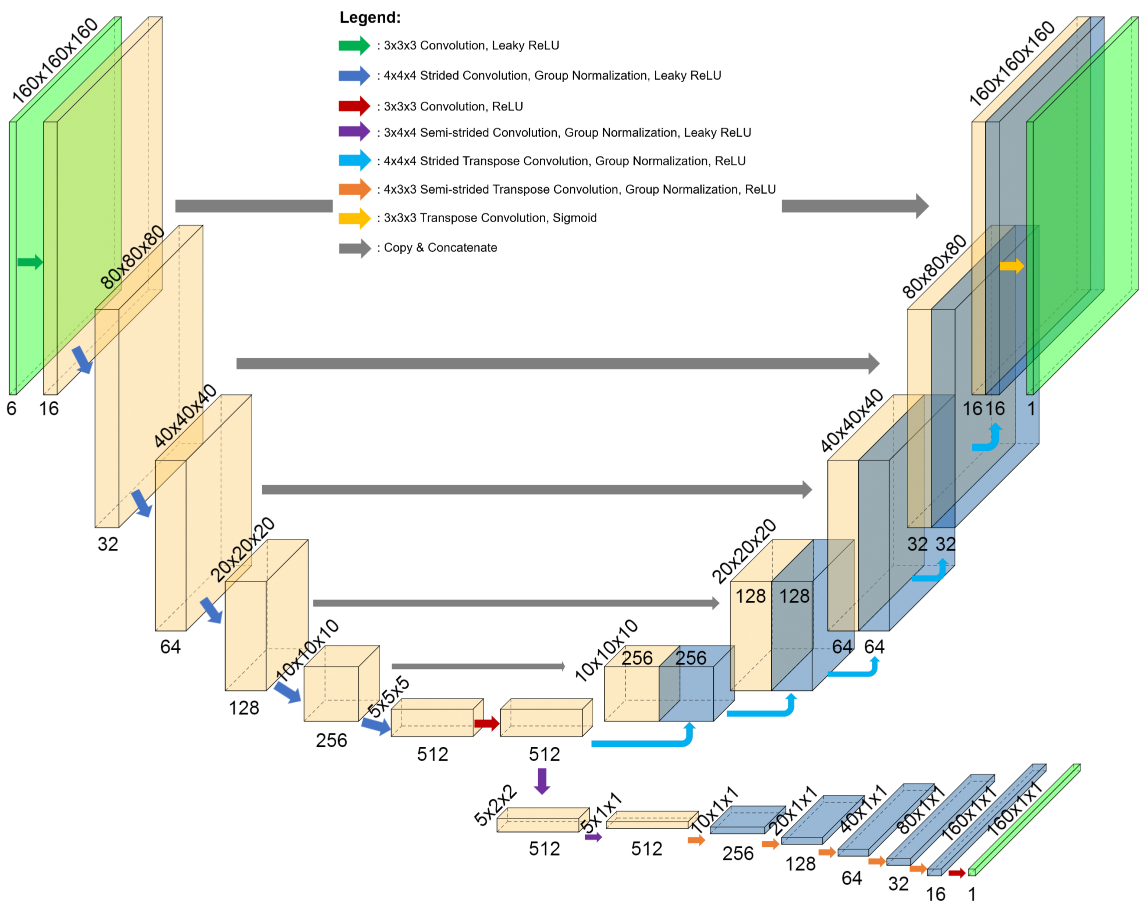

Legend:
: 3x3x3 Convolution, Leaky ReLU
: 4x4x4 Strided Convolution, Group Normalization, Leaky ReLU
: 3x3x3 Convolution, ReLU
: 3x4x4 Semi-strided Convolution, Group Normalization, Leaky ReLU
: 4x4x4 Strided Transpose Convolution, Group Normalization, ReLU
: 4x3x3 Semi-strided Transpose Convolution, Group Normalization, ReLU
: 3x3x3 Transpose Convolution, Sigmoid
: Copy & Concatenate
160x160x160
6
16
80x80x80
32
40x40x40
64
20x20x20
128
10x10x10
256
5x5x5
512
512
5x2x2
512
5x1x1
512
10x1x1
256
20x1x1
128
40x1x1
64
80x1x1
32
160x1x1
16
160x1x1
1
10x10x10
256
256
20x20x20
128
128
40x40x40
64
64
80x80x80
32
32
160x160x160
16
16
1

Figure 3.17: Architecture of the Pix2pix inspired U-net that is used for the segment prediction. Again the 2D squares represent the 3D data. The numbers below the cuboids show the number of channels in the data, while the diagonal numbers show the size of the data. The differently coloured arrows represent the different operations done.

3.4.4. Training and Evaluation

The training of the segment predictor is done similarly to the training of the Anatomy-based dose prediction model. The main difference is that data augmentation is not implemented, as it is much less straightforward to predict how the augmentation of the input data affects the ground truth data. Apart from that, overfitting prevention methods like early stopping, dropout regularisation and weight decay are implemented during training here as well.

Another difference is how the loss is calculated for this model. as this model produces two separate predictions, two different loss calculations are done for the MLC maps and the MU values. As the MU values are generally some value between zero and 10, a simple MSE will suffice to compute the loss of the predicted beam intensity. For the loss on the MLC map prediction, the Binary Cross-Entropy loss is used to assess the loss for each pixel. This is, in general, the best loss function to use when a sigmoid activation function is used in the final layer. The two losses are then summed together, where the weighing of the two losses is adjusted to ensure that they are of the same order of magnitude. The

MSE and BCE loss are computed according to Equations 2.13 and 2.14, respectively.

MLC Prediction Post-processing
After the model has been trained, it should be able to generate MLC maps, where all pixel values lie somewhere between zero and one, representing the probability of it being an open pixel. To be able to use the prediction in an actual treatment plan, it needs to be binarised into a map of zeros and ones for it to be a realistic MLC map, as a pixel can not be slightly open or closed. As this is a non-differentiable operation, this post-processing step can not be included in the training process and is therefore only for the purpose of testing and using the prediction as an actual plan.

Furthermore, there is no guarantee that the contour of the MLC opening follows the shape of the individual leaves. This means that there is still a post-processing step needed to transform the prediction into a realistic MLC map. Figure 3.18 shows an example of how a prediction is processed into an actual MLC map.

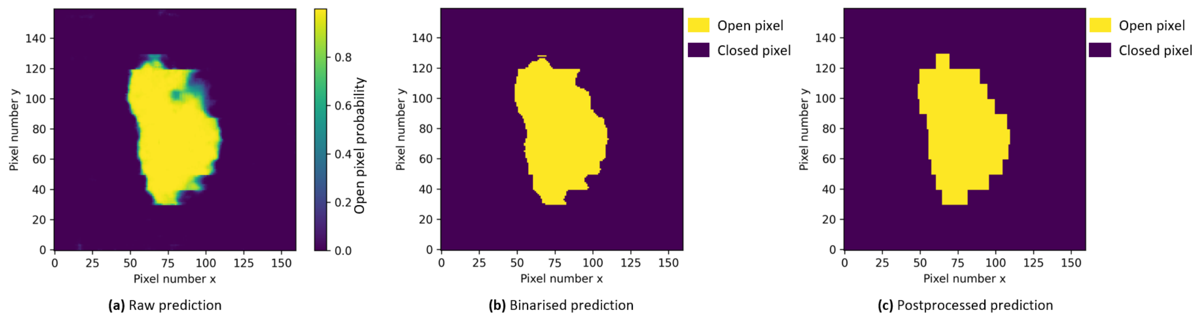


Figure 3.18: Example of a single predicted MLC shape. The left figure shows the raw output of the predictor, where each pixel contains a probability between zero and one to be open. The middle figure shows the result when all probabilities above 0.5 are set to one and all probabilities below 0.5 to zero. The right figure shows the post-processed MLC aperture, where the binary probability map is transformed into a realistic MLC shape.

Segment Evaluation Metrics
To evaluate the quality of the model's predictions, some evaluation metrics are used to compare the ground truth MLC maps and beam intensities to the predictions. As the beam intensity data is only a set of 140 MU values, evaluation of these is done using the Mean Absolute Error (MAE), which is defined as

$$MAE(m_{pr}, m_{gt}) = \frac{1}{140}\sum_{i=0}^{140}\left|m_{pr}^{i} - m_{gt}^{i}\right|, \tag{3.5}$$

where m_{pr} stands for the predicted beam intensities, m_{gt} for the ground truth beam intensities, m_{pr}^{i} for the predicted beam intensity of segment i and m_{gt}^{i} for the ground truth beam intensity of segment i.

The evaluation of the MLC maps is done with a similar metric, which is called the Mean Absolute Leaf Difference (MALD). This metric is also used in other research on segment predictors [62]. The MALD directly evaluates the absolute difference between the leaf positions in the prediction and the ground truth. As this is, in the end, the only thing necessary for the treatment plan, it is a reasonable evaluation metric to use. To compute this difference, it is required to translate the post-processed binary prediction maps to the individual leaf positions.

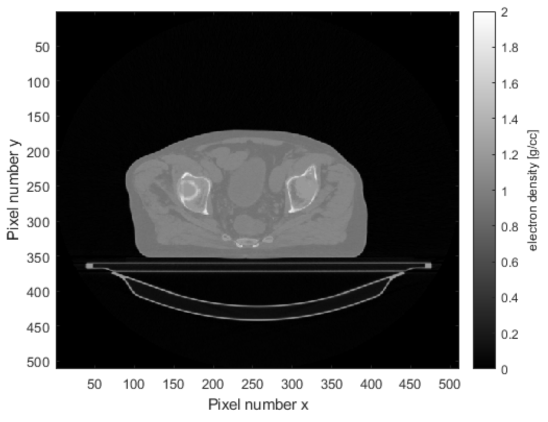
3.5. Dose Engine

So far, two deep learning models have been discussed: one that predicts a dose distribution and one that predicts the MLC maps and the beam intensities. To include physics in the deep learning models, it is important that the treatment plan data can be transformed into the corresponding dose distribution. In Section 2.2.3, different dose calculation algorithms have been discussed. In this section, the dose calculation implementation used in this project is addressed, which can transform any treatment plan, ground truth or predicted, into a dose distribution.

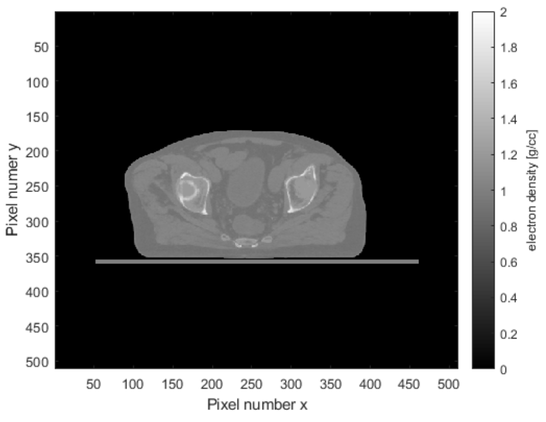
3.5.1. Matrad Pencil Beam Dose Calculation

The dose calculation implementation is based on the pencil beam algorithm of the Matrad open-source software for radiation treatment planning [63]. In Section 2.2.3, the finer details of how such an algorithm works have already been explained. As this dose calculation method is used in the training of deep learning models, it is essential that the dose calculation can be done quickly. To this end, the choice has been made to precompute the dose influence matrices for all pencil beams, all control points and all patients in the dataset and to store all the computed dose influence matrices in an extensive database. During run-time, the necessary pencil beams can then be quickly selected and loaded to form a total dose distribution based on the MLC maps and beam intensities.

A grid of 14 by 14 pencil beams is used to represent all different MLC maps. A bixel width of 10 mm has been chosen, which defines the size of the pencil beams that are used. The chosen bixel width is small enough to distinguish between individual leaves with different bixels but large enough to keep the database storage size acceptable. For each patient, the 196 pencil beam positions are inserted into the dose engine, together with the CT data of the specified patient. One thing to note is that the CT table on which the patient lies is different from the table used during treatment. As part of the radiation is delivered from below the patient, this could potentially impact the calculated dose. To solve this issue, the “table” structure is used from the RT Structure data. All patient-exterior CT data is then set to zero, and the table structure is used to burn a 1 g/cc electron density into the CT array, which is the known electron density of the VMAT table [64]. In Figure 3.19, an example slice comparison is shown between the original CT and the modified CT.



(a) Original CT slice



(b) Modified CT slice

Figure 3.19: Example comparison between the initial CT slice, as it is in the DICOM file, and the modified CT slice with the VMAT table artificially added. When comparing the two figures, it can be seen that all detail outside the patient volume has been removed and that a new table is added to the CT.

With the CT corrected and the pencil beam positions defined, the database can be generated. 196 dose influence matrices are calculated per segment: one for each pencil beam. For each patient, 140 segments are calculated, which means that for all 109 patients in dataset II, a total of almost three million pencil beam dose influence matrices are computed.

To minimise the storage required for the database, these matrices are converted to sparse arrays and flattened into lexicographically ordered dose influence vectors. As the individual pencil beams are small, the dose delivered by a single beam only affects a small part of the patient volume, which means many voxels will have a dose value of 0 Gy. Therefore, using sparse arrays to store the pencil beam dose will significantly decrease the storage space required. Furthermore, the hdf5 format is used to store the data, which is an open-source file format capable of relatively quick compression to reduce file size. After all calculations are done, a database of approximately 500 GB in storage size is created.

3.5.2. Total dose calculation

With the database created, a series of quick computations can convert the ground truth or predicted MLC maps and beam intensities into a calculated dose. To achieve this, a few steps are required. First, the MLC maps must be translated to bixel weights, which are intensity weights assigned to the individual pencil beams. Next, the pencil beam vectors must be combined into the per-segment dose influence vectors. Lastly, the beam intensities and per-segment dose influence vectors are combined into a total dose distribution.

MLC Map Conversion to Bixel Weights

The MLC maps are 160 by 160 binary maps, which define the location and shape of the MLC aperture. As the pencil beams are only defined on a 14 by 14 grid, the number of open pixels close to a specific pencil beam position determine how much dose should be delivered by that pencil beam, which is done using pencil beam weights. If all pixels are open near a specific pencil beam, that beam will receive a weight of one, and if all pixels are closed, it will receive a weight of zero. If some pixels are closed, and some are open, the weight will be of a fraction equal to the fraction of open pixels. Figure 3.20 shows a visualisation of this conversion process.

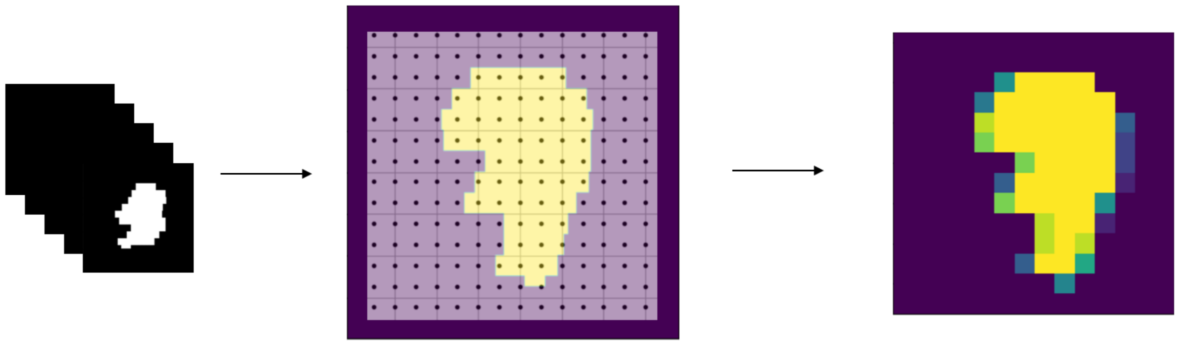


Figure 3.20: Overview of the MLC conversion to pencil beam weights. For each MLC map present in the (predicted) treatment plan, the 196 pencil beam positions are, in essence, overlayed on the MLC map. The weight value assigned to each pencil beam is then determined by the number of open pixels in the direct vicinity of that beam. The right image shows a map of the resulting pencil beam weights for this specific example.

Per-segment Dose Influence Vector Calculation

The next step is to compute the per-segment dose influence vectors for all segments. This is done through a matrix-vector multiplication, where the matrix contains all dose influence vectors of the pencil beams and the vector the bixel weights, which are obtained from the MLC maps. The per-segment dose influence vector is then obtained as

$$\begin{bmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,196} \\ d_{2,1} & d_{2,2} & \dots & d_{2,196} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N,1} & d_{N,2} & \dots & d_{N,196} \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{196} \end{bmatrix} = \begin{bmatrix} d_{1,1} \cdot q_1 + d_{1,2} \cdot q_2 + \dots + d_{1,196} \cdot q_{196} \\ d_{2,1} \cdot q_1 + d_{2,2} \cdot q_2 + \dots + d_{2,196} \cdot q_{196} \\ \vdots \\ d_{N,1} \cdot q_1 + d_{N,2} \cdot q_2 + \dots + d_{N,196} \cdot q_{196} \end{bmatrix}, \qquad (3.6)$$

where $d_{i,j}$ stands for the dose influence in the i^{th} voxel from pencil beam j, q_j for the bixel weight of pencil beam j and N for the total number of voxels. With this single operation, all pencil beam dose influences are multiplied by their corresponding pencil beam weights and then summed together to obtain a single dose influence.

Total Dose Calculation

The last step is to combine the per-segment dose influence vectors with the beam intensities, and obtain the total dose distribution. This is simply done by multiplying the dose influence of all segments with the corresponding beam intensities. After that, all per-segment doses are summed together and reshaped into their original 3D shape.

As mentioned in Section 3.4, the exact calibration of the Monitoring Unit is different for each specific machine. This means that a final multiplication with a single scalar is needed to account for this specific calibration. As it is not that simple to find this scalar without performing experiments, the scalar has been determined empirically by calculating the dose of all patients in the dataset using the ground truth treatment plan, and comparing it to the ground truth dose in the RT Dose file. The scalar multiplication value is then altered until the Matrad doses are as close as possible to the ground truth doses. Figure 3.21 shows an example slice of a pencil beam, segment and total dose influence matrix.

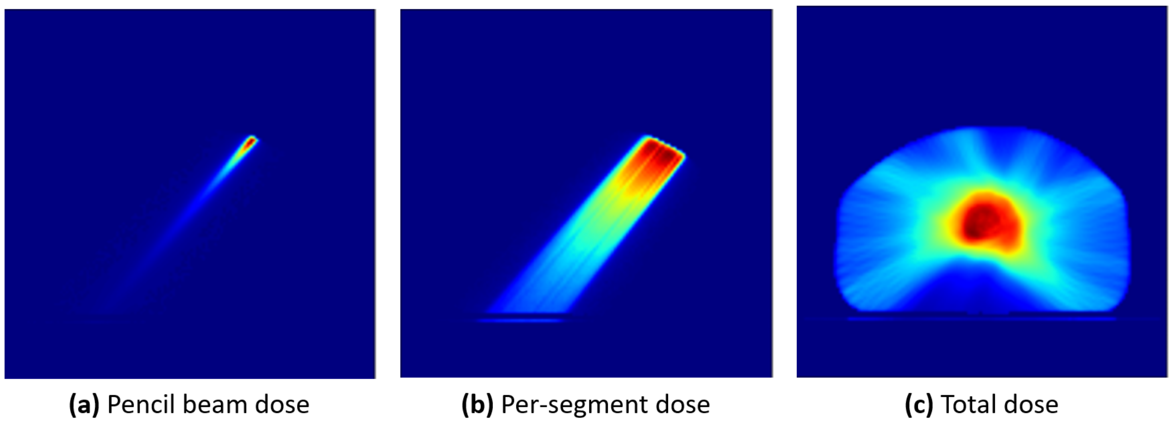


Figure 3.21: From left to right: a slice of an individual pencil beam, an individual segment and a total dose influence matrix. Note that the colour gradients are scaled differently in each image.

3.5.3. Dose Engine Evaluation

It is important to verify that the dose engine works correctly and identify differences with the ground truth dose. As this dose is created with the Pinnacle dose engine, which is based on the collapsed cone convolution algorithm, it can be expected that some differences are found. The comparison between the dose engines is made by using the ground truth treatment plan data as input to the dose engine and calculating a dose distribution for all patients in the dataset. Note that for this part of the thesis, only dataset II can be used as there is no CT data available in dataset I. After calculating a new dose distribution for all patients, a comparison can be made between the Pinnacle dose and the Matrad dose. Evaluation metrics, as described in Subsection 3.3.4, can also be used for the comparison that is done here.

3.6. Physics-Guided Prediction Model

So far, three individual components or models have been described, each capable of predicting or calculating either a dose distribution or the treatment plan. Now the last step is to combine these individual components into a complete model, which can predict both the treatment plan and a deliverable dose distribution that belongs to this treatment plan, based on the anatomic input data given.

3.6.1. Model Definition

In the Physics-guided prediction model, the two deep learning models are combined with the dose engine to obtain a deliverable dose and the treatment plan as output. The idea of this model is that first, a dose prediction is made using the Anatomy-based dose prediction model as described in Section 3.3. This dose prediction is then BEV processed and combined with BEV images of the CT and structure set. These BEV images are created using the BEV processing methodology described in Subsection 3.4.2. The BEV data is then used as input to the Segment prediction model, described in Section 3.4, which predicts a treatment plan. This treatment plan should then partially be based on the patient's anatomy and partially on the dose predicted by the Anatomy-based prediction model. Next, the predicted treatment plan data is used as input to the dose engine, which computes the final dose distribution. Figure 3.22 shows an overview of the entire model.

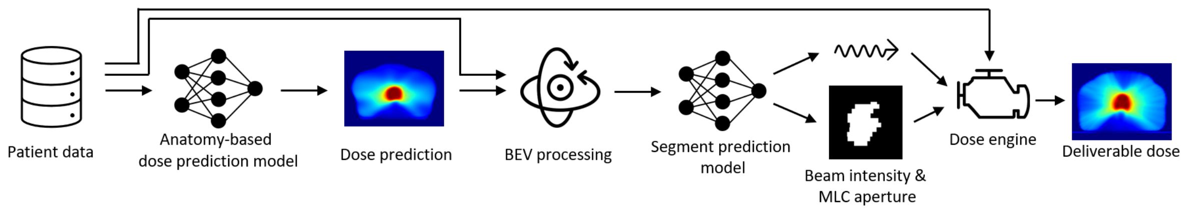


Figure 3.22: Overview of the Physics-guided prediction model. The patient data is used as input to the first deep learning model, which predicts a dose distribution. Then, the BEV processed patient data, including the predicted dose, are used as input to the second model, which predicts the treatment plan. This predicted treatment plan is then used as input of the dose engine, combined with the patient CT. This results in a deliverable dose as output.

The model has been designed such that the dose produced by the model is based on deep learning predictions, but because it is made by a dose engine, it is by definition respecting the physical rules of dose delivery. Furthermore, by using a segment prediction, the treatment plan is also predicted, which is directly related to the dose distribution. This means that this model produces all required information to deliver a treatment plan while also showing the dose that it delivers.

3.6.2. Model Training and Evaluation

The model described contains different components, some of which are deterministic, while others are neural networks with trainable parameters. The question then becomes how to train a complex model like this? Two different training methods are examined.

Training from Scratch

The first training method is without any pre-training of the models. This means that the trainable parameters of the models are randomly initialised. A single training epoch then consists of three steps: the dose prediction training step, the segment prediction training step and the deliverable dose training step.

- During the dose prediction training step, the Anatomy-based dose prediction model is simply trained as it would be trained on its own. In other words, the predicted dose output is compared to the ground truth Pinnacle dose, a loss is computed of the difference between the two, and the parameters of this model are optimised after backpropagating this loss through the model. An overview of this training step can be found in Figure 3.23.
- The segment prediction training step is a bit more intricate. In this step, the segment prediction of the Segment prediction model is compared to the ground truth treatment plan. An overview of this training step is shown in Figure 3.24. The part that complicates the training here is that the segment prediction uses the BEV dose distribution as an input, which is dependent on the dose prediction delivered by the first deep learning model. This means that the gradients have to be backpropagated not only through the Segment prediction model but also through the Anatomy-based deep learning model. Furthermore, it is therefore necessary that the BEV processing step performed here is differentiable to be able to back-propagate the gradients to the Anatomy-based prediction model.
- In the last training step, the input data is fed forward through the entire model, now including the dose engine. Here, the individual per-segment doses are compared to the ground truth segment doses, and all segment losses are summed together. Similar to the previous training step, these losses are then backpropagated through the entire model, which means that also the dose engine can only contain operations that are differentiable. Figure 3.25 shows an overview of the third training step.

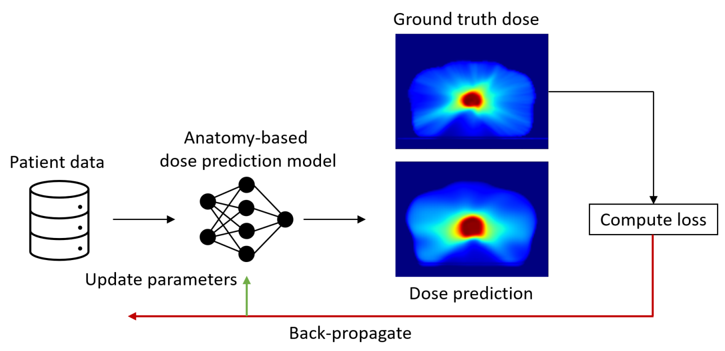


Figure 3.23: Schematic of the first training step of the Physics-guided prediction model. In this step, the Anatomy-based dose prediction model is simply optimised by computing a loss from the difference between the predicted dose and the ground truth dose.

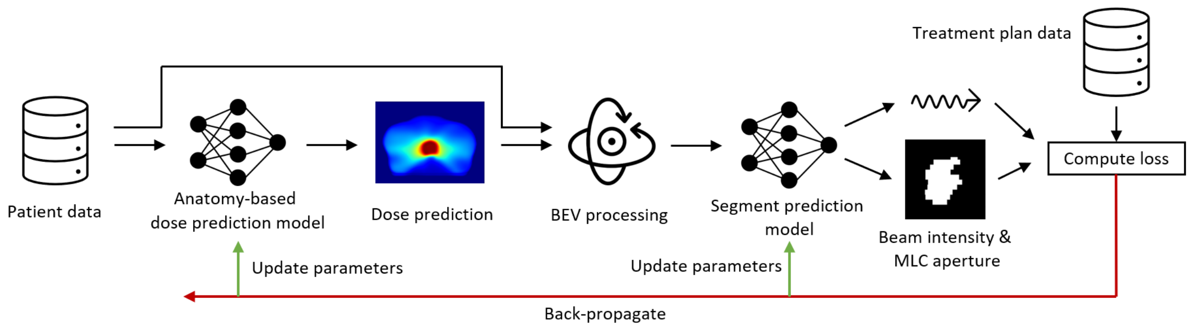


Figure 3.24: Schematic of the second training step of the Physics-guided prediction model. In this step, both deep learning models are optimised based on the loss calculated from the differences between the predicted MLC maps and beam intensities and the ground truth treatment plan.

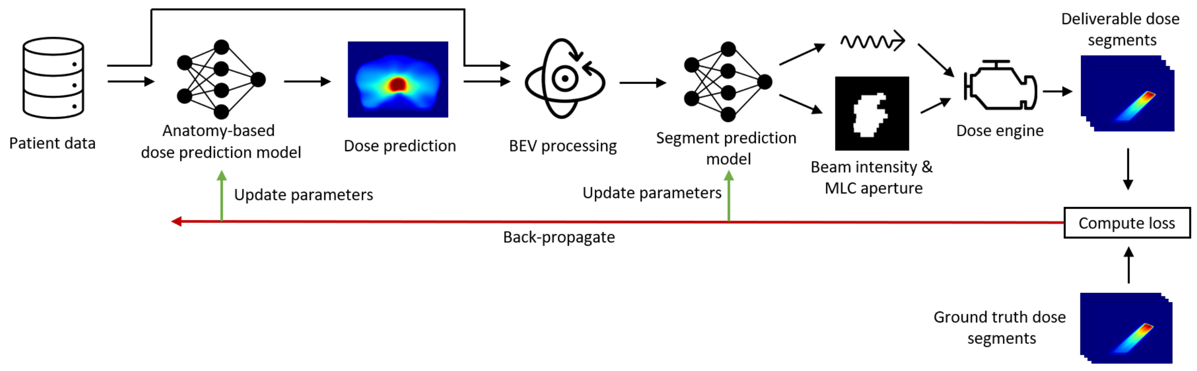


Figure 3.25: Schematic of the third training step of the Physics-guided prediction model. In this step, again both deep learning models are optimised, but now based on the difference between the ground truth and predicted per-segment dose distributions. This loss is backpropagated through the entire model.

Training with Pre-trained Models

Another training method that is investigated is using pre-trained models. With this type of training, the Anatomy-based dose prediction model and the segment prediction model are first trained by themselves as described in Sections 3.3 and 3.4. After pre-training, the models are combined with the dose engine to form the model described in this section. Now the last training is done where only the third training step as described in Figure 3.25. The reason for this is that the deep learning models have already trained on their own, meaning that high-quality dose and segment predictions should already occur. After the validation loss stops decreasing, the training is stopped, and the resulting model can be tested using the test subset.

Model Evaluation
For this model, both the predicted dose and the treatment plan should be evaluated. This can simply be done by combining the evaluation methods used in Sections 3.3 and 3.4. This means that for the evaluation of the MLC maps, the MALD is used, while for the assessment of the predicted beam intensities, the MAE is used.

For the evaluation of the predicted deliverable dose, individual dose slices are inspected, as well as the DVHs. Furthermore, a comparison between the dose coverage statistics is made. For these comparisons, it would not be useful to compare the outcome of the model to a Pinnacle ground truth. The reason for this is that even if perfect segments are predicted, there will still be a significant difference between the ground truth of Pinnacle and the dose produced by the Matrad dose engine. Therefore, a slight alteration should be done where instead the outcome of the model is compared to the Matrad ground truth, which is the dose calculated with the Matrad dose engine from the ground truth treatment plan. This way, it will actually be possible for the model to achieve the ground truth dose distribution. The downside of this is, of course, that the ground truth might no longer be a good quality dose. Still, if the model can produce very similar dose distributions, it proves the effectiveness of the model.

3.7. Dose Mimicking Model

From literature, it is already established that some form of the Anatomy-based dose prediction should work quite accurately. However, there is not yet much research done on a segment prediction model, which means that the success of this secondary deep learning model is uncertain. As the Physics-guided prediction model depends on the success of the segment predictor, it could be the case that this model proves unsuccessful. Therefore, a second model has been designed, which can serve as an alternative to the Physics-guided prediction model without the need for a segment prediction neural network.

In this second model, the only deep learning part that is incorporated is the Anatomy-based dose prediction model. The goal of this deep learning model is to predict a dose distribution as accurately as possible. As discussed, the issue with these predictions is that there is no guarantee that they are deliverable. Therefore, a second step is added, where a dose mimicking optimisation algorithm is used to mimic the predicted dose as close as possible so that the dose is ensured to be deliverable. Section 2.2.2 already gives a short description of the general mechanics behind a dose mimicking algorithm.

In the implemented model, an input of patient data and initialised bixel weights and beam intensities is used. The patient data is used as input to the neural network to predict a dose distribution, while the bixel weights and beam intensities are used to create a first attempt of a mimicked dose. The first attempt will not be an acceptable estimation, and a few optimisation iterations are needed in which the bixel weights and beam intensities are optimised. As soon as the mimicked dose is deemed acceptable, the optimised bixel weights are post-processed so that they can be translated to realistic MLC apertures. Then, another round of optimisation is done, where this time, the bixel weights are kept fixed, and only the beam intensities are optimised until improvements on the mimicked dose can no longer be made. The resulting mimicked dose, together with the corresponding treatment plan parameters, are considered the output of the model. Figure 3.26 shows an overview of this model.

The advantage of this second model over the first one is that fewer steps are involved, which means that it is easier to implement, and less vulnerable to individual failing components. The downside, however, is that this does include multiple optimisation steps. The reason for the involvement of deep learning in treatment planning is that optimisation algorithms, which are often time-intensive, are no longer needed. With this model, that goal is only partially achieved.

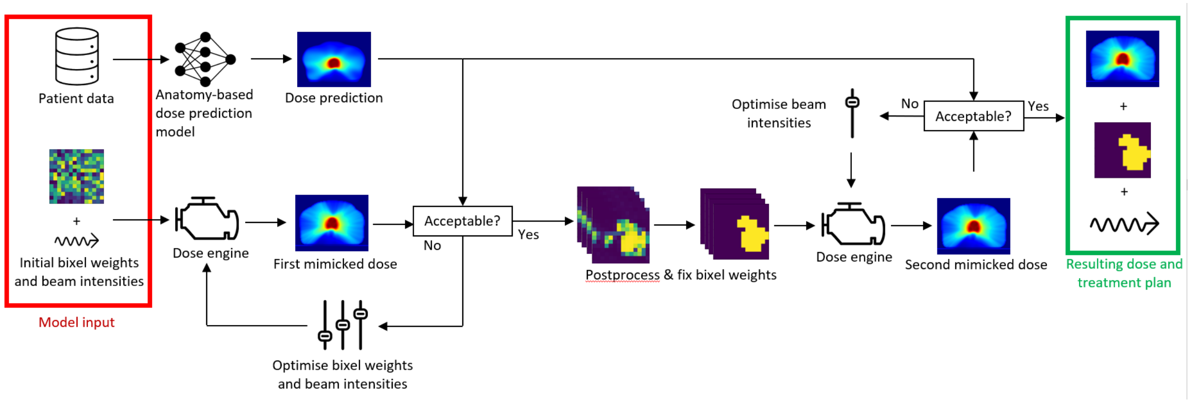

Patient data
Anatomy-based dose prediction model
Dose prediction
Optimise beam intensities
No
Acceptable?
Yes
+
+
Initial bixel weights and beam intensities
Dose engine
First mimicked dose
Acceptable?
Yes
No
Postprocess & fix bixel weights
Dose engine
Second mimicked dose
Resulting dose and treatment plan
Model input
Optimise bixel weights and beam intensities

Figure 3.26: Overview of the Dose mimicking model. A dose mimicking algorithm is used together with a dose prediction neural network. The goal of the mimicking algorithm is to mimic the predicted dose as closely as possible, resulting in a mimicked dose with corresponding treatment plan parameters as output to the model.

4 Results

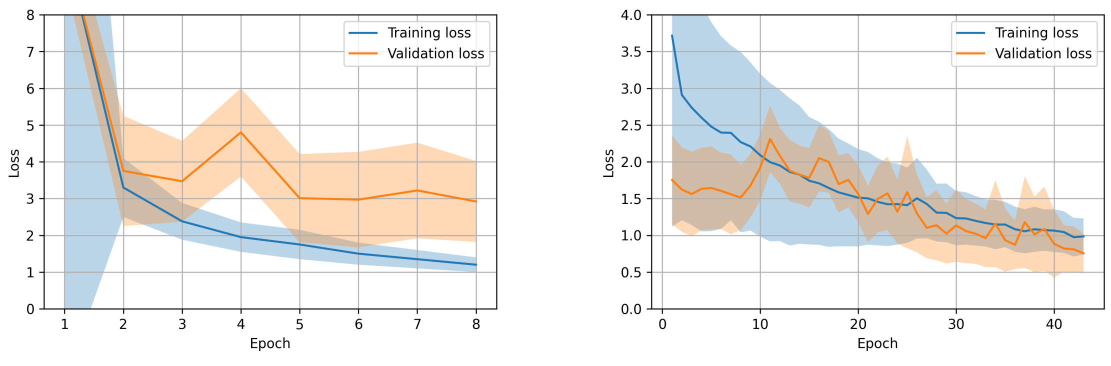
In this chapter, the most important results that have been obtained throughout the thesis are presented. Results are given for all the models and components mentioned in Chapter 3. The main goal of this chapter is to assess the different models and compare their performance to the results from Meerbothe [7] or other models that are evaluated in this thesis. In Section 4.1, 4.2 and 4.3, the performance of the individual components which are used in the Physics-guided model and the Dose mimicking model are evaluated. In Section 4.4 and 4.5, the performance of these two final models is investigated.

4.1. Anatomy-based Dose Prediction Model Performance

In this section, the performance of the Anatomy-based dose prediction model is evaluated. First, a comparison is made between the two network architectures mentioned in Subsection 3.3.1. This comparison is made using dataset I. Then, the performance difference between the two datasets and the impact of adding CT to the input is presented.

4.1.1. Performance Comparison HD U-net and U-net

For both network architectures, three different data augmentation methods have been tested: using no augmentation, using the multi-phased training augmentation and using randomly selected augmentations (see Section 3.3.4 for more detail). In the case of the U-net architecture, the best results were obtained with the multi-phased training augmentation. In Figure 4.1, the average loss obtained in different epochs is shown for the two training phases.



(a) Training phase 1 **(b)** Training phase 2

Figure 4.1: The training and validation loss during the two training phases of the Anatomy-based dose prediction model with U-net architecture. The shaded areas represent the standard deviation over the patients.

During the first training phase, the model was saved after epoch 8. In the second training phase, the model was saved after epoch 43, at which point the validation loss was equal to 0.755. The training was continued for some time to see if the validation loss would drop below this minimum again but was eventually stopped after epoch 115.

For the HD U-net, the best results were obtained by using the randomly selected augmentation scheme. In Figure 4.2, the average losses during the training of the HD U-net are shown.

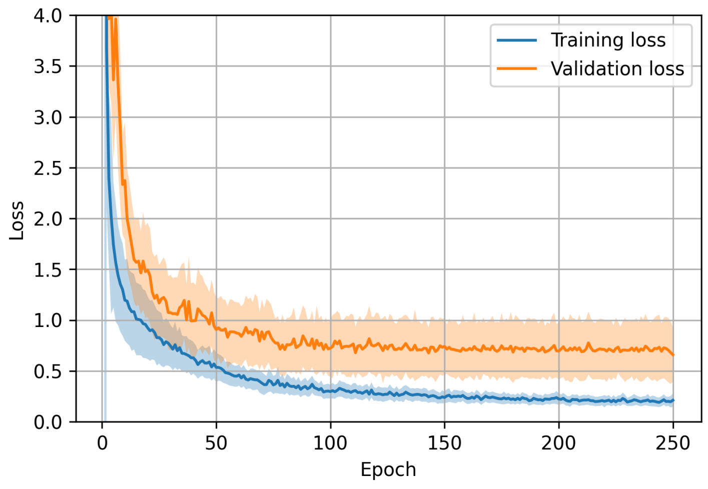


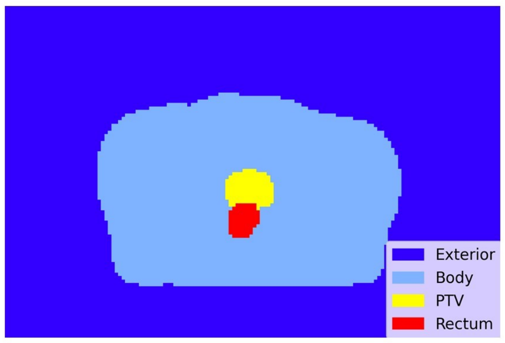
Figure 4.2: The training and validation loss during the training of the Anatomy-based dose prediction model with HD U-net architecture. The shaded areas represent the standard deviation over the patients.

For this architecture, the training was ended at epoch 250, with a validation loss of 0.659, which is slightly lower than the validation loss of the U-net. It should be noted, however, that the training could have been saved earlier, with a very similar loss value. Even though a slightly lower loss value is obtained here, it takes a bit longer to reach this value, especially considering the fact that each epoch takes five times as long due to the augmentation scheme used here. Furthermore, it can be seen that slight overfitting has occurred in this training, as the validation loss stays above the training loss during the entire training in this case. Lastly, the training process is much more stable when compared to the non-augmented training, as there is much less volatility visible in the graphs.

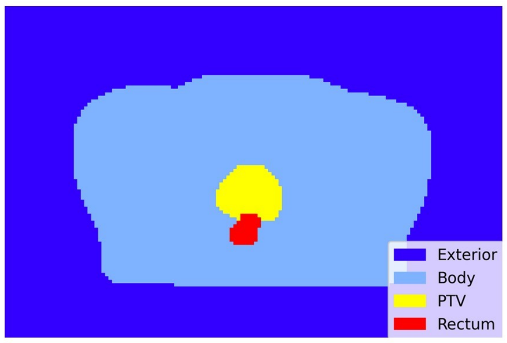
To assess the performance of the models, the test subset of 12 patients was used to generate predictions with the trained models. Figure 4.3 shows an example slice of the predictions and the ground truths of both the best and worst-performing patient in the test subset. A patient's performance is decided upon with the use of the WMSE loss function.

A few things can be noted from this visual inspection. First of all, the shape of the high dose area seems to be quite accurately predicted in both cases. Secondly, it is interesting to see that the U-net is not capable of predicting the treatment table, while the HD U-net does have a reasonable prediction of this. Also, the HD U-net prediction seems to show some more high-frequency details in the prediction compared to the U-net prediction, which is much more uniform. The last point that can be made is that in both models, the ray effects get lost in the prediction, as the points of entry of the photon beams visible in the ground truths are no longer seen in the two predictions. This is not unexpected for the Anatomy-based dose prediction model and confirms that models like these produce dose distributions for which the deliverability is questionable at the least.

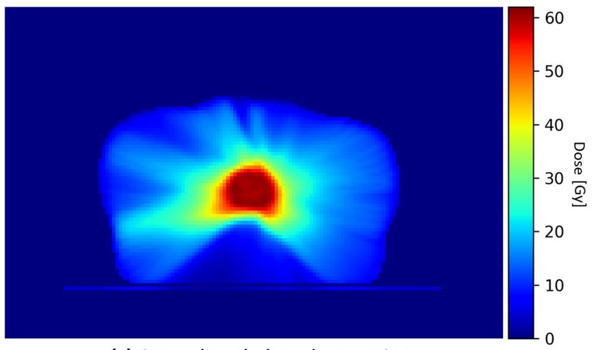
Dose Volume Histogram
Figure 4.4 shows the DVHs of the different structures for the best and worst-performing test cases. From the figure can be seen that all lines seem to follow approximately the same curve, showing good accuracy in all cases, even for the worst-performing patient.



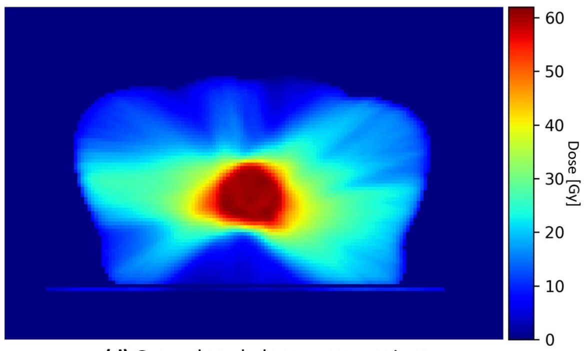
(a) Structure labels, best patient



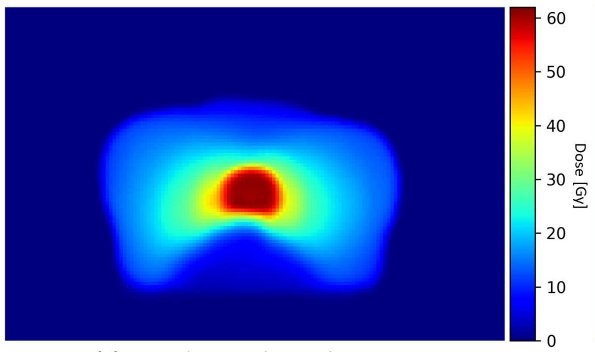
(b) Structure labels, worst patient



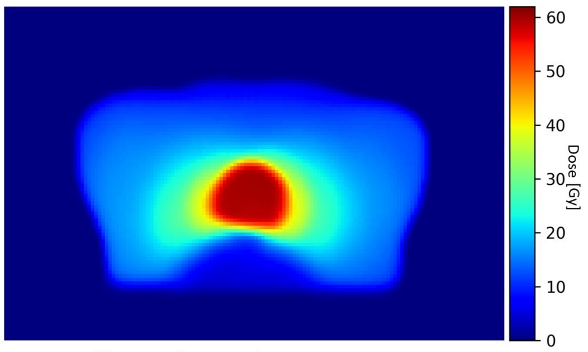
(c) Ground truth dose, best patient



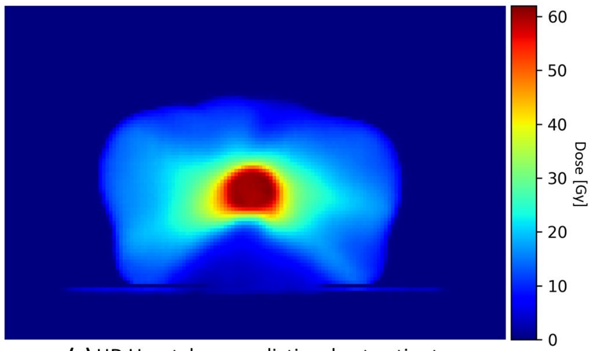
(d) Ground truth dose, worst patient



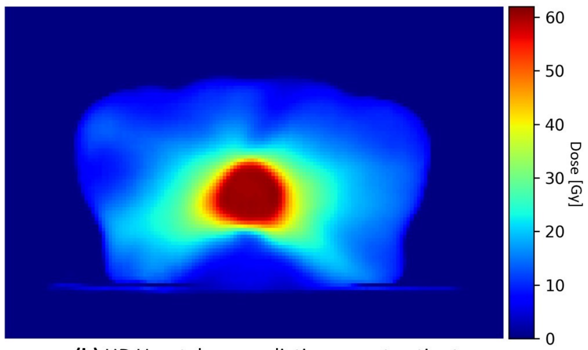
(e) U-net dose prediction, best patient



(f) U-net dose prediction, worst patient



(g) HD U-net dose prediction, best patient



(h) HD U-net dose prediction, worst patient

Figure 4.3: Two example slices from the best and worst-performing test set patients. The ground truth, structure labels and predictions from the two Anatomy-based dose prediction model architectures are shown.

To assess all 12 test cases and quantify the difference in the Dose Volume Histograms between prediction and ground truth, the MAE (see Equation 3.3) is calculated for each point on the DVH. Note that this difference is expressed in Gy, meaning that it presents the horizontal difference visible in the DVH. These values are then averaged over all test cases, allowing a direct numerical comparison between the network architectures and the results from Meerbothe. The results of these calculations are shown in Table 4.1. As can be seen, both the HD U-net and the U-net improve for all structures on the results from Meerbothe and have difference values that are very close to each other and below the 1 Gy. Even though the differences between the two architectures are minor, the HD U-net performs slightly better than the U-net.

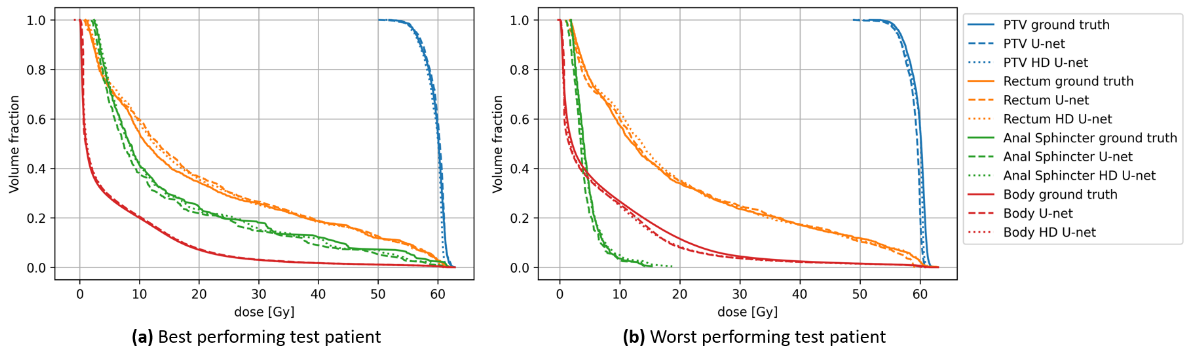


Figure 4.4: Dose Volume Histograms for the different structures, made from the ground truths and the predictions from the Anatomy-based dose prediction model with the U-net and HD U-net architecture.

Table 4.1: Mean Absolute Error of the DVH curves, averaged over all 12 test patients, for the Anatomy-based dose prediction model predictions. The values represent the mean and standard deviation over the different patients and are given in Gy, representing the average horizontal difference between the predicted and ground truth DVH graph.

	Mean Absolute Error DVH [Gy]			
Model	PTV	Rectum	Anal Sphincter	Body
	Mean ± std	Mean ± std	Mean ± std	Mean ± std
U-net	0.38 ± 0.15	0.86 ± 0.66	0.65 ± 0.54	0.37 ± 0.36
HD U-net	0.34 ± 0.18	0.67 ± 0.57	0.48 ± 0.57	0.33 ± 0.47
Results Meerbothe [7]	0.84 ± 0.40	0.91 ± 0.72	0.97 ± 0.69	0.43 ± 0.52

Dose Coverage Statistics

The dose coverage statistics as described in Subsection 3.3.4 are assessed and compared to the results from Meerbothe. Specifically, coverage statistics on the Rectum and PTV are investigated, as the dose coverage in these structures is the most important. The goal is to obtain values for these statistics as close as possible to the ground truth. Table 4.2 shows the differences for the different models as percentages of the prescribed dose (which is 60 Gy) for the PTV. Table 4.3 shows the relative dose differences for various coverage statistics of the Rectum.

For the PTV coverage, the numbers show that both the U-net and HD U-net are again performing better than the results from Meerbothe, achieving less than 1% difference on nearly all statistics. The HD U-net seems to perform slightly worse than the U-net and this case, but again, the differences are small.

For the rectum coverage, differences are a bit higher, but this can be expected as the dose is less uniform in this structure. In this structure, the three results are comparable, with the HD U-net performing best and the U-net closely behind.

To properly evaluate the spread of the dose coverage statistics over the different test patients, a box plot is also made, comparing the percentage differences between prediction and ground truth for both the U-net and HD U-net. The box plot is shown in Figure 4.5. Note that the absolute value of the difference is no longer taken in this case, which means that a distinction can be made between an over- and underprediction. The U-net seems to perform slightly better on the PTV, while the HD U-net performs better on the rectum, even though the differences are again marginal. Furthermore, it can be seen that the spread of difference values is generally lower for the HD U-net. A last point that can be made is that both architectures seem to have a bias towards making under-predictions, as the dose difference percentages are more often positive than negative.

Table 4.2: Absolute dose difference as a percentage of the prescribed dose between the ground truth and the prediction of the Anatomy-based dose prediction model for four different coverage statistics on the PTV. The numbers represent averages and standard deviations over the 12 test patients.

	Relative dose difference [%]: $100\% \cdot \frac{D_{\text{gt}} - D_{\text{pred}}}{60\ Gy}$			
	PTV coverage statistics			
Model	D_{95}	D_{98}	D_{max}	D_{mean}
	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std
U-net	0.54 ± 0.23	0.71 ± 0.40	0.63 ± 0.54	0.61 ± 0.27
HD U-net	0.78 ± 0.33	0.78 ± 0.33	1.06 ± 0.39	0.55 ± 0.22
Results Meerbothe [7]	1.42 ± 1.26	1.35 ± 1.47	2.43 ± 1.01	0.99 ± 0.67

Table 4.3: The relative absolute dose difference between the ground truth and the prediction of the Anatomy-based dose prediction model for three different coverage statistics on the rectum. The numbers represent averages and standard deviations over the 12 test patients.

	Relative dose difference [%]: $100\% \cdot \frac{D_{\text{gt}} - D_{\text{pred}}}{D_{\text{gt}}}$		
	Rectum coverage statistics		
Model	D_{max}	D_{mean}	V_{45}
	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std
U-net	1.51 ± 0.72	2.74 ± 2.16	2.25 ± 1.74
HD U-net	0.67 ± 0.63	2.60 ± 1.48	2.72 ± 2.08
Results Meerbothe [7]	1.81 ± 0.99	3.39 ± 2.47	3.50 ± 2.47

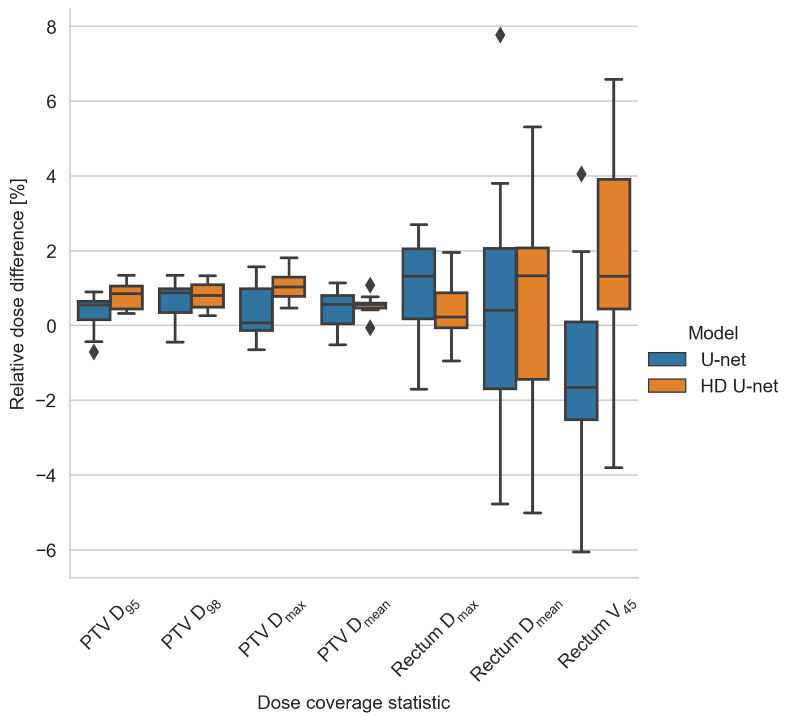


Figure 4.5: Box plot of the relative difference in the dose coverage statistics between the ground truth and the prediction made with the Anatomy-based dose prediction model.

Sørensen-Dice Index
The SDI is computed for 100 different isodose contours, resulting in Figure 4.6. In general, the index remains very close to one, which means that the isodose contours of the prediction overlap well with the ground truth isodose contours. Furthermore, the standard deviation is minimal, showing a consistent

performance on this metric over all test patients. It is interesting to see that there is a slight dip in performance around the 30% isodose contour.

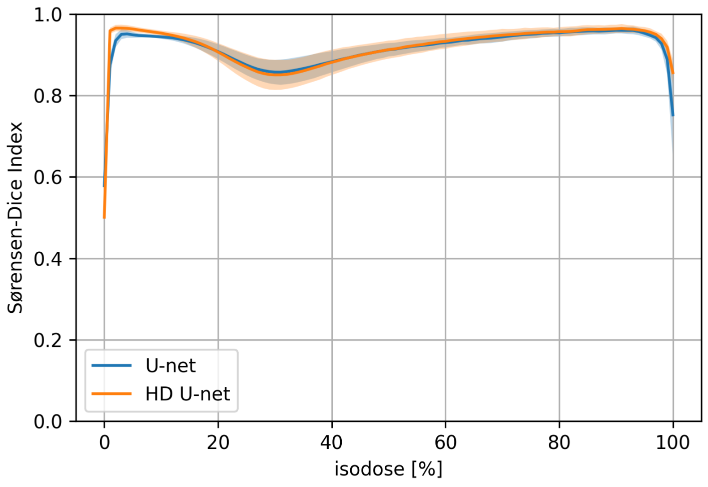


Figure 4.6: The Sørensen-Dice index for different isodose contour overlap between the ground truth and the prediction made with the Anatomy-based dose prediction model, averaged over all test patients. The shaded area represents the standard deviation.

To investigate this dip, the 30% isodose contour for an example slice is shown for both the predicted and ground truth dose in Figure 4.7. From this figure can be seen why this dip is visible in Figure 4.6. It seems that at this dose level, ray effects of the incoming radiation beams are affecting the contour, which is visible on the ground truth dose. As the prediction cannot adequately predict the ray effects, the contour looks much more round and uniform here. This detail shows that the 30% isodose contour seems to be a good metric to evaluate how well the ray effects are taken into account in a prediction.

Overall, the two architectures perform very comparably. On average, both the U-net and HD U-net have an overall Sørensen-Dice index of 0.92. In the results of Meerbothe, an overall index score of 0.91 was found.

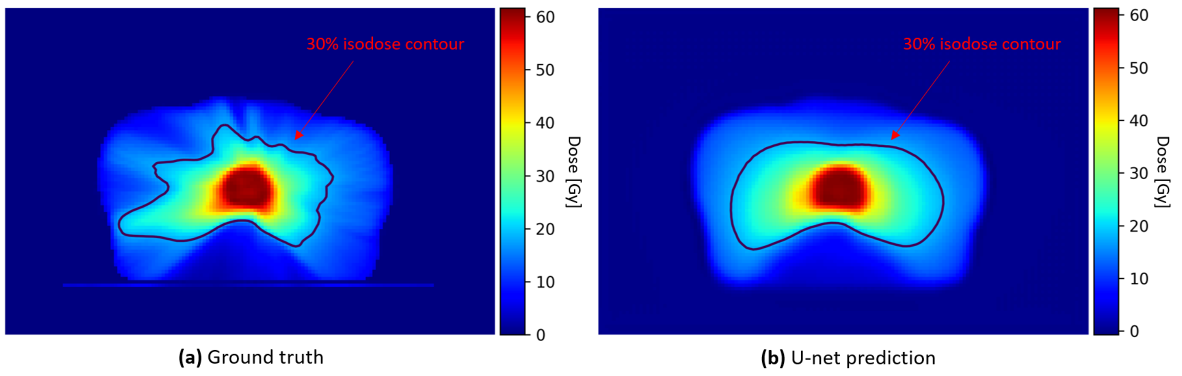
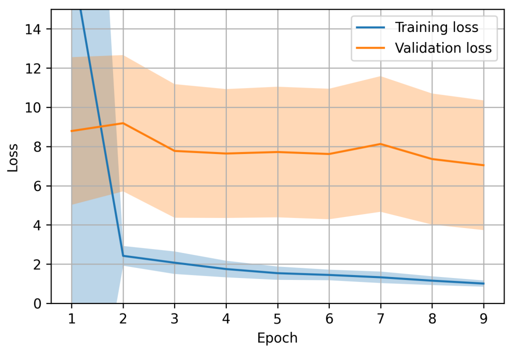


Figure 4.7: Example slice of a ground truth and predicted dose distribution made with the Anatomy-based dose prediction model and the U-net architecture, for a patient from the test subset.

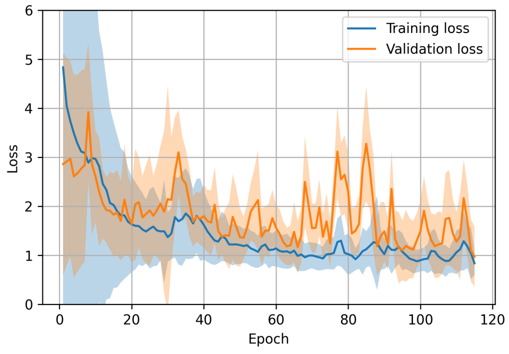
4.1.2. Improvement on Dose Prediction with CT

Apart from the more intricate models mentioned in Sections 3.6 and 3.7, the dose prediction could potentially be improved by adding the patient CT data into an extra channel of the input data. With the CT, the model has more information on the patient's anatomy compared to only using the structure masks. The issue is that there is no CT data available in dataset I, which means that dataset II needs to be used for training with CT data. As a different dataset can already have a significant impact on the performance of a neural network, first another training is done on this new dataset without using CT data, and this trained model is then compared to the results from the previous subsection. This way, it can be certain that any change in performance is caused by adding the CT data as input. During the last subsection, it became clear that the performance difference between the U-net as HD U-net is very small. As the U-net can be trained much faster due to the different augmentation scheme used, it will be used from this point onward as the main architecture for the Anatomy-based dose prediction model.

Figure 4.8 shows the loss curves of the U-net training on dataset II without using CT data. The multi-phased augmentation scheme has been used for the training. During the first training phase, the model is saved at epoch 9 with a validation loss of 7.04, while in the second training phase, the model is saved at epoch 115 with a validation loss of 0.97. When comparing this loss curve to the U-net training on dataset I, as seen in Figure 4.1, a few things already become evident. First of all, it takes a lot longer for both the training loss and the validation loss to get to a loss value that is below one. This can be expected as dataset II is both larger and more versatile in patient anatomy. Furthermore, The training curve in the second training phase seems to be much more volatile, both in the validation and training loss. Again, this could be caused by the larger dataset as it struggles to generalise to all the patients. This could also be the reason that, in the end, the final validation loss is a bit higher on dataset II than on dataset I.



(a) Training phase 1



(b) Training phase 2

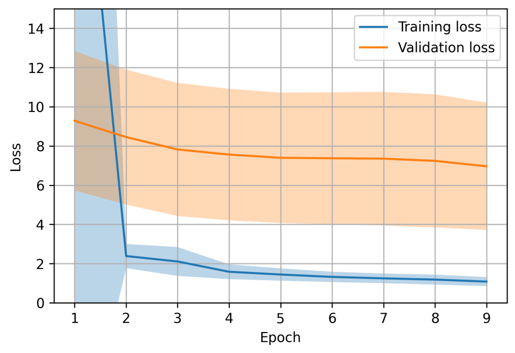
Figure 4.8: The training and validation loss during the two training phases of the Anatomy-based dose prediction model, with the U-net architecture, on dataset II. The shaded areas represent the standard deviation over the patients.

Next, a training is done on dataset II, now including the CT data as input. The resulting training curves are shown in Figure 4.9. For this training, the model is saved in the first training phase after nine epochs, with a validation loss of 6.96, and in the second training phase, the model is saved after 141 epochs with a validation loss of 0.98. The curves are very similar to the loss curves without CT input, the only real difference being that a slightly longer training is needed to arrive at the same validation loss. This could be explained by the fact that the model now has more data for which it has to learn how to use it. To visually inspect the differences between individual slices of the dose prediction, see Figure A.1 in Appendix A.

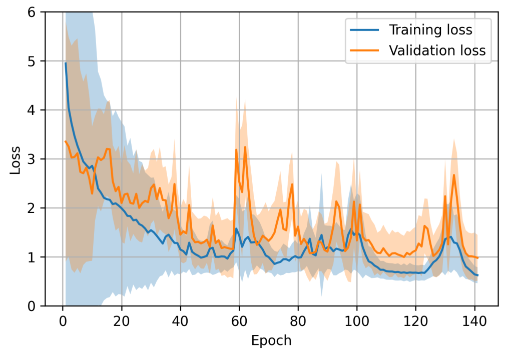
Dose Volume Histogram

A DVH plot is used to assess the differences in the prediction without CT data and the prediction with CT data. Figure 4.10 shows both a good-performing test case and a bad-performing test case for both models. From the DVH plots can be seen that there seems to be little difference in performance, especially on the good-performing test case. On the bad-performing test case, the model with CT input appears to do slightly better on the rectum dose, while both models struggle to accurately predict the PTV coverage.

To quantitatively assess the differences in all DVH plots, the mean absolute DVH difference is used. Table 4.4 shows these differences between prediction and ground truth for the different structures. Here, a comparison is also made with the performance of the U-net on dataset I. From this quantitative assessment can be seen that the model trained on dataset II is producing slightly worse predictions for the test cases. Furthermore, no significant differences are seen between a training where CT is used and a training where no CT is used. The only difference is that the training without CT data performs slightly better on the PTV, while the training with CT data performs somewhat better on the rectum.

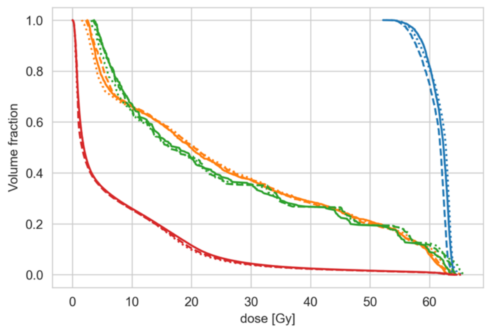


(a) Training phase 1

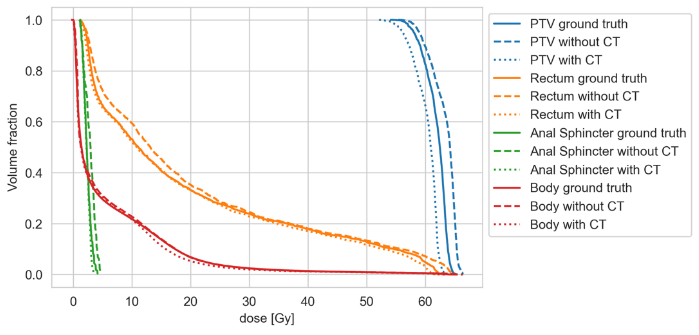


(b) Training phase 2

Figure 4.9: The training and validation loss during the two training phases of the Anatomy-based dose prediction model, with U-net architecture, training on dataset II with the CT data as added input. The shaded areas represent the standard deviation over the patients.



(a) Best performing test patient



(b) Worst performing test patient

Figure 4.10: Dose Volume Histograms for the different structures, made from the ground truths and the predictions from the Anatomy-based dose prediction model, trained on dataset II.

Table 4.4: Mean Absolute Error of the DVH curves, averaged over all 12 or 16 test patients (depending on the dataset), for the Anatomy-based dose prediction model predictions. The values represent the mean and standard deviation over the different patients and are given in Gy, representing the average horizontal difference between the predicted and ground truth DVH graph.

	Mean Absolute Error DVH [Gy]			
Model	PTV	Rectum	Anal Sphincter	Body
	Mean ± std	Mean ± std	Mean ± std	Mean ± std
Dataset II without CT	0.80 ± 0.28	1.30 ± 0.88	0.87 ± 0.85	0.39 ± 0.39
Dataset II with CT	1.23 ± 0.29	0.98 ± 0.72	0.80 ± 0.80	0.44 ± 0.62
Dataset I without CT	0.38 ± 0.15	0.86 ± 0.66	0.65 ± 0.54	0.37 ± 0.36

Dose Coverage Statistics

Tables 4.5 and 4.6 show the results of the dose coverage statistics assessment for the PTV and the rectum. As was seen in the mean absolute DVH differences, the models trained on dataset II seem to perform slightly worse. Again, the model with CT input does not seem to improve the performance at all, and on the PTV, performance appears to be marginally worse.

Table 4.5: Absolute dose difference as a percentage of the prescribed dose between the ground truth and the prediction of the Anatomy-based dose prediction model for four different coverage statistics on the PTV. The numbers represent averages and standard deviations over the 12 or 16 test patients (depending on the dataset).

	Absolute dose difference [%]: $100\% \cdot \left\|\frac{D_{\text{gt}} - D_{\text{pred}}}{60\ Gy}\right\|$			
	PTV coverage statistics			
Model	D_{95}	D_{98}	D_{max}	D_{mean}
	Mean ± std	Mean ± std	Mean ± std	Mean ± std
Dataset II without CT	1.23 ± 1.06	1.41 ± 1.38	1.57 ± 0.92	1.30 ± 0.78
Dataset II with CT	2.57 ± 1.28	2.77 ± 1.74	1.65 ± 1.41	1.98 ± 1.38
Dataset I without CT	0.54 ± 0.23	0.71 ± 0.40	0.63 ± 0.54	0.61 ± 0.27

Table 4.6: The relative absolute dose difference between the ground truth and the prediction of the Anatomy-based dose prediction model for three different coverage statistics on the rectum. The numbers represent averages and standard deviations over the 12 or 16 test patients (depending on the dataset).

	Absolute dose difference [%]: $100\% \cdot \left\|\frac{D_{\text{gt}} - D_{\text{pred}}}{D_{\text{gt}}}\right\|$		
	Rectum coverage statistics		
Model	D_{max}	D_{mean}	V_{45}
	Mean ± std	Mean ± std	Mean ± std
Dataset II without CT	1.43 ± 1.06	5.30 ± 4.20	4.52 ± 3.86
Dataset II with CT	1.46 ± 1.17	3.23 ± 2.55	4.99 ± 2.54
Dataset I without CT	1.51 ± 0.72	2.74 ± 2.16	2.25 ± 1.74

Sørensen-Dice Index

Figure 4.11 shows a plot of the SDI for the different isodose contours of prediction and ground truth. From the figure can be seen that there is no significant difference between the SDI values and that the 30% isodose dip is still present. Together with all the other results obtained in this part, it can be concluded that dataset II performs slightly worse on the Anatomy-based dose prediction model. Furthermore, adding the CT data as an extra input channel to the model does not improve performance.

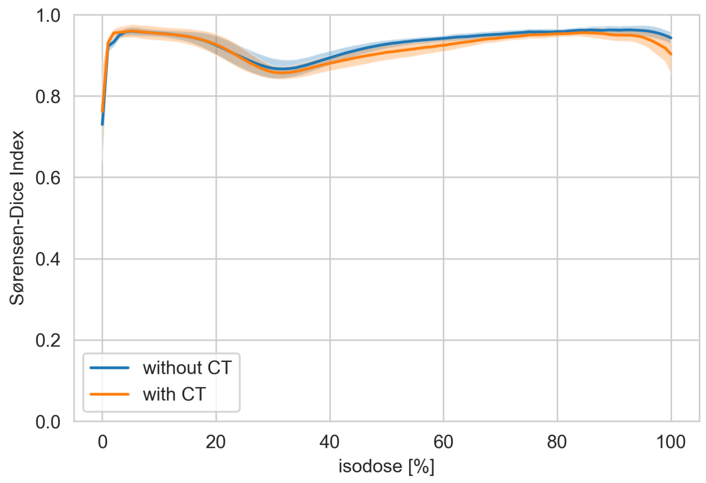


Figure 4.11: The Sørensen-Dice index for different isodose contour overlap between the ground truth and the prediction made with the Anatomy-based dose prediction model, averaged over all test patients. The shaded area represents the standard deviation.

4.2. Segment Prediction Model Performance

In this section, the performance of the Segment prediction model, as described in Section 3.4, is evaluated. In Figure 4.12, the average loss for the training and validation set during the training process is shown. The model is saved at epoch 51, after which the validation loss starts to increase. The curve indicates that the validation loss quickly levels out to a specific value, and the training loss slowly follows. Only when the training loss decreases further, does overfitting happen, which is why the model is stopped just before this moment. Note that the loss is a combination of the BCE loss on the MLC output and the MSE loss on the MU output.

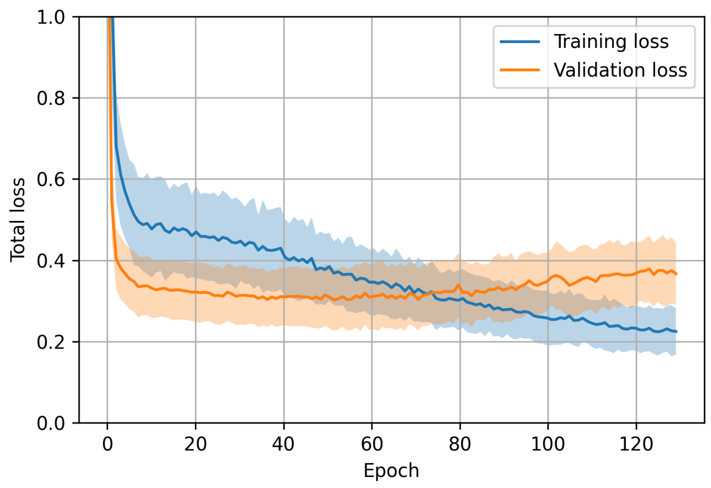


Figure 4.12: The training and validation loss during the Segment prediction model training. The model is saved at epoch 51, just before the validation loss starts to increase. The shaded areas represent the standard deviation over the patients.

4.2.1. Evaluation of MLC Predictions

As the training is done on dataset II, 16 test patients were used to evaluate the performance of the trained model. For the evaluation of the MLC prediction, three overlays of ground truth and predicted MLC shape are shown in Figure 4.13, showing the best-performing, worst-performing and most average performing segment in the test set. Also, the contour of the PTV from the corresponding BEV is shown in these figures. From the three figures can be seen that there is a lot of spread in the quality of the MLC predictions. The best prediction is of very high quality, showing nearly no difference with the ground truth. On the other hand, the worst-performing segment is also completely wrong. The shape of the average performing segment is similar to the ground truth but slightly too large in this case.

From the PTV contours can be seen that a good MLC prediction is obtained when the ground truth MLC follows the PTV contour closely. When this is not the case, worse predictions are obtained. This is especially clear in the worst-case example, where the prediction still follows the PTV contour closely with the MLC shape, while this is not the case for the ground truth.

Figure 4.14 shows the spread of the MALD over the test cases for all the segments. There seems to be some variance in both the average and the standard deviation of the per-segment MALD. This means that there are consistently specific segments for which the model has trouble predicting good MLC shapes. The overall average of the MALD for this model is 3.22 mm ± 1.65 mm, meaning that on average, the leaves are estimated 3.22 mm away from the correct location. However, some heavy outliers are present, as is shown in the worst-performing segment figure, which has a much higher MALD value.

4.2.2. Evaluation of Beam Intensity Predictions

In Figure 4.15 the beam intensities for the best and worst-performing test patients are shown. In both cases can be seen that the prediction of these intensities is not very accurate. The ground truth seems to have constant MU values, with sudden, large dips at specific segments. The predicted MU values are constantly changing over the segments, which means that the behaviour is different from the ground truth. In the best case, the continually changing curve of the predicted beam intensities tries to follow the ground truth curve a bit, but in the worst case, where the ground truth curve also fluctuates heavily at specific groups of segments, the prediction can not keep up.

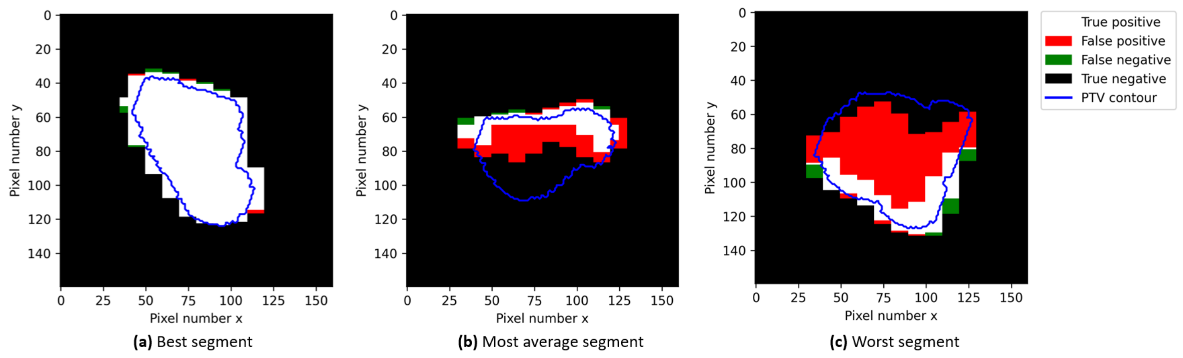


Figure 4.13: Ground truth and predicted MLC overlay of three segments in the test subset, with the PTV contour overlayed. The predictions are made with the Segment prediction model. The left figure has a MALD of 0.4 mm, the middle figure has a MALD of 3.0 mm, while the right figure has a MALD of 11.6 mm.

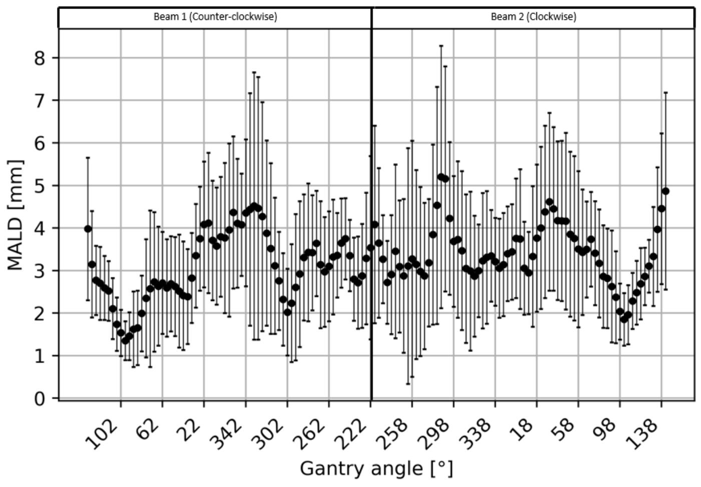
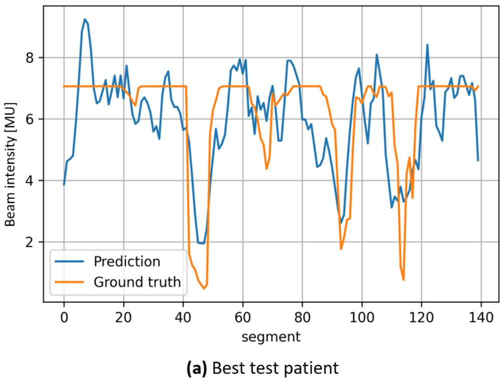


Figure 4.14: The per-segment Mean Absolute Leaf Difference (MALD) in mm of the Segment prediction model, averaged over the 16 test patients. The error bars represent the standard deviation.

Overall, a Mean Absolute Error of 1.39 MU ± 1.18 MU over all segments and all test patients was found. Figure 4.16 shows how this error is spread over the different segments. In this case, the spread in MAE between patients seems to stay more constant over the segments. The average MAE, however, also varies over the segments, with a high peak in the last few segments.



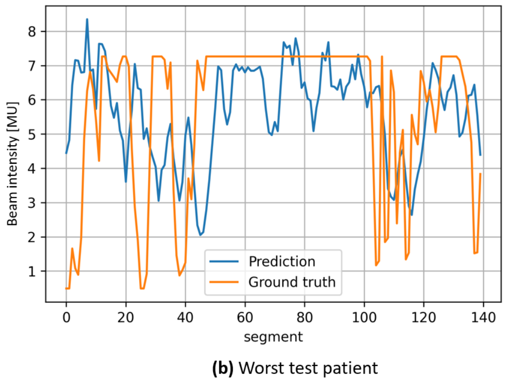


Figure 4.15: The beam intensities in Monitoring Units of all 140 segments of the Segment prediction model prediction and ground truth of both the best and worst-performing test patient.

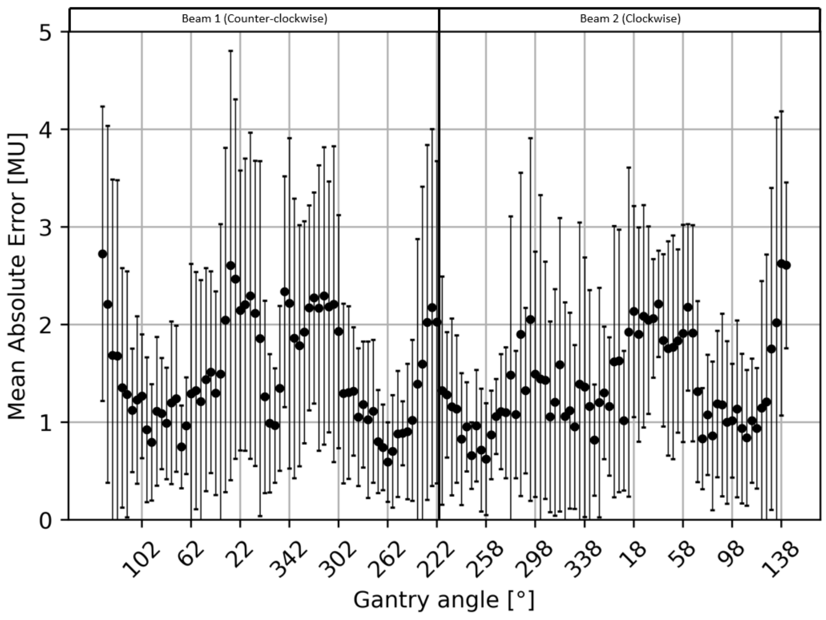


Figure 4.16: Per-segment mean absolute error in Monitoring Units of the beam intensities for the Segment prediction model. The error bars represent the stand deviation over the 16 test patients.

4.3. Dose Engine Performance

In Figure 4.17, an example slice of both the Pinnacle dose and the Matrad dose is shown for the best and worst-performing patients from dataset II. What is immediately noticeable in comparison to dose prediction examples is that now the ray effects are clearly and correctly visible. This is also confirmed by the 30% isodose contours plotted on top of the images, which are now very similar. This verifies that the treatment plan input is correctly used to generate a dose distribution, as similar high dose entry points are visible in the Pinnacle and Matrad dose. This goes both for the worst and best-performing test patient. A difference that can be noticed between the dose engines is that the Pinnacle dose has a smoother distribution. This is visible at the edges, where the individual beamlets are visible in the Matrad dose, whereas the Pinnacle dose is more smooth. Furthermore, the high dose area in the Matrad dose also seems not very uniform, which is an important difference with the Pinnacle dose.

In Figure 4.18, the DVH curves of different structures are also shown, again for the best and worst-performing patients. From the plots can be seen that in the best case, the differences are not very large. For the worst case, however, differences are considerable. An area of concern here is the relatively large volume fraction of the PTV that receives a lower dose than the prescription dose of 60 Gy.

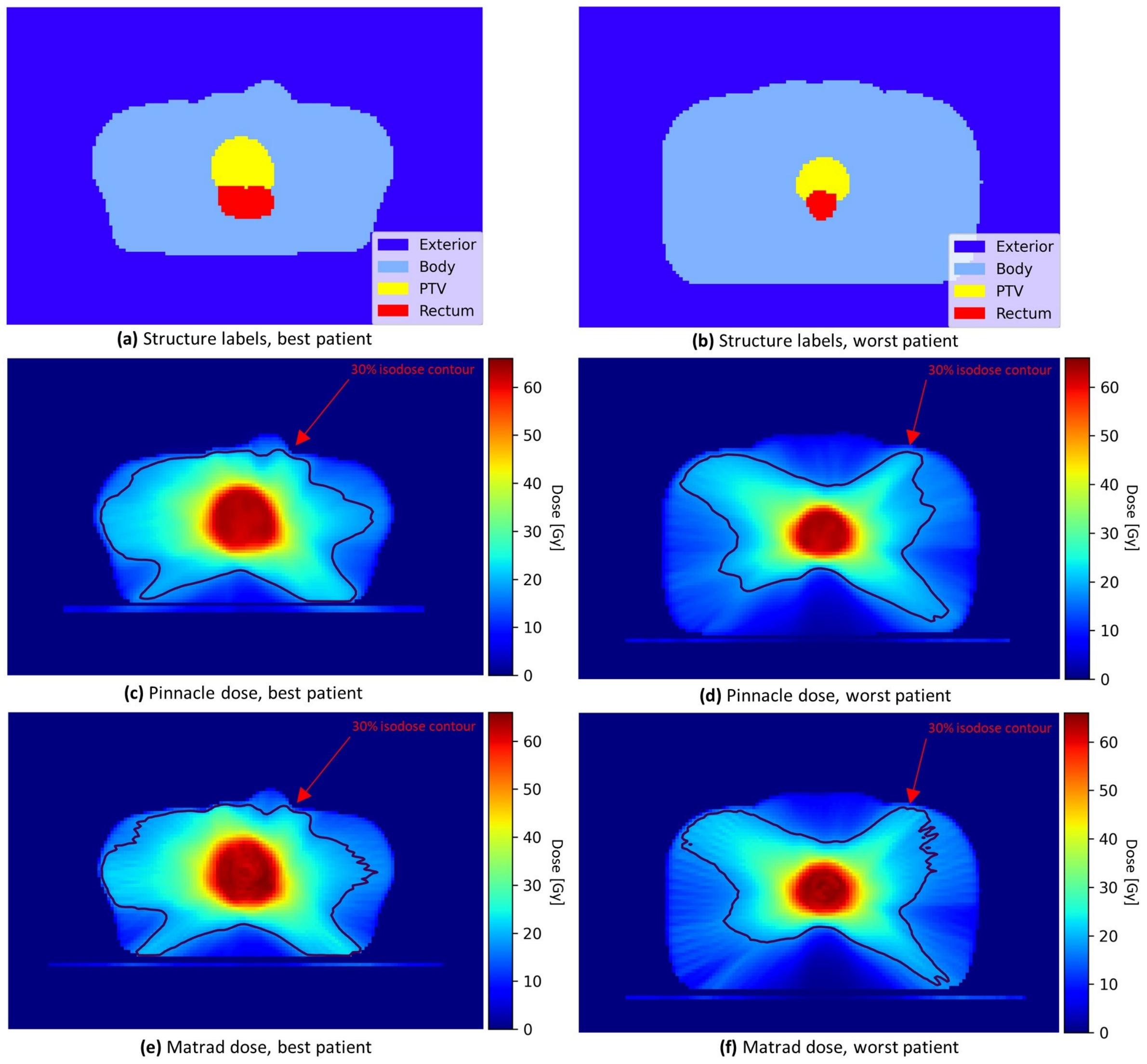

Exterior
Body
PTV
Rectum
(a) Structure labels, best patient
(b) Structure labels, worst patient
30% isodose contour
Dose [Gy]
0
10
20
30
40
50
60
(c) Pinnacle dose, best patient
(d) Pinnacle dose, worst patient
(e) Matrad dose, best patient
(f) Matrad dose, worst patient

Figure 4.17: Example slices of the dose distributions from the Pinnacle and the Matrad dose engine.

Table 4.7 shows the mean absolute difference in Gy between the DVH curves of Pinnacle and Matrad, averaged over all 109 patients in dataset II. Also, the differences between the Pinnacle ground truth and the U-net prediction made with the Anatomy-based dose prediction model, trained on dataset II, are shown here. The difference between the Matrad and Pinnacle dose engines is considerably higher than the error in the prediction, indicating that the difference between the dose engines is significant.

Table 4.7: Mean Absolute Error of the DVH curves, averaged over all 109 patients of dataset II or over the 16 test patients in case of the prediction, for the dose engine performance. The values represent the mean and standard deviation over the different patients and are given in Gy, representing the average horizontal difference between the predicted and ground truth DVH graph.

	Mean Absolute Error DVH [Gy]			
	PTV	Rectum	Anal Sphincter	Body
	Mean ± std	Mean ± std	Mean ± std	Mean ± std
Pinnacle vs Matrad	0.98 ± 1.13	2.40 ± 1.77	2.96 ± 2.98	0.70 ± 0.29
Pinnacle vs U-net prediction	0.80 ± 0.28	1.30 ± 0.88	0.87 ± 0.85	0.39 ± 0.39

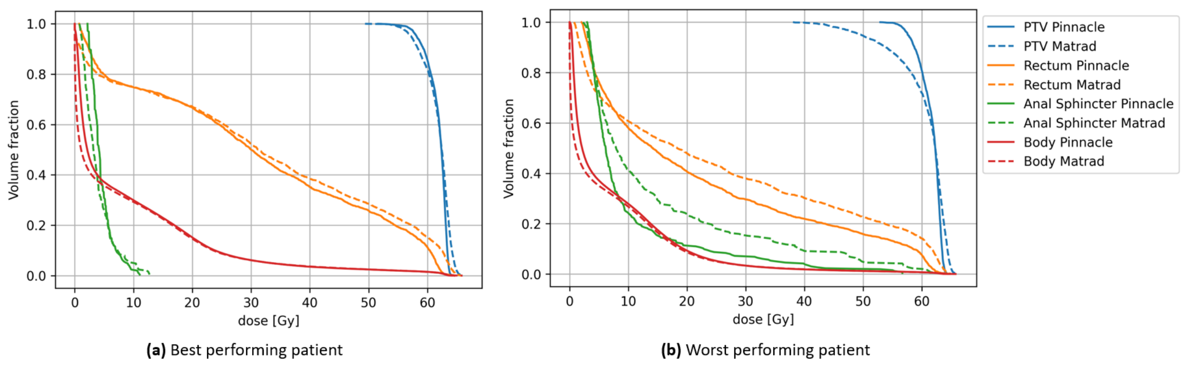


Figure 4.18: Dose Volume Histograms of different structures for both the Pinnacle and the Matrad dose engine.

4.4. Physics-Guided Prediction Model Performance

The Physics-guided prediction model that has been trained obtained the best results when using pre-trained models for both the Segment prediction model and the Anatomy-based dose prediction model. After the pre-training, however, minimal improvements in the loss functions are made, and both the validation and training loss seem to level out quickly. This suggests that minor improvements are made over the separately trained models, which would mean that using the segment and dose predictors on their own with the dose engine should give similar results. Still, using pre-trained models performed better than the training from scratch, as in the training from scratch, it took too long for the Anatomy-based dose prediction model to reach a point where it can actually be helpful for the segment prediction. Also, training the model without the dose BEV as input for the segment predictor was tested, but this too gave slightly worse results than the results found in the instance described here.

4.4.1. Evaluation of the Predicted Segments

A similar evaluation can be done as in Section 4.2, to investigate the quality of the predicted treatment plan. First, in Figure 4.19, the average MALD is plotted for all the different segments. When comparing this to Figure 4.14, it can be seen that there is not much difference in performance. The only difference visible between the two is the slightly different average values in some segments. Overall, an average MALD of 3.18 mm ± 1.67 mm is obtained with this model, which is only a very slight improvement over the 3.22 mm ± 1.65 mm found for the segment prediction training on its own.

For the evaluation of the beam intensity prediction, the MAE is plotted for the different segments in Figure 4.20. Again, very little difference is seen here when comparing these results to Figure 4.16, and the overall average of the MAE is 1.53 MU ± 1.20 MU, which is in this case slightly higher than the 1.39 MU ± 1.18 MU obtained when training the model on its own. These results show that, indeed, the segment prediction has not improved with the use of the calculated dose in the training.

4.4.2. Calculated Dose

Figure 4.21 shows example slices of the dose distribution for the best and worst-performing test patient, obtained with the Physics-guided prediction model. The first thing to notice from these figures is that the 30% isodose contour is not at all as uniform as was the case in the anatomy-based dose prediction in Figure 4.7, showing that indeed more ray effects are present here. In the case of the best-performing example, the shape of the predicted 30% isodose contour is indeed quite similar to the ground truth. In the case of the worst-performing example, however, fewer similarities are visible between the two contours. For this case it can be seen how an opposite gantry angle seems to be favoured more in the prediction compared to the ground truth. As the shape of the dose distribution in the centre still appears to be quite similar, this fact suggests that there are multiple ways of achieving a specific dose. In the worst-case prediction, a different approach might have been taken to generate an equivalent dose in the region close to the isocentre. However, a point of concern in these predictions is the dose value close to the isocentre. Especially for the worst-performing case, the maximum dose value seems to lie much lower in the prediction than in the ground truth, most likely caused by some significant errors in the beam intensity predictions.

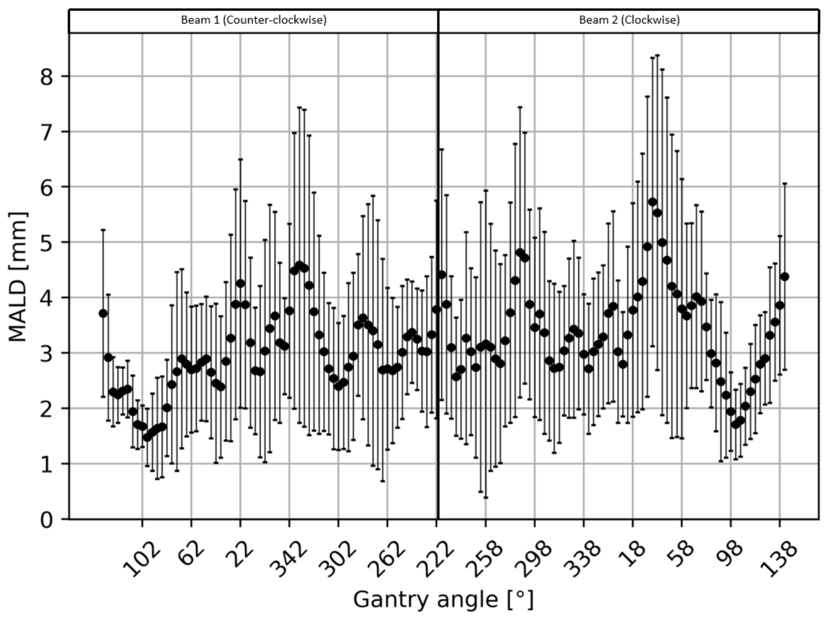


Figure 4.19: The per-segment Mean Absolute Leaf Difference (MALD) in mm, obtained with the Physics-guided prediction model, averaged over the 16 test patients. The error bars represent the standard deviation over the 16 test patients.

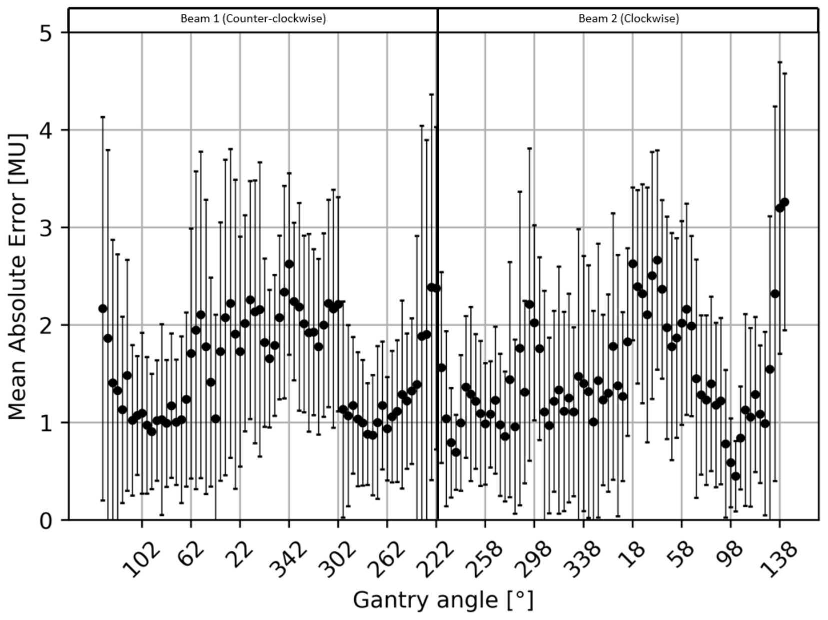
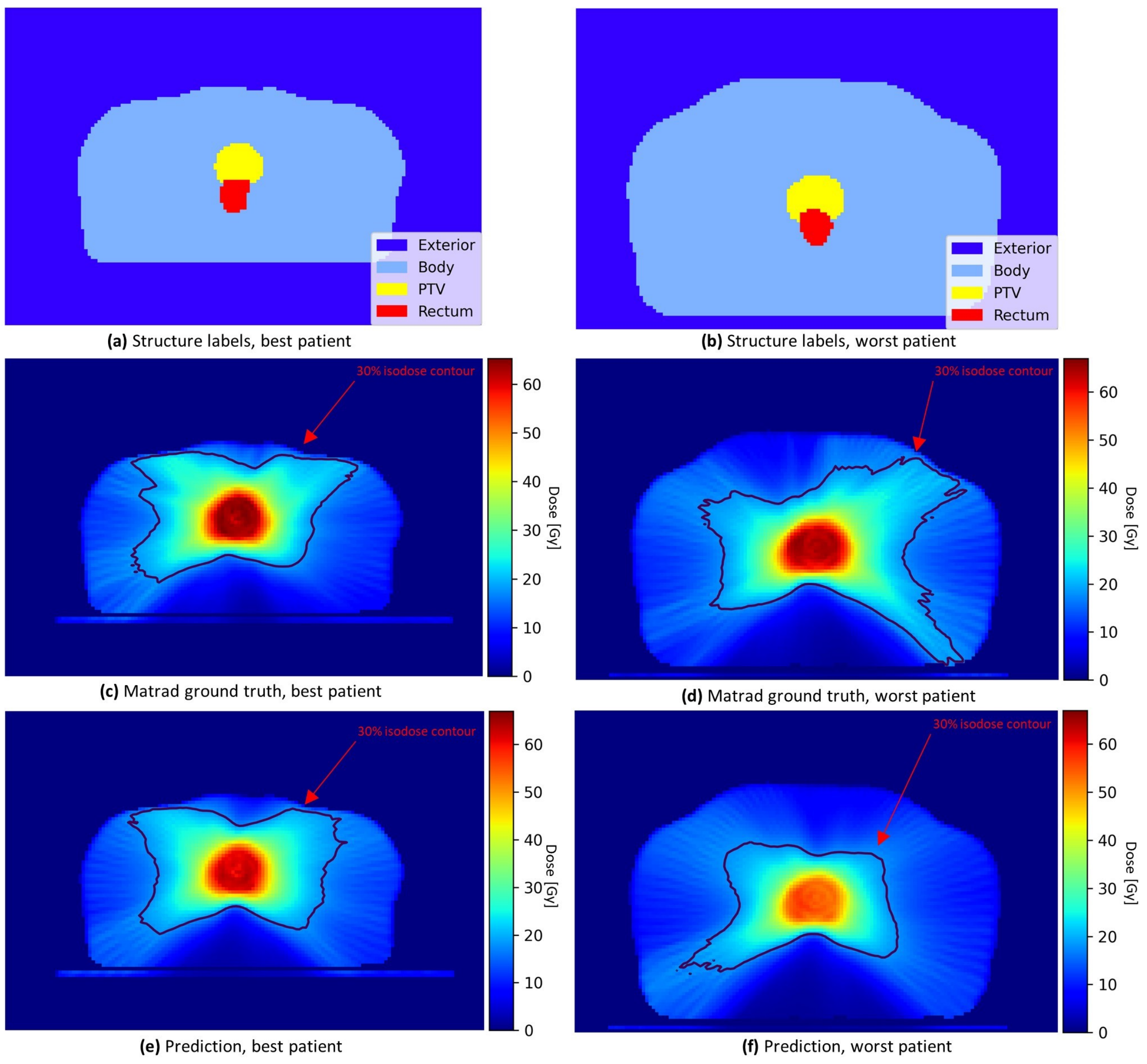


Figure 4.20: The per-segment Mean Absolute Error (MAE) in Monitoring Units of the beam intensities for the different segments, obtained with the Physics-guided prediction model. The error bars represent the standard deviation over the 16 test patients.

To evaluate the dose distribution of the two examples better, the DVH of the different structures is given in Figure 4.22. In the DVH plot, it becomes clear that, especially in the worst case, the predicted dose is quite far from the ground truth dose. This is especially apparent in the DVH curve of the PTV. Furthermore, it should be noted that in this case, both the ground truth and the prediction shown here do not represent clinically acceptable dose distributions, which is due to the different mechanics behind the Matrad dose engine in comparison to the Pinnacle dose engine. The goal here is to show that this model can predict a dose distribution accurately, which has been generated by the same dose engine. From the two DVH graphs can be seen that it succeeds in some cases, but not in all.



Exterior
Body
PTV
Rectum
(a) Structure labels, best patient
(b) Structure labels, worst patient
30% isodose contour
Dose [Gy]
0
10
20
30
40
50
60
(c) Matrad ground truth, best patient
(d) Matrad ground truth, worst patient
(e) Prediction, best patient
(f) Prediction, worst patient

Figure 4.21: Example slices of the structure labels and dose distributions from the ground truth and the Physics-guided prediction model. The best and worst-performing test patients are shown.

To evaluate the accuracy of the DVH curves for all test patients, Table 4.8 shows the mean absolute DVH differences of the predicted dose with the ground truth. Also, the values for the Anatomy-based dose prediction model on dataset II are added for comparison. When measured over all the test patients, the performance of the Physics-guided prediction model is significantly worse on the PTV compared to the Anatomy-based dose prediction model. For the structures outside the PTV, the difference is not as much, but it is still slightly worse. The differences in the dose coverage statistics are also significant, and these results can be found in appendix B.

Table 4.8: Mean Absolute Error of the DVH curves, averaged over the 16 test patients, for the Physics-guided prediction model. The values represent the mean and standard deviation over the different patients and are given in Gy, representing the average horizontal difference between the predicted and ground truth DVH graph.

	Mean Absolute Error DVH [Gy]			
	PTV	Rectum	Anal Sphincter	Body
	Mean ± std	Mean ± std	Mean ± std	Mean ± std
Matrad vs final model prediction	3.05 ± 1.17	1.99 ± 1.55	1.07 ± 1.02	0.39 ± 0.65
Pinnacle vs U-net prediction	0.80 ± 0.28	1.30 ± 0.88	0.87 ± 0.85	0.39 ± 0.39

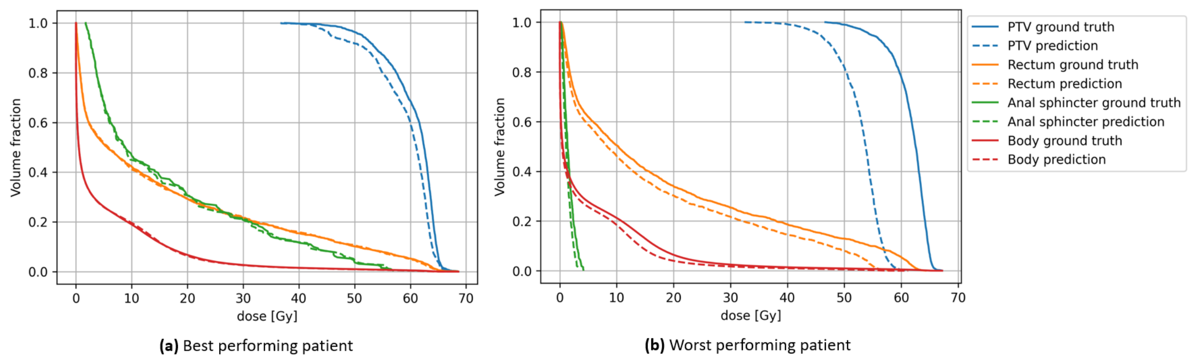


Figure 4.22: Dose Volume Histograms of the different structures for the ground truth and the Physics-guided prediction model.

4.5. Dose Mimicking Model Performance

From the previous section, it became clear that the proposed model did not have the expected outcome. Therefore it is interesting to see how well a dose mimicking algorithm can perform to obtain better results on both the segments and the dose output. To evaluate this, a single test patient is taken with a dose prediction from the Anatomy-based dose prediction model as described in Section 3.3. In Figure 4.23, an example slice of the ground truth, predicted and mimicked dose is shown.

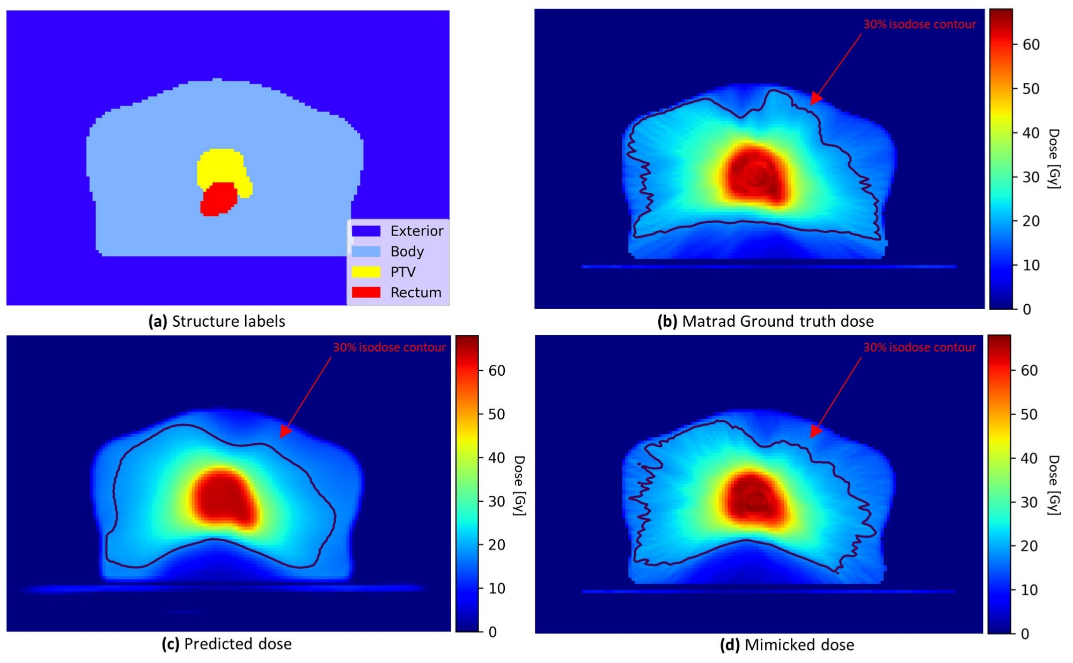


Figure 4.23: Example slices of the structure labels, Matrad ground truth, Anatomy-based dose prediction model prediction and mimicked dose for a test patient.

From these images can be seen that the prediction shows again the lack of ray effects, even though the overall shape of the dose distribution follows the ground truth nicely. In the mimicked dose, however, the ray effects are much more visible, causing the jagged shape of the 30% isodose contour. The main difference here with the previous model is that in the previous model, the overall dose distribution seemed to poorly follow the ground truth, which appears to be better in this model. This can be seen in, for example, the colour of the high dose region of the mimicked dose, which is quite similar to the ground truth. Still, it can be seen that not the exact same ray effects are visible, meaning that the dose is delivered differently.

This is also noticed when inspecting Figure 4.24, where the best and worst-performing bixel weight maps are shown for the ground truth and the prediction. Similar issues are seen here as were seen for the Segment prediction model in Figure 4.13, where some maps might be pretty accurate, while others are entirely wrong.

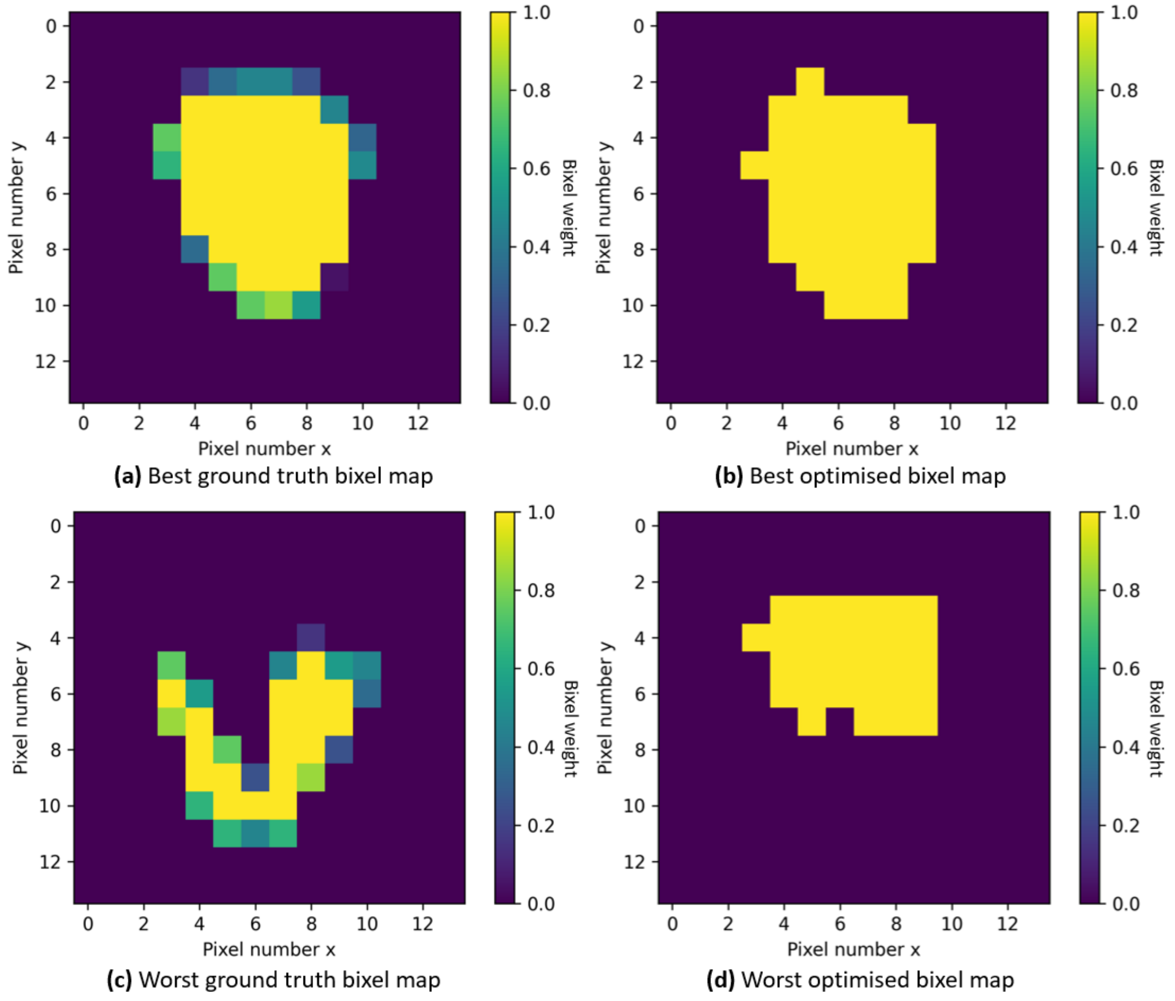


Figure 4.24: Comparison of the bixel weight maps of the ground truth treatment plan and optimised in the Dose mimicking model for the most and least accurate maps.

In Figure 4.25, the optimised beam intensities are presented and compared to the ground truth beam intensities. It is interesting to see that in this case, completely different intensity values are obtained, and there seems to be no resemblance at all between the ground truth and the optimised values. The combination of the optimised bixel weight maps and beam intensity values seem to be very different from the ground truth, even though the mimicked dose appears to be quite similar to the ground truth.

To see how accurate the mimicked dose is precisely, a DVH is plotted of the ground truth, predicted and mimicked dose in Figure 4.26. Two main points can be made from this plot. First of all, it seems that for the OAR structures, the mimicked dose is capable of following the prediction dose closely. This means that the quality of the mimicked dose is mainly dependent on the quality of the predicted dose. This fact is well portrayed in the rectum structure, where the prediction seems the be slightly different from the ground truth, but the mimicked dose stays very close to the prediction. Improvement on this DVH curve would therefore mainly be possible by improving the dose prediction and not the mimicking algorithm. On the other hand, for the performance on the PTV, it is the other way around. Here the prediction seems to be quite accurate, but the mimicked dose does not follow the correct curve at all. Still, the differences are not as big as the differences obtained with the Physics-guided prediction model.

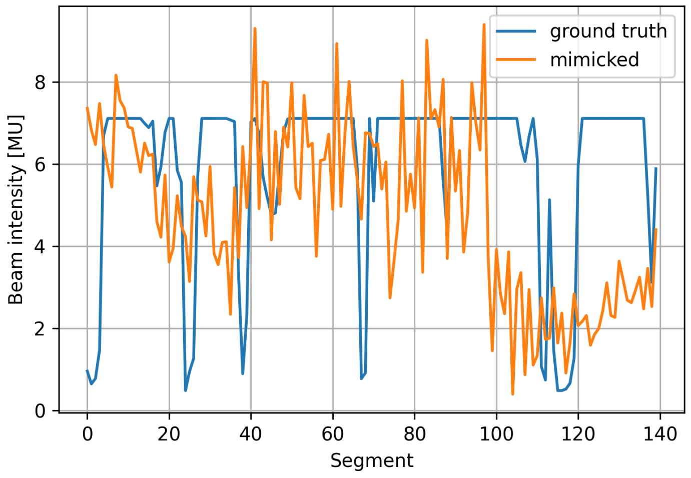


Figure 4.25: The beam intensity values of the ground truth and the Dose mimicking model.

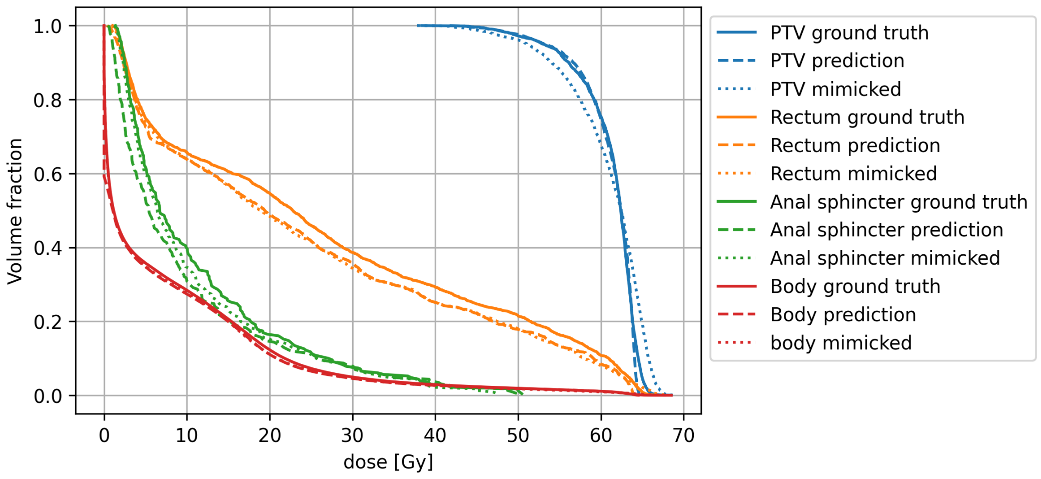


Figure 4.26: Dose Volume histogram of the ground truth, predicted and mimicked dose distribution for the different structures of the example test patient.

In Table 4.9 the mean absolute DVH differences between the ground truth, prediction and mimicked dose are shown for the chosen test patient. The findings mentioned above are confirmed with these numbers. In the PTV DVH, the prediction is quite accurate and has a minimal difference with the ground truth of about 0.24 Gy on average. The difference in the PTV DVH of the mimicked dose with the other doses is slightly higher and slightly above the 1 Gy difference. For the OARs, the difference between predicted and mimicked dose is generally much lower than the difference between prediction and ground truth, and for the anal sphincter and the body volume, the difference between ground truth and mimicked dose is even smaller than the difference between prediction and ground truth. This last point shows that by adding the physics to the predicted dose, a better dose distribution is obtained for these specific structures.

Table 4.9: Mean Absolute Error of the DVH curves for the Dose mimicking model. The values represent the mean and standard deviation and are given in Gy, representing the average horizontal difference between the predicted and ground truth DVH graph.

	Mean Absolute Error DVH [Gy]			
	PTV	Rectum	Anal Sphincter	Body
	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std
Ground truth vs prediction	0.24 ± 0.29	2.09 ± 1.44	1.57 ± 0.64	0.38 ± 0.36
Prediction vs mimicked dose	1.20 ± 0.81	0.39 ± 0.43	1.07 ± 0.43	0.26 ± 0.24
Ground truth vs mimicked dose	1.10 ± 0.61	2.16 ± 1.61	0.72 ± 0.74	0.30 ± 0.44

5 Discussion

In this chapter, a discussion is held on the results that have been presented in Chapter 4. The goal is to find meaning behind the results and place them in the context of other research done in this field of science. The structure of this chapter follows the order in which the different models and components have been presented.

5.1. Anatomy-based Dose Prediction Model

There were two main reasons for creating and training the Anatomy-based dose prediction model. First, the Physics-guided prediction model and the Dose mimicking model need a dose prediction neural network that works well. Second, one of the main goals of this thesis was to solve the problem of undeliverability of dose predictions made with Anatomy-based dose prediction models, which meant that the undeliverability of the dose predictions had to be confirmed in some way.

5.1.1. Quality of the Model

The results have shown that the U-net and HD U-net are very comparable in performance, with the HD U-net performing slightly better on some evaluation metrics, while the U-net performs better on others. The HD U-net has shown success both for head-and-neck and lung cancer patients [6], [41]. Especially in the work of Nguyen et al., it was shown how the HD U-net outperforms the U-net with dose predictions. It seems that in this thesis, the difference between the architectures is not as significant, which could be caused by several reasons.

First of all, from Figures 4.1 and 4.2 it is quite clear that the HD U-net is less capable of keeping the model more generalised. For the U-net, the training and validation loss seem to stay at the same value throughout the training, but for the HD U-net, the training loss quickly becomes considerably lower than the validation loss. Different growth rates have been used for the HD U-net to solve this issue, but no changes were observed in the obtained training curves. Moreover, using different augmentation schemes did not solve the issue. The lack of generalisation for the HD U-net could mean that not the most optimal training is achieved for this model, limiting its full potential. It seems that this combination of patient data en model architecture is more prone to overfitting and possible solutions could be the use of a larger dataset, higher dropout probabilities or smaller initial learning rate.

Another reason for the lack of differences could be that the architecture of the U-net is already good enough for a prostate cancer prediction. A prostate cancer dose distribution is generally much less complex than a head-and-neck cancer dose distribution and much less versatile compared to lung cancer patients. The difference in complexity between head-and-neck and prostate cancer patients is mostly established in the more intricate shapes of the isodose contours for head-and-neck cancer, which require more high-frequency details for an accurate prediction. From Figure 4.3 can be seen that the HD U-net example slices show slightly more high-frequency details compared to the U-net, where a very uniform dose distribution is visible. A clear example of this is how the HD U-net is capable of predicting a small detail like the dose in the VMAT table, which is ignored by the U-net. What this could mean is that the U-net is better in predicting more averaged and uniform dose distributions, while the HD U-net is better at predicting complex, highly detailed dose distributions. As both head-and-neck and lung cancer dose distributions are more complex than prostate cancer dose distributions, this could explain why the HD U-net performs better there. As for the prostate dose, small details are less important for the evaluation metrics used, and therefore less of a difference is visible between the architectures.

Apart from testing the two different architectures, the effect of adding CT data input was also investigated, which was done on dataset II. From the results was shown that no significant improvement was made on the dose prediction with this addition, even though adding this input means that the model has more information on the patient's anatomy. Adding CT as input essentially gives the model information on the radiological depth in the patient. As most of the patient's tissue will act very similar to water (except for the femurs) the added effect of this can be expected to be marginal. Furthermore, the difference between patients on this information will also be very small, meaning that there is not much to gain from adding it as input. This could explain the indifference of the model towards this input.

This conclusion also fits in other research where the difference in performance has been tested between a CT-only prediction and a CT + contours prediction for prostate cancer (Willems et al. [65]). The insight obtained by Willems et al., is that adding the contours to a CT-only prediction improves the prediction quality significantly. Combined with the findings in this part of the study, it can be argued that the structure contours are more important for the dose prediction model than the CT input.

Of course, another possibility is that by simply adding the CT in an extra channel, the model is not able to extract the information properly from the input data. An example of successful implementation of a model with CT is the dual pyramid adversarial network from *Momin et al* [66]. Here, separate dose prediction neural networks are used to generate a prediction based on CT-only and contour-only input. In a third fusion network, these outputs are then combined in a final dose distribution. The reason that adding the CT input is more successful is that an entire network is dedicated to extracting the information from CT, which is done separately from the contour information extraction.

Even though the HD U-net and the addition of CT to the input did not give any significant performance increase, the U-net with a contour-only input did already perform accurately with excellent performance metrics values. The model was shown to perform significantly better on both the DVH evaluation and the dose coverage statistics compared to the results from Meerbothe on dataset I [7]. This increase in performance can be assigned to the fact that bugs in the network design were removed, which caused single layers with the same parameters to be used multiple times in the network. Furthermore, additional measures to prevent overfitting were implemented, like early stopping and dropout regularisation. Also, when comparing the quantitative analysis of the U-net performance to other recent research in VMAT dose predictions for prostate cancer, the model proves to be of high quality. This can be seen in Table 5.1, where the mean absolute difference between ground truth and predicted dose coverage statistics are compared to other recent studies done on VMAT dose predictions for prostate cancer. From the table can be seen that the error in the dose coverage statistics for this model is very low when compared to these other studies.

Table 5.1: Comparison of the best results of the Anatomy-based dose prediction model with other recent studies on VMAT dose predictions for prostate cancer. The percentages denote the mean absolute difference between the predicted and ground truth dose statistic as a percentage of the prescribed dose.

Study	**Architecture**	**PTV D_{95}**	**PTV D_{98}**	**PTV D_{max}**	**PTV D_{mean}**
This study	3D U-net	0.54%	0.71%	0.63%	0.61%
Lempart et al. [67]	Dense 2.5D U-net	1.0%	1.9%	-	-
Wang et al. [68]	Progressive U-net	-	1.2%	3.2%	-
Kumanee et al. [69]	GAN	-	7.11%	1.38%	4.82%
Willems et al. [65]	3D U-net	-	1.0%	1.3%	-
Nguyen et al. [70]	2D U-net (IMRT)	-	-	1.80%	1.03%

The reason for this significant difference in performance with other research, can most likely be assigned to the homogeneity of the dataset. As all plans have been generated with an auto-planning system without making any manual alterations, the way in which the plans are generated is very consistent through the entire dataset. This is most likely not the case in other research, as treatment plans are often adjusted slightly to create the most optimal plan.

5.1.2. The Issue of Deliverability

Even though the U-net model seems to be performing very well on all dose evaluation metrics, the fact remains that the deliverability of the predicted dose seems questionable at the least. This is confirmed in Figures 4.3 and 4.7 and the difference between the ground truth and predicted 30% isodose contours. This shows indeed that there is a need for a model where deliverability is ensured. In most other studies on dose predictions, this issue seems to be ignored, even though the undeliverability is very apparent in the example slices shown. There is some research present on dealing with this issue, but this is always done using either an extra optimisation step or using a second deep learning model to, for example, predict fluence maps [67], [71], [72]. This confirms what was expected: to be able to generate deliverable dose predictions, an extra step has to be added.

5.1.3. Clinical Applicability

Even though deliverability of the dose prediction is not guaranteed, the quality of these predictions are very good and consistent, meaning that the Anatomy-based dose prediction model could still serve as a good QA method for an auto-planning system. However, it should be noted that due to the lack of deliverability it is uncertain if this model will be capable of making correct predictions for outlier patients, which is exactly where a quality assessment tool should function properly. Implementation of the model is simple once it has been trained, and making a prediction for an individual patient only takes a few seconds. The auto-planned dose distribution and the predicted dose distribution can then be compared by using assessment metrics such as the DVH or the dose coverage statistics.

5.2. Segment Prediction Model

The best results obtained with the Segment prediction model were slightly underwhelming, and the model has shown to be incapable of predicting the segments accurately enough.

5.2.1. Quality of the Model

From Figure 4.14 can be seen that there is a vast spread in the quality of the MLC maps, where most lie around a MALD value of 3 mm, with some outliers towards MALD values of 7 mm and even 11.6 mm in the worst case, but also positive outliers up until a MALD of 0.4 mm. Looking at the example segment predictions in Figure 4.13, it can be seen that the MLC is sometimes set to quite a large opening, where the PTV shape is mostly followed, while in other cases, the PTV contour is not at all followed and a tiny opening occurs. The high MALD outlier segments seem to happen at places where the prediction takes an opening that is following the PTV contour, while the ground truth takes a small opening or vice versa.

Furthermore, the error in the beam intensity prediction was also high, with an overall mean absolute error of 1.39 MU. When inspecting Figure 4.15, it is clear that the beam intensity predictions look very different from the ground truth, and the model is not capable of understanding what a beam intensity prediction should look like.

As there are not a lot of studies done on the prediction of VMAT segments, it is not easy to find research to compare the results to. Two papers were found on predicting treatment plans, from Hibbard [62] and Fan et al. [73], where especially the work of Hibbard can be related to this study. In his work, a similar type of prediction is done for VMAT prostate cancer patients using a cycleGAN network. In this case, the prediction had an overall average MALD of 1.62 mm, which is quite a bit better than the results obtained in this study.

An Explanation for the better performance here could be that a cycleGAN might perform better, which seems unlikely as a conditional GAN has also been tested during this study, which has shown results that are very similar to the single-encoder, dual-decoder Pix2pix-inspired U-net, presented in Subsection 3.4.3.

Another reason could be that in Hibbard's study, 2D convolutions are used instead of 3D convolutions. This means that predictions are created per segment, and the different segments are used in the channels. The reason for avoiding 2D convolutions in this study is that it would mean that all structures are put in a single channel, as the different channels must be used for the segments. This means that the structures would have to be labelled, and it will be challenging to make the model recognise overlapping structures.

A last reason for the difference could be the difference in plan and input data. In the study of Hibbard, 178 patients were used, with a treatment plan defined on 128 control points. The fact that, in this thesis, fewer patients are available while also needing to predict more segments could be a reason for the performance difference.

5.2.2. Clinical Applicability

Even though the Segment prediction model is not performing well enough in its current state, at some point, it could be performing better. In that it case it could potentially be used for clinical application as a much faster alternative to auto-planning systems. For the application, BEV images of the delineated structures will have to be made in real-time, which could take a few minutes in the current implementation. After that, prediction of the segments are very fast, and the per-segment MLC maps and MU values can be quickly translated to leaf positions and cumulative weights and arc intensities. It should be noted, however, that an implementation like this is still very far away and it will be difficult to guarantee quality in a deep learning treatment plan prediction.

A much more realistic application for the segment prediction model could be to decrease the computation time for the auto-planning process. The prediction could for example be used as a warm start for the system to start the optimisation from, which could potentially save on the number of iterations required before an optimal plan is obtained.

5.3. Dose Engine

From the comparison between the Matrad and Pinnacle dose engine was found that even though good deliverability was ensured and correct ray effects were visible, corresponding with a good 30% isodose contour overlap, there were significant differences in the DVH dose coverage metrics. These significant errors were especially apparent in the high dose areas, where the Matrad dose is much less uniform, which is also visible in the example slices of Figure 4.17. Obviously, some differences could have been expected as different algorithms are used in Pinnacle and Matrad. The reason for going for the different algorithm was mainly to be able to create a pencil beam database, allowing quick dose calculations necessary for the training of the final models. Furthermore, to allow for the backpropagation through the dose engine, every operation on the data had to be differentiable, which would have been very difficult to implement for a collapsed cone convolution algorithm.

Still, the different algorithms do not account for all differences, and when the DVHs are considered in Figure 4.18, it can be seen that the clinical accuracy of the dose engine is simply not sufficient in some cases. This error can mainly be assigned to the large bixel size of 10 mm. With this large bixel size, it is impossible to translate the intricate details of the MLC maps to the dose distribution, which are especially important for the details in the high dose area. In the early stage testing of the dose engine implementation, 5 mm bixels were tested, and the conformity was much better compared to Figure 4.17. The issue, however, is that storing the individual pencil beams is not feasible with 5 mm bixels, which is why the 10 mm bixel size was chosen in the end.

By using the Matrad engine to regenerate the ground truths, it is possible for the final models to obtain the same dose as the ground truth. The only downside is that in some cases, these ground truths are clinically not making too much sense. Apart from the clinical inaccuracies, however, the dose engine was implemented correctly with accurate ray effects and visually very similar dose distributions. Furthermore, the way that the dose engine was implemented allowed for rapid calculation times of about 2 minutes per patient (without backpropagation).

5.4. Physics-Guided Prediction Model

The Physics-guided prediction model was implemented successfully, and the model was capable of backpropagating losses through the entire network. Unfortunately, the model did not perform as well as was expected.

5.4.1. Quality of the Model

Training the model from scratch did not work very well due to the fact that it took quite some time for the Anatomy-based dose prediction model to reach a point where it outputs reasonable dose distributions. Before this point is reached, the Segment prediction model is fed nonsense data, meaning that this model does not perform correctly until this point is reached. An easy solution to this problem is to pre-train both the segment and dose prediction model and use these to kick-start the training of the Physics-guided prediction model. This idea worked well enough; nevertheless, no improvements seemed to be made during the final training. This is also apparent in the results shown in Section 4.4, which do not show any considerable improvement in the treatment plan prediction compared to the results from Section 4.2.

The most likely reason for the fact that this model performs poorly is the poor performance of the Segment prediction model. It is seen in Section 4.2 that training this model on its own does not give accurate predictions, which means that at the start of the training, the model is quite far off. The Segment prediction model seems to simply be incapable of predicting the segments accurately enough, and even after adding the physics-guided training step, the model does not improve itself. This inaccuracy is, in some cases, clearly large enough to cause a significant difference in the calculated dose obtained from the predicted treatment plan. This shows that the task given to the Segment prediction model is too complex for the proposed architecture.

An interesting finding for this model is that the reason for the difficulties in predicting the segments is caused by the fact that it is a highly degenerate problem. In the example slices of Figure 4.21 it can be seen that sometimes different gantry angles are favoured more in the prediction to deliver the dose compared to the gantry angles favoured in the ground truth. As the shape of the high dose area still looks pretty similar, this shows that there are multiple ways of delivering the required dose. This could explain why the Segment prediction model struggles to perform accurately as there are simply many good treatment plans that can be defined, and based on the input data given, there is no reason to favour one over the other. So while the ground truth might choose to deliver a lot of dose from a specific direction, the prediction might choose another direction and still obtain a good calculated dose. The main issue seems to be the absolute dose value that is achieved in the high dose region, which is caused by the poor prediction of the beam intensities.

This suggests that it might be fruitful to simply let go of the ground truth treatment plan and no longer force the segment predictor to adhere to a specific ground truth. Instead, allowing it to find its own best treatment plan, based only on the calculated dose, could be a better solution, as in the end, an accurate dose distribution is what is really important. By forcing it to adhere to the ground truth treatment plan, the model tries to stick somewhat to the ground truth, but because it is failing in that, the result is a not so good estimation of this, resulting in an inadequate dose distribution.

5.4.2. Clinical Applicability

If at some point the Segment prediction model could be improved, it can be interesting to see if the Physics-guided prediction model performance could perform on par or even surpass the quality of the Anatomy-based dose prediction model. If this is the case, there are definitely some advantages to be gained, not only because deliverability is guaranteed with this model, but also because the complete treatment plan is defined in the prediction. When comparing this plan to the auto-planned treatment plan for QA, a more in-depth comparison can be done. It should be noted, however, that making a prediction takes slightly longer (a few minutes) compared to the Anatomy-based dose prediction model, due to the real-time BEV processing step and the dose calculation required.

5.5. Dose Mimicking Model

For the specific example patient that has been testing for the Dose mimicking model, the mimicked dose was of better quality than the calculated dose in the Physics-guided prediction model. The main reason for the quality improvement is that the dose calculation no longer depended on the poor performing segment predictor but on the optimisation algorithm, which tried to get the dose as close as possible to the predicted dose. The added effect of this is that the model does not try to keep its treatment plan close to the ground truth treatment plan but only keeps in mind the final dose distribution. This was also apparent in the ray effects of the mimicked dose, which were clearly visible but different from the ground truth. Again, it became clear from these results that a completely different treatment plan does not mean a worse dose prediction. It confirms the high level of degeneracy present in the treatment planning and could be a potential cause why the segment predictor struggles to keep to the ground truth treatment plan. This is noticed here through the fact that the treatment plan and especially the beam intensities are less comparable to the ground truth than the predicted treatment plan.

6 Conclusion

In conclusion, different deep learning models have been (further) developed and evaluated to obtain accurate, realistic and deliverable dose distributions. The goal of this study was to add physics information to the deep learning model and force dose predictions to be realistic and deliverable, and the proposed models were successfully implemented.

The Anatomy-based dose prediction model performed very well and was able to outperform both the previous work from Meerbothe [7], as well as other recent work done on prostate cancer [65], [67], [68], [69], [70], mainly due to the high level of homogeneity in patient anatomy and in how the treatment plans were generated. Furthermore, it was verified that the Anatomy-based dose prediction model does not ensure deliverability due to the missing ray effects in the predicted dose. This was best quantified in the 30% isodose SDI dip present in the prediction to ground truth comparison.

Even though a correct implementation of the Physics-guided prediction model was achieved and deliverability with this model was confirmed with the presence of the ray effects in the predicted dose, the performance of this model was underwhelming, with a considerably worse dose prediction quality in comparison to the Anatomy-based dose prediction model. The component responsible for the lack of performance is the Segment prediction model, for which the specific data input, network architecture, and implementation used in this thesis have proven to be incapable of executing the complex task of predicting correct segments. The main issue for this component seems to be the high degeneracy of treatment planning, as there are many possibilities of obtaining a certain dose. This fact is also confirmed by the Dose mimicking model, where a similar dose distribution could be mimicked through a completely different treatment plan.

In the end, the Anatomy-based dose prediction and Physics-guided prediction models perform in an opposite manner, where the Anatomy-based dose prediction model makes high-quality predictions but has no guarantee of deliverability at all. The Physics-guided prediction model completely guarantees that the predicted dose is deliverable but makes poor-quality predictions. For a consistently accurate and reliable QA method, some middle ground between the two will be required.

7 Recommendations

For further research, there are two main areas where development could prove fruitful. First of all, the development of an alternative physics-guided model, which has a better balance between deliverability and dose quality, could be investigated. This would mean that no longer the treatment plan parameters of the predicted dose will be available. If the only goal of the model is QA, this is not a problem, and as long as deliverability is ensured to some extent, it could be a more consistent QA tool than a conventional dose prediction model.

Second, the Physics-guided prediction model still seems like a potentially interesting model, where both treatment plan and dose distribution can be predicted. For it to function better, at least a better Segment prediction model will need to be developed. Improvements to the model could be achieved by considering different model architectures, input data or implementation. Once a good-performing Segment prediction model is obtained, the Physics-guided prediction model could be reinvestigated with potentially better results. The model could then act as a consistent and reliable QA method, and this could even mean that the entire TPS can be replaced by such a model in the far future.

When reinvestigating the Physics-guided model, it might be better not to involve the loss of the predicted treatment plan in the training. The ultimate goal is obtaining an accurate and deliverable dose distribution, and which of the many possible treatment plans is chosen for this is not of importance. As predicting a treatment plan prediction is a highly degenerate problem, removing this loss in the training could improve the model further. Instead, adding penalties to the loss function based on how realistic the used MLC shapes and beam intensities are could be more helpful.

References

[1] WHO International Agency for Research on Cancer. *Cancer Today*. URL: https://gco.iarc.fr/today/home (visited on 03/23/2022).

[2] Hyuna Sung et al. “Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries”. In: *CA: A Cancer Journal for Clinicians* 71.3 (2021), pp. 209–249. DOI: https://doi.org/10.3322/caac.21660. eprint: https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.3322/caac.21660. URL: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21660.

[3] Thom G. A. Reuvers, Roland Kanaar, and Julie Nonnekens. “DNA Damage-Inducing Anticancer Therapies: From Global to Precision Damage”. In: *Cancers* 12.8 (2020). ISSN: 2072-6694. DOI: 10.3390/cancers12082098. URL: https://www.mdpi.com/2072-6694/12/8/2098.

[4] Jim Tol et al. “Can knowledge-based DVH predictions be used for automated, individualized quality assurance of radiotherapy treatment plans?” In: *Radiation Oncology* 10 (Nov. 2015). DOI: 10.1186/s13014-015-0542-1.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: (2015). Ed. by Nassir Navab et al., pp. 234–241.

[6] Dan Nguyen et al. “3D radiotherapy dose prediction on head and neck cancer patients with a hierarchically densely connected U-net deep learning architecture”. In: *Physics in Medicine & Biology* 64.6 (Mar. 2019), p. 065020. DOI: 10.1088/1361-6560/ab039b. URL: https://doi.org/10.1088/1361-6560/ab039b.

[7] Thierry Meerbothe. *A physics guided neural network approach for dose prediction in automated radiation therapy treatment planning*. Master thesis, Delft University of Technology, 2021.

[8] Geoff Delaney et al. “The role of radiotherapy in cancer treatment”. In: *Cancer* 104.6 (2005), pp. 1129–1137. DOI: https://doi.org/10.1002/cncr.21324. eprint: https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.1002/cncr.21324. URL: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.1002/cncr.21324.

[9] Kuo Ann Lee, Richard Yeo, and Kheng-Wei Yeoh. “Cancer and Radiation Therapy: Current Advances and Future Directions”. In: *International journal of medical sciences* 9 (Feb. 2012), pp. 193–9. DOI: 10.7150/ijms.3635.

[10] Saad Saeed. “Dynamic Log Files Analysis For Different Dose Rate IMRT Using DVH and Gamma Index”. PhD thesis. July 2015. DOI: 10.13140/RG.2.1.2254.5045.

[11] Alex Oliveira, J.W. Vieira, and Fernando Lima. “Monte Carlo Modeling of Multileaf Collimators Using the Code Geant4”. In: *Brazilian Journal of Radiation Sciences* 3 (May 2015), pp. 01–12. DOI: 10.15392/bjrs.v3i1A.144.

[12] Eric Klein et al. “Validation of calculations for electrons modulated with conventional photon multileaf collimators”. In: *Physics in medicine and biology* 53 (Apr. 2008), pp. 1183–208. DOI: 10.1088/0031-9155/53/5/003.

[13] Y. Nishimura and R. Komaki. *Intensity-Modulated Radiation Therapy*. 1st ed. Tokyo, Japan: Springer, 2015.

[14] S Webb. “The physical basis of IMRT and inverse planning”. In: *The British Journal of Radiology* 76.910 (2003). PMID: 14512327, pp. 678–689. DOI: 10.1259/bjr/65676879. eprint: https://doi.org/10.1259/bjr/65676879. URL: https://doi.org/10.1259/bjr/65676879.

[15] Keli Otto. “Volumetric modulated arc therapy: IMRT in a single gantry arc”. In: *Medical physics* 35 (Feb. 2008), pp. 310–7. DOI: 10.1118/1.2818738.

[16] Flemming Kjær-Kristoffersen et al. “RapidArc volumetric modulated therapy planning for prostate cancer patients”. In: *Acta oncologica (Stockholm, Sweden)* 48 (Nov. 2008), pp. 227–32. DOI: 10.1080/02841860802266748.

[17] Ben Vanneste et al. “Prostate Cancer Radiation Therapy: What Do Clinicians Have to Know?” In: *BioMed Research International* 2016 (Jan. 2016), pp. 1–14. DOI: 10.1155/2016/6829875.

[18] *Diagnostic Radiology Physics*. Non-serial Publications. Vienna: INTERNATIONAL ATOMIC ENERGY AGENCY, 2014. ISBN: 978-92-0-131010-1. URL: https://www.iaea.org/publications/8841/diagnostic-radiology-physics.

[19] Atchar Sudhyadhom. “On the molecular relationship between Hounsfield Unit (HU), mass density, and electron density in computed tomography (CT)”. In: *PLOS ONE* 15.12 (Dec. 2021), pp. 1–15. DOI: 10.1371/journal.pone.0244861. URL: https://doi.org/10.1371/journal.pone.0244861.

[20] Philippe Giraud et al. “Evaluation of microscopic tumor extension in non–small-cell lung cancer for three-dimensional conformal radiotherapy planning”. In: *International Journal of Radiation Oncology*Biology*Physics* 48.4 (2000), pp. 1015–1024. ISSN: 0360-3016. DOI: https://doi.org/10.1016/S0360-3016(00)00750-1. URL: https://www.sciencedirect.com/science/article/pii/S0360301600007501.

[21] Ramona Charaghvandi et al. “Redefining radiotherapy for early-stage breast cancer with single dose ablative treatment: A study protocol”. In: *BMC Cancer* 17 (Mar. 2017), p. 181. DOI: 10.1186/s12885-017-3144-5.

[22] International Centre for Theoretical Physics. *VMAT Dosimetric characteristics and delivery*. 2022. URL: http://indico.ictp.it/event/a14234/session/7/contribution/42/material/slides/.

[23] Werner Bär et al. “A comparison of forward and inverse treatment planning for intensity-modulated radiotherapy of head and neck cancer”. In: *Radiotherapy and Oncology* 69.3 (2003), pp. 251–258. ISSN: 0167-8140. DOI: https://doi.org/10.1016/j.radonc.2003.08.002. URL: https://www.sciencedirect.com/science/article/pii/S0167814003003062.

[24] Sebastiaan Breedveld et al. “Multi-criteria optimization and decision-making in radiotherapy”. In: *European Journal of Operational Research* 277.1 (2019), pp. 1–19. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2018.08.019. URL: https://www.sciencedirect.com/science/article/pii/S0377221718307148.

[25] Uwe Oelkfe and Christian Scholz. “Dose Calculation Algorithms”. In: *New Technologies in Radiation Oncology*. Ed. by Wolfgang Schlegel, Thomas Bortfeld, and Anca-Ligia Grosu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 187–196. ISBN: 978-3-540-29999-8. DOI: 10.1007/3-540-29999-8_15. URL: https://doi.org/10.1007/3-540-29999-8_15.

[26] Anders Ahnesjö and Maria Mania Aspradakis. “Dose calculations for external photon beams in radiotherapy”. In: *Physics in Medicine and Biology* 44.11 (Oct. 1999), R99–R155. DOI: 10.1088/0031-9155/44/11/201. URL: https://doi.org/10.1088/0031-9155/44/11/201.

[27] Woong Cho et al. “Practical implementation of a collapsed cone convolution algorithm for a radiation treatment planning system”. In: *Journal of the Korean Physical Society* 61 (Dec. 2012), pp. 2073–2083. DOI: 10.3938/jkps.61.2073. URL: https://doi.org/10.3938/jkps.61.2073.

[28] Yelda Elcim, Bahar Dirican, and Omer Yavas. “Dosimetric comparison of pencil beam and Monte Carlo algorithms in conformal lung radiotherapy”. In: *Journal of Applied Clinical Medical Physics* 19.5 (2018), pp. 616–624. DOI: https://doi.org/10.1002/acm2.12426. eprint: https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1002/acm2.12426. URL: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/acm2.12426.

[29] Thomas Bortfeld, Wolfgang Schlegel, and Bernhard Rhein. “Decomposition of pencil beam kernels for fast dose calculations in three-dimensional treatment planning.” In: *Medical physics* 20 2 Pt 1 (1993), pp. 311–8.

[30] Tomas M. Janssen et al. “Independent knowledge-based treatment planning QA to audit Pinnacle autoplanning”. In: *Radiotherapy and Oncology* 133 (2019), pp. 198–204. ISSN: 0167-8140. DOI: https://doi.org/10.1016/j.radonc.2018.10.035. URL: https://www.sciencedirect.com/science/article/pii/S016781401833576X.

[31] Barbara Vanderstraeten et al. “Automated Instead of Manual Treatment Planning? A Plan Comparison Based on Dose-Volume Statistics and Clinical Preference”. In: *International Journal of Radiation Oncology*Biology*Physics* 102.2 (2018), pp. 443–450. ISSN: 0360-3016. DOI: `https://doi.org/10.1016/j.ijrobp.2018.05.063`. URL: `https://www.sciencedirect.com/science/article/pii/S0360301618309155`.

[32] Kanabu Nawa et al. “Evaluation of a commercial automatic treatment planning system for prostate cancers”. In: *Medical Dosimetry* 42.3 (2017), pp. 203–209. ISSN: 0958-3947. DOI: `https://doi.org/10.1016/j.meddos.2017.03.004`. URL: `https://www.sciencedirect.com/science/article/pii/S0958394717300365`.

[33] Alexander L Fradkov. “Early history of machine learning”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1385–1390.

[34] Pariwat Ongsulee. “Artificial intelligence, machine learning and deep learning”. In: *2017 15th International Conference on ICT and Knowledge Engineering (ICT KE)*. 2017, pp. 1–6. DOI: `10.1109/ICTKE.2017.8259629`.

[35] Sinan Kaplan. “DEEP GENERATIVE MODELS FOR SYNTHETIC RETINAL IMAGE GENERATION”. PhD thesis. July 2017.

[36] Daniele Grattarola. “Deep Feature Extraction for Sample-Efficient Reinforcement Learning”. PhD thesis. Oct. 2017. DOI: `10.13140/RG.2.2.30267.31527`.

[37] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. 1st ed. Cambridge, Massachusetts: The MIT press, 2017.

[38] Yuxin Wu and Kaiming He. “Group Normalization”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[39] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[40] Gao Huang et al. “Densely Connected Convolutional Networks”. In: (2017), pp. 2261–2269. DOI: `10.1109/CVPR.2017.243`.

[41] Ana Barragán Montero et al. “Three□Dimensional Dose Prediction for Lung IMRT Patients with Deep Neural Networks: Robust Learning from Heterogeneous Beam Configurations”. In: *Medical Physics* 46 (May 2019). DOI: `10.1002/mp.13597`.

[42] Anuj Karpatne et al. “Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling”. In: *CoRR* abs/1710.11431 (2017). arXiv: `1710.11431`. URL: `http://arxiv.org/abs/1710.11431`.

[43] Ruiyang Zhang, Yang Liu, and Hao Sun. “Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling”. In: *Engineering Structures* 215 (2020), p. 110704. ISSN: 0141-0296. DOI: `https://doi.org/10.1016/j.engstruct.2020.110704`. URL: `https://www.sciencedirect.com/science/article/pii/S0141029619345080`.

[44] Nikhil Muralidhar et al. “PhyNet: Physics Guided Neural Networks for Particle Drag Force Prediction in Assembly”. In: Jan. 2020, pp. 559–567. ISBN: 978-1-61197-623-6. DOI: `10.1137/1.9781611976236.63`.

[45] Jinjiang Wang et al. “Physics guided neural network for machining tool wear prediction”. In: *Journal of Manufacturing Systems* 57 (2020), pp. 298–310. ISSN: 0278-6125. DOI: `https://doi.org/10.1016/j.jmsy.2020.09.005`. URL: `https://www.sciencedirect.com/science/article/pii/S0278612520301655`.

[46] National Electrical Manufacturers Association (NEMA). *About DICOM: Overview*. 2022. URL: `https://www.dicomstandard.org/about-home` (visited on 01/04/2022).

[47] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`. URL: `https://doi.org/10.1038/s41586-020-2649-2`.

[48] Darcy Mason et al. “pydicom/pydicom: pydicom 2.2.2”. In: (Oct. 2021). DOI: `10.5281/zenodo.5543955`.

[49] Stefan Van der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (2014), e453.

[50] National Electrical Manufactuerers Association (NEMA). *The DICOM Standard*. 2022. URL: https://www.dicomstandard.org/current (visited on 01/06/2022).

[51] Jiawei Fan et al. “Automatic treatment planning based on three-dimensional dose distribution predicted from deep learning technique”. In: *Medical Physics* 46.1 (2019), pp. 370–381. DOI: https://doi.org/10.1002/mp.13271. eprint: https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1002/mp.13271. URL: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.13271.

[52] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[53] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[54] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[55] PyTorch.org. *Build the neural network*. 2021. URL: https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html (visited on 01/21/2022).

[56] PyTorch.org. *Overview of PyTorch Autograd Engine*. 2021. URL: https://pytorch.org/blog/overview-of-pytorch-autograd-engine/ (visited on 01/26/2022).

[57] Vasant Kearney et al. “DoseNet: a volumetric dose prediction algorithm using 3D fully-convolutional neural networks”. In: *Physics in Medicine & Biology* 63.23 (Dec. 2018), p. 235022. DOI: 10.1088/1361-6560/aaef74. URL: https://doi.org/10.1088/1361-6560/aaef74.

[58] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[59] Aaron Carass et al. “Evaluating White Matter Lesion Segmentations with Refined Sørensen-Dice Analysis”. In: *Scientific Reports* 10 (May 2020), p. 8242. DOI: 10.1038/s41598-020-64803-w.

[60] Timothy Holmes, Charlie Ma, and Lu Wang. “Monitor Unit”. In: *Encyclopedia of Radiation Oncology*. Ed. by Luther W. Brady and Theodore E. Yaeger. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 515–515. ISBN: 978-3-540-85516-3. DOI: 10.1007/978-3-540-85516-3_351. URL: https://doi.org/10.1007/978-3-540-85516-3_351.

[61] Phillip Isola et al. “Image-To-Image Translation With Conditional Adversarial Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[62] Lyndon Hibbard. “Adversarial Prediction of Radiotherapy Treatment Machine Parameters”. In: *Multimodal Learning for Clinical Decision Support and Clinical Image-Based Procedures*. Springer, 2020, pp. 85–94.

[63] Hans-Peter Wieser et al. “Development of the open-source dose calculation and optimization toolkit matRad”. In: *Medical Physics* 44.6 (2017), pp. 2556–2568. DOI: https://doi.org/10.1002/mp.12251. eprint: https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1002/mp.12251. URL: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.12251.

[64] T. Janssen. “Personal communication”. In: (2021).

[65] Siri Willems et al. “Feasibility of CT-Only 3D Dose Prediction for VMAT Prostate Plans Using Deep Learning”. In: *Artificial Intelligence in Radiation Therapy*. Ed. by Dan Nguyen, Lei Xing, and Steve Jiang. Cham: Springer International Publishing, 2019, pp. 10–17. ISBN: 978-3-030-32486-5.

[66] Shadab Momin et al. “Learning-based dose prediction for pancreatic stereotactic body radiation therapy using dual pyramid adversarial network”. In: *Physics in Medicine & Biology* 66.12 (June 2021), p. 125019. DOI: 10.1088/1361-6560/ac0856. URL: https://doi.org/10.1088/1361-6560/ac0856.

[67] Michael Lempart et al. “Volumetric modulated arc therapy dose prediction and deliverable treatment plan generation for prostate cancer patients using a densely connected deep learning model”. In: *Physics and Imaging in Radiation Oncology* 19 (2021), pp. 112–119. ISSN: 2405-6316. DOI: https://doi.org/10.1016/j.phro.2021.07.008. URL: https://www.sciencedirect.com/science/article/pii/S2405631621000439.

[68] Jianyong Wang et al. “VMAT dose prediction in radiotherapy by using progressive refinement UNet”. In: *Neurocomputing* (2021). ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.11.061. URL: https://www.sciencedirect.com/science/article/pii/S0925231221017380.

[69] Patiparn Kummanee et al. “Predicting Three-Dimensional Dose Distribution of Prostate Volumetric Modulated Arc Therapy Using Deep Learning”. In: *Life* 11.12 (2021). ISSN: 2075-1729. DOI: 10.3390/life11121305. URL: https://www.mdpi.com/2075-1729/11/12/1305.

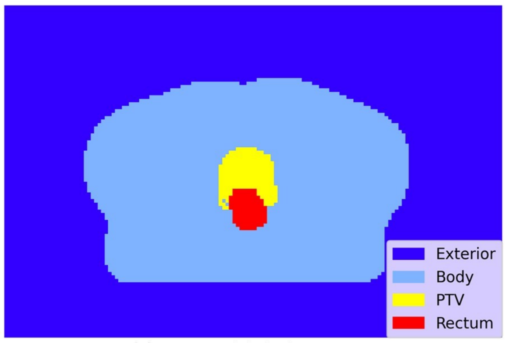
[70] Dan Nguyen et al. “A feasibility study for predicting optimal radiation therapy dose distributions of prostate cancer patients from patient anatomy using deep learning”. In: *Scientific reports* 9.1 (2019), pp. 1–10.

[71] Hoyeon Lee et al. “Fluence-map generation for prostate intensity-modulated radiotherapy planning using a deep-neural-network”. In: *Scientific Reports* 9 (Oct. 2019). DOI: 10.1038/s41598-019-52262-x.

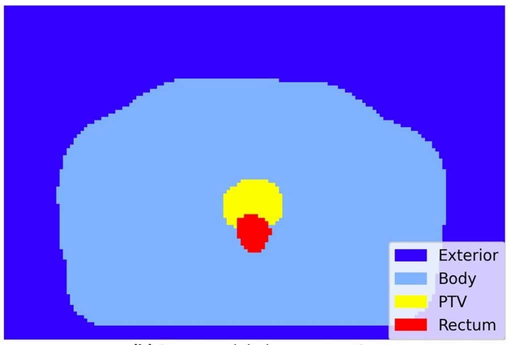
[72] Wentao Wang et al. “Fluence Map Prediction Using Deep Learning Models – Direct Plan Generation for Pancreas Stereotactic Body Radiation Therapy”. In: *Frontiers in Artificial Intelligence* 3 (2020). ISSN: 2624-8212. DOI: 10.3389/frai.2020.00068. URL: https://www.frontiersin.org/article/10.3389/frai.2020.00068.

[73] Jiawei Fan et al. “Verification of the machine delivery parameters of a treatment plan via deep learning”. In: *Physics in Medicine & Biology* 65.19 (Sept. 2020), p. 195007. DOI: 10.1088/1361-6560/aba165. URL: https://doi.org/10.1088/1361-6560/aba165.

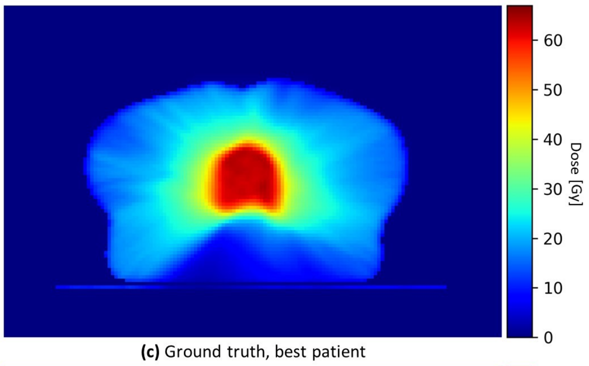
A
Dose Prediction with and without CT



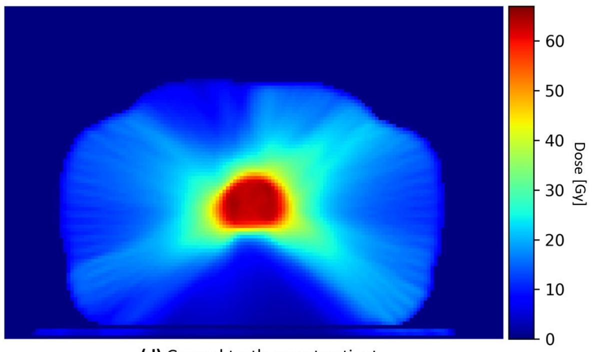
(a) Structure labels, best patient



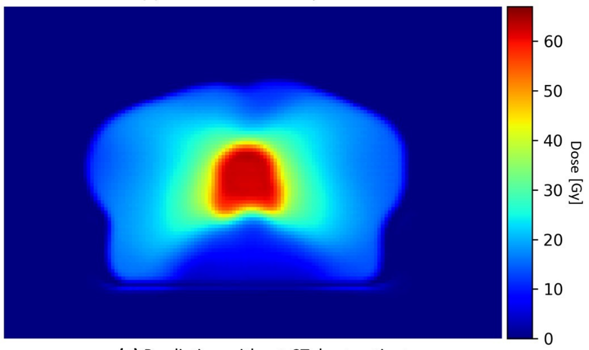
(b) Structure labels, worst patient



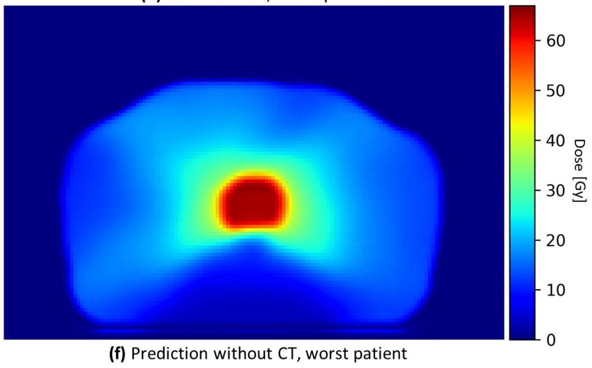
(c) Ground truth, best patient



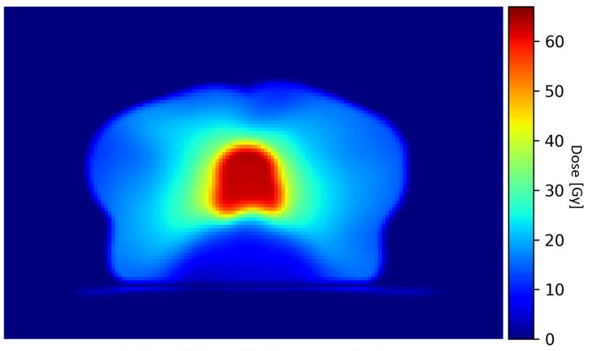
(d) Ground truth, worst patient



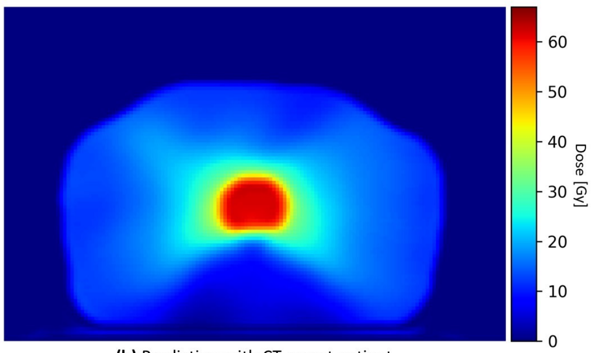
(e) Prediction without CT, best patient



(f) Prediction without CT, worst patient



(g) Prediction with CT, best patient



(h) Prediction with CT, worst patient

Figure A.1: Example slices for the best and worst performing test patient of dataset II of the structure labels, ground truth dose and anatomy-based dose prediction model predictions with and without CT input.

B
Dose Coverage Statistics for the Physics-Guided Prediction Model

In this appendix, the difference between ground truth and prediction with the physics-guided prediction model of the dose coverage statistics is given in percentages and compared to the performance of the anatomy-based dose prediction model. The results are given in tables B.1 and B.2. From these results can be seen that in its current state, the physics-guided prediction model is performing significantly worse than the anatomy-based prediction model.

Table B.1: Absolute dose difference as percentage of the prescribed dose between the ground truth and the prediction for four different coverage statistics on the PTV. The numbers represent averages and standard deviations over the 16 test patients. The physics-guided prediction model performance is compared to the anatomy-based prediction model performance.

	Absolute dose difference [%]: $100\% \cdot \frac{D_{\text{gt}} - D_{\text{pred}}}{60\ Gy}$			
	PTV coverage statistics			
Model	D_{95}	D_{98}	D_{max}	D_{mean}
	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std
Matrad ground truth vs Physics-guided prediction	6.68 ± 3.89	7.48 ± 3.99	5.35 ± 4.29	4.69 ± 3.72
Pinnacle ground truth vs U-net prediction	1.23 ± 1.06	1.41 ± 1.38	1.57 ± 0.92	1.30 ± 0.78

Table B.2: The relative absolute dose difference between the ground truth and the prediction for three different coverage statistics on the Rectum. The numbers represent averages and standard deviations over the 16 test patients. The physics-guided prediction model performance is compared to the anatomy-based dose prediction model performance.

	Absolute dose difference [%]: $100\% \cdot \frac{D_{\text{gt}} - D_{\text{pred}}}{D_{\text{gt}}}$		
	Rectum coverage statistics		
Model	D_{max}	D_{mean}	V_{45}
	Mean $\pm$ std	Mean $\pm$ std	Mean $\pm$ std
Matrad ground truth vs Physics-guided prediction	5.40 ± 4.17	6.53 ± 5.64	11.28 ± 8.45
Pinnacle ground truth vs U-net prediction	1.43 ± 1.06	5.30 ± 4.20	4.52 ± 3.86