Implementing beam shape variations in a deep learning based proton dose calculation algorithm

Nathan Geerts

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on the 23^d of August 2022

Student number: 5015413
Project duration: April 20, 2022 - August 1, 2022

Thesis committee:
Dr. Z. Perkó,
Mrs. Dr. ir. M.C. Goorden,
O. Pastor-Serrano

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Bachelor final project (BEP)
Bachelor Applied Physics
Delft University of Technology

Delft, Netherlands August 22, 2022

Supervisor: Z. Perkó
Supervisor: O. Pastor-Serrano

Abstract

In this report, a transformer based deep learning proton dose calculation algorithm called Dose Transformer Algorithm (DoTA) is described. This model learns to predict proton dose distributions by being trained with Monte Carlo generated data. Monte Carlo is the golden standard of proton dose calculation because it is very accurate, but it has relatively long computation times. In current proton therapy treatment programmes, Monte Carlo algorithms are the most commonly used models to perform dose calculation. The goal of the DoTA model is to predict proton dose distributions with Monte Carlo accuracy in the fraction of the computation time of Monte Carlo algorithms to speed up the dose calculation process in proton therapy treatment. The DoTA model can take patient geometry, random proton beam energy and random proton beam shape (2D Gaussian with different major and minor axes) as input. The addition of the random proton beam shape input is discussed in this report, together with a detailed explanation of the DoTA model. The DoTA model is trained with data from 9 lung cancer patients and 9 head & neck cancer patients. Being used on an Intel(R) Core(TM) i7-8565U CPU, the DoTA model managed to produce results with a gamma pass rate of 98.45 ± 2.60 % with an average computation time of 0.3 seconds. The gamma pass rate determines how similar the DoTA predicted dose is to the reference (Monte Carlo generated) dose. Compared to the average computation time of the Monte Carlo algorithm that was used to generate the training data, which is 20 seconds on the same CPU, we can conclude that the DoTA model has the potential to greatly improve dose calculation times in proton therapy treatment, especially when used on a system with greater processing power. Because the DoTA model is able to deliver accurate results in a small amount of time, it has the potential to be used for real-time dose calculation. Real-time dose calculation could account for small changes in patient geometry during treatment, which increases the accuracy of the treatment and minimizes side effects. The DoTA model can also be used for other radiotherapy types like phonon therapy and electron therapy (in that case it needs to be trained with different data).

Table of contents

1 Introduction

1.1. Radiotherapy

1.1.1. Radiotherapy in general

Ever since the discovery of X-rays [27] and Radium in the late 19th century [22], radiotherapy has been used as a treatment for cancer and up until present day has been one of the most common ways of treating cancer. Radiotherapy is a way of treating cancer that uses ionizing radiation to kill cancer cells and thus shrink a tumor in a patient. There are two main categories of radiotherapy treatments: internal radiotherapy and external radiotherapy [14][17]. For internal radiotherapy, the ionizing radiation is emitted from a liquid or an implant inside the body, while for external radiotherapy the ionizing radiation comes from a particle beam that is emitted by a machine outside of the patient. In this report the focus lies on a specific type of external radiotherapy: proton therapy. Other external radiotherapy types are electron therapy and, the most used type, photon therapy [19].

1.1.2. Benefits of proton therapy

Because electrons can't penetrate a patient much further than the skin, electron therapy can only be used for skin cancer and cancer cells close to the skin. Photon therapy is the most commonly used external radiotherapy but the fact that the photon beam passes all the way through the patient causes damage to healthy tissue and several different side effects. Proton therapy has an advantage on both of the methods previously mentioned because the proton beam stops somewhere around the tumor site and is very precise. Because of this, less healthy tissue damage occurs when proton therapy is used compared to when photon therapy is used [15][18][4].

In Figure 1.1 the difference between the propagation of an X-ray beam through tissue and the propagation of a proton beam through tissue is showcased. We can see that in the case of this figure, 12 proton beams are used to deliver radiation to the tumor. Each proton beam has a sharp energy delivery peak in a small range, this is what we call the Bragg Peak [16] and what makes proton beams so precise.

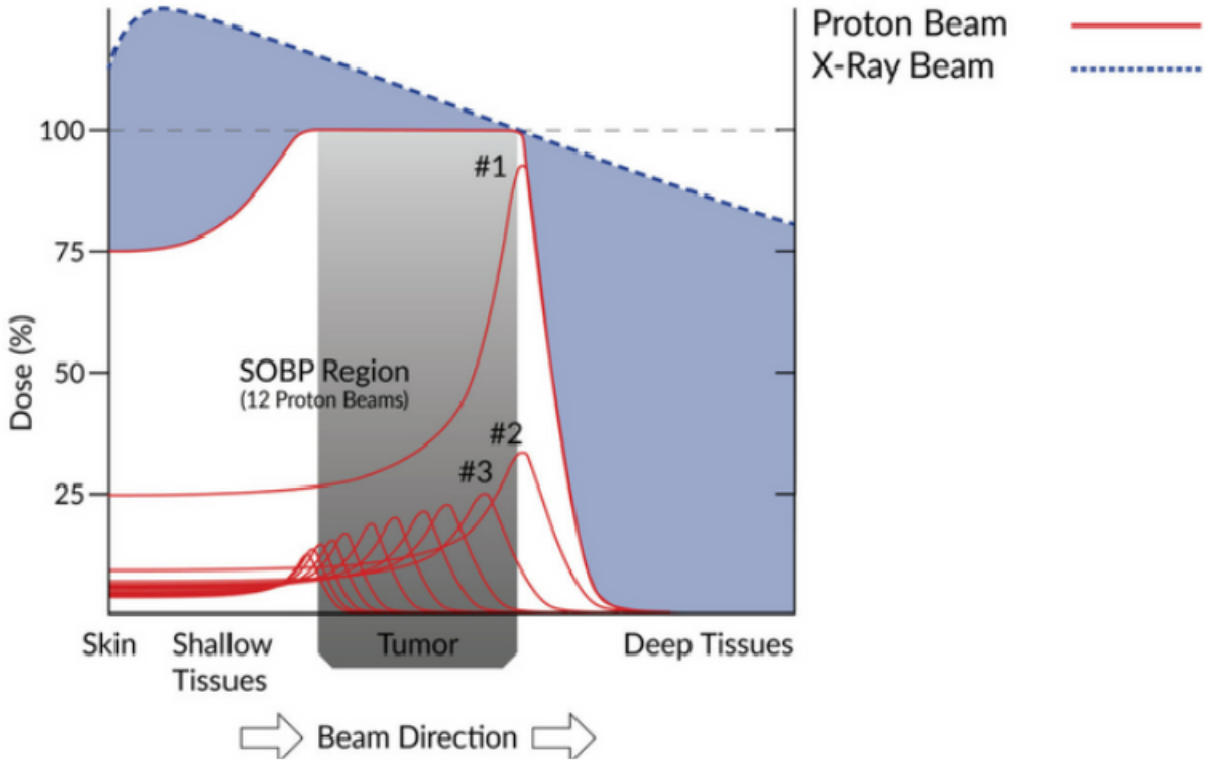


Figure 1.1: In this figure the difference between X-ray dose delivery and proton beam dose delivery is showcased by plotting the delivered dose in percentage relative to the necessary dose against the depth of the beam in a patient. The proton beam dose delivery is made up of 12 Bragg Peaks that make up the Spread-Out Bragg Peak (SOBP) region. It is visible that the proton beam delivers the necessary dosage in a more accurate manner than the X-ray beam. Image retrieved from [16].

1.2. Dose calculation

1.2.1. Importance of improving dose calculation speed

The process of using proton therapy on a patient consists out of several different steps: First consultation, simulation, treatment planning, treatment delivery and possible follow-up consultations [7]. The focus of this report is an important part of the treatment planning and treatment delivery: dose calculation. The dose that needs to be delivered to malignant tissue needs to be calculated very precisely so that surrounding healthy tissue isn't damaged. During the treatment planning, a dose calculation is performed based on CT scans of the patient geometry. After the dose calculation, the treatment delivery is provided in daily sessions of around 40 minutes, 5 days a week, for around 5 to 8 weeks on average [5] [28][9]. A new dose calculation can be performed during the treatment process if something drastically changes in the patient geometry. In modern day proton therapy delivery, there is no real time dose calculation. In most cases, dose calculation is performed at the start of the proton therapy treatment with a Monte Carlo algorithm (see section 1.2.2) which is not suitable for real time dose calculation due to its very long computation time. Real time dose calculation could change the delivered dose in real time to account for small changes in the patient geometry that are caused by coughing, breathing, intestinal movements, etc. and thus creates a more precise dose delivery. A more precise dose delivery means less irradiation of healthy tissue. This is why research is continuously done to find solutions for faster dose calculation.

1.2.2. Different dose calculation algorithms

The current most ideal calculation algorithms are Monte Carlo Algorithms (MCA) and Pencil Beam Algorithms (PBA). In the MCA, complex mathematical simulations allow the delivered particle dose to be determined with great accuracy. However, due to these complex simulations, the MCA has a very long computation time and therefore it is not usable for real time dose calculation. The PBA is an algorithm where the particle beam is simulated by combining a set of narrow pencil beams. The particle dose is then calculated in a slab geometry around every pencil beam. The main disadvantage of the PBA is that it has problems adjusting the dose calculation in the presence of heterogeneities lateral to the pencil beams [8][23][24][25]. The calculation per slab makes the PBA a very fast algorithm but it also makes the PBA lack in accuracy compared to the MCA [11].

The MCA is the golden standard due to its great accuracy, but the downfall of the MCA is the computation time. The PBA is way faster than the MCA but lacks accuracy. In this report, we will discuss a deep learning algorithm that produces MCA accuracy with PBA speeds. This algorithm is a Dose Transformer Algorithm (DoTA) that predicts the particle dose based on patient geometry input, beam energy input and beam shape input. Previously, this algorithm only had the geometry input and beam energy input, with the beam shape being a constant 2D Gaussian. The main task of this Bachelor Project was to implement beam shape as an input token for the deep learning algorithm so that the beam shape can be varied.

1.3. Structure of the report

In chapter 2, the theory behind the MCA generated data will be described and the theory needed to understand the DoTA model together with some general info about neural networks will be given.

In chapter 3, a description of how the MCA data generation will be performed and how the DoTA model will be utilized to predict proton dose distributions will be given. In this chapter there will also be discussed how we want to produce results.

In chapter 4, the results will be showcased with some additional discussion.

In chapter 5, a discussion about the DoTA model will be given. This includes: Model use case, limitations and future work/possible improvements.

Finally, in chapter 6, a conclusion about the project and the results will be drawn.

2

Theoretical Background

2.1. Neural networks and deep learning

Later in this chapter, the theory of the model that will be used in this project will be given. First a very basic explanation of a neural network will be given to create an understanding of deep learning for the reader. The neural network is an algorithm that is inspired by the human brain. It consists of several layers of so called neurons which mimic the way real biological neurons signal to one another. A neural network consists of an input layer, several hidden layers, and an output layer. Each layer consists of artificial neurons where every neuron of a certain layer is connected to all the neurons of the previous and next layer as can be seen in Figure 2.1 [10].

2.1.1. Forward propagation

Every neuron has its own assigned value. Each link between two neurons is called a connection and has a certain weight value attached to it that determines its "importance". For each neuron in the hidden layers, we need to determine if it will get activated or not (not activated = value 0) to know if this neuron will pass on its information or if it won't. To do this, call a random neuron from one of the hidden layers neuron A. For neuron A, we calculate the sum of the products of the values of the neurons in the previous layer with the weight that links these neurons to neuron A plus the bias that is attached to neuron A. We can formulate this explanation for a neuron in layer k into an equation called the feed-forward operation, as can be seen in Equation 2.1.

$$\sum_{z=1}^{N} w_z x_{z,k} + b_{k+1}, \tag{2.1}$$

with N being the amount of neurons in layer k, w_z being the weight of the connection between a neuron in layer k+1 and neuron number z in layer k, $x_{z,k}$ being the assigned value of neuron number z in layer k and b_{k+1} being the bias belonging to the neuron in layer k+1 [10].

Because of Equation 2.1 we can see that the size of the weights play a role in the determination of the activation of a certain neuron, hence why it was mentioned earlier that the weights determines a connections "importance". We can also conclude that the data is propagated through the neural network in the direction of input to output. This is what we call forward propagation.

2.1.2. Activation functions

To determine if a neuron gets activated or not, Equation 2.1 will be passed through what we call an activation function. There are many different activation functions [2], but the ones we will use in our DoTA model are

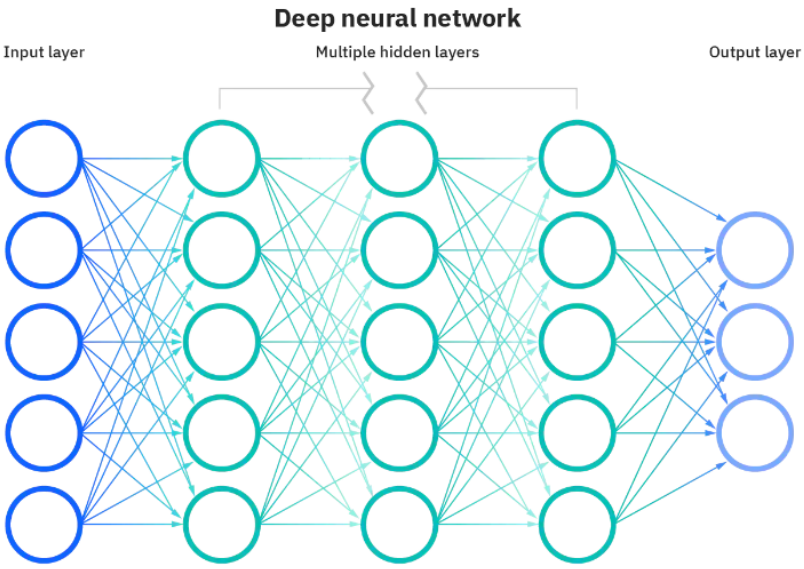


Figure 2.1: A simple representation of a neural network with an input layer, hidden layers, an output layer. The circles represent the neurons and the lines between the circles represent connections. Each neuron has its own value and each connection has its own weight. Image retrieved from [10].

the Rectified Linear Unit (ReLU) activation function (Equation 2.2) and the Gaussian Error Linear Unit (GELU) activation function (Equation 2.3):

$$ReLU(x) = max(0, x), \tag{2.2}$$

$$GELU(x) = x\Phi(x), \tag{2.3}$$

with $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.

2.1.3. Backpropagation

We do not know yet how accurate the output of the neural network is. To know this, a neural network needs to be trained with data that is already correct, we call this the target output. When we train the neural network, we want to minimize the error function. The error function is a function of the difference between the output of the neural network and the target output with respect to the weights of the neural network. By a method called gradient descent, the weights for which the error function is minimized will be determined. The information about the new weights is transferred backward through the neural network. We call this backpropagation [10].

In this project, the neural network we use will be trained by data that is generated with a MCA, which will be explained in the next subsection.

2.2. Monte Carlo Algorithms

The Monte Carlo algorithm is the golden standard for proton dose calculation. The algorithm is based on random sampling to obtain numerical results. In the case of proton path simulation, probability density functions are sampled. The probability density functions represent the probabilities of different particle interactions (absorption, annihilation, change of direction or change of energy). We call these probabilities cross sections. To accurately simulate the path a certain proton takes (track the proton), the sampling is done one small step at a time. These steps should be small enough so that the cross sections at the beginning and end of each step are similar. If the step size is too big, the MCA will give an inaccurate simulation of the real world physics but as the step size is decreased, the computation time will increase.

The use of so called multiple scattering theories make the simulation less complex. These multiple scattering theories provide probability density functions that contain the net result of multiple single scattering interactions. Sampling these new probability density functions will give you the scattering angle after the multiple single scattering events while only one sample was taken.

It is assumed that the energy of the tracked proton keeps decreasing continuously and thus is a continuous process. However, for protons there are also certain events that need to be analysed as a discrete process: nuclear interactions, secondary particle production and large angle Coulomb scattering.

In Monte Carlo algorithms, the proton transport simulation is stopped based on a particle energy threshold or particle range threshold that the user defines. These thresholds determine until what amount of energy and until what distance you want to simulate the proton respectively. The lower the energy threshold and the higher the particle range threshold, the more accurate the proton transport simulation will be, but the longer the simulation will take to compute. [20]

2.3. Theory for the Dose Transformer Algorithm (DoTA)

In this section, background theory about the following operations that will be used to build up our DoTA model will be given: convolution, pooling, normalization, self-attention and dropout.

2.3.1. Convolution

The convolution operation [13] is denoted with an * and is an operation that determines the overlap of 2 (discrete) functions. In our case we will use the convolution operation to reduce the dimensions of 2D input. This gives us the relation in Equation 2.4

$$(K * I)(i, j) = \sum_{m=1}^{M} \sum_{n=1}^{N} I(i-m, j-n)K(m, n), \tag{2.4}$$

with I being the 2 dimensional input, K being the convolutional filter (also called kernel) and M and N being the dimensions of the convolutional filter K. Figure 2.2 shows what convolution applied to a 2D data set looks like with M = 2 and N = 2.

2.3.2. Pooling

The pooling operation [13] replaces the output of a certain neuron with a summary of the output of the nearby neurons. In our DoTA model, we will specifically use max pooling. The max pooling operation replaces the output of a neuron with the maximum output in a rectangular neighbourhood of this neuron. Figure 2.3 shows the effect of the max pooling operation when the pooling region has a width of 3 neurons.

2.3.3. Normalization

Normalization [29] is an operation that normalizes a set of neighbouring features in the input image by their mean and variance to reduce model training time. Normalization methods perform Equation 2.5,

$$\hat{x}_i = \frac{1}{\sigma_i}(x_i - \mu_i), \tag{2.5}$$

where x is the feature computed by a layer, $\hat{x}$ is the normalized feature, σ is the standard deviation of a certain selected set of pixels, μ is the mean of this same set of pixels and i is an index. Because we deal with 2D images, i is a 4D vector ($i = (i_N, i_C, i_H, i_W)$ that indexes the features in the order (N,C,H,W) where N is the batch axis, C is the channel axis, H is the spatial height axis and W is the spatial width axis. The batch size indicates with how many input samples a neural network gets trained at once, the channel number indicates the input sample number in a batch and the height and width describe the 2D spacial dimensions of the input sample.

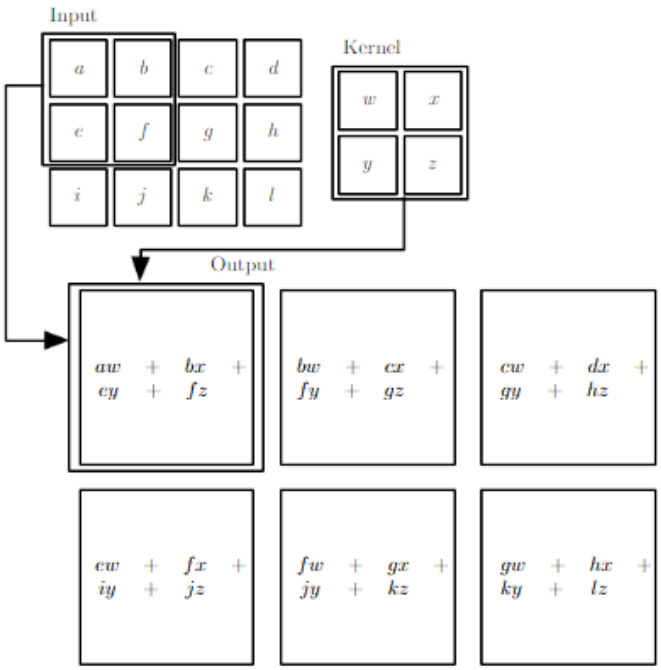


Figure 2.2: This figure shows how a 2D M = 2 by N = 2 convolutional kernel reduces the dimensions of a 2D discrete input by multiplying the values of the convolutional kernel with the values of the input it overlaps with and then summing all the multiplied values. This sum becomes the new value of one input cell. Image retrieved from [13].

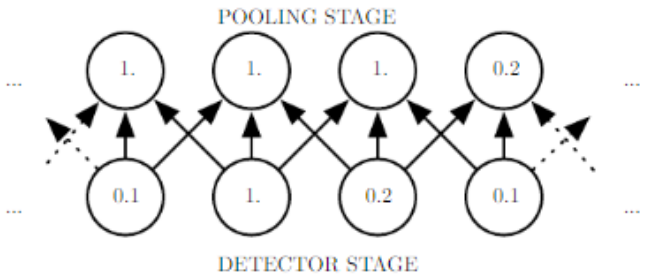


Figure 2.3: The above layer shows the values of the neurons after the pooling stage with a pooling region that has a width of 3 neurons. This means that for every value of a neuron in the bottom layer, the maximum value of this neuron, the neuron to the left and the neuron to the right is chosen as the output of the pooling layer, which is demonstrated in the figure. Image retrieved from [13].

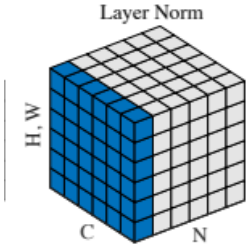


Figure 2.4: Figure that showcases that in layer normalization, μ and σ are computed along the (C,H,W) axes where C is the channel axis, H is the spatial height axis and W is the spatial width axis. Image retrieved from [29].

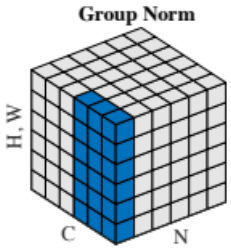


Figure 2.5: Figure that showcases that in group normalization, μ and σ are computed along the (H,W) axes and along $\frac{C}{G}$ channels per group where H is the spatial height axis and W is the spatial width axis. In this figure there are two groups (G = 2) which each contain 3 channels (C = 6). Image retrieved from [29].

In our DoTA model, we make use of Layer Normalization (LN) and Group Normalization (GN). LN computes μ and σ along the (C,H,W) axes for each feature. This is showcased in Figure 2.4. GN computes μ and σ along the (H,W) axes and along a group of $\frac{C}{G}$ channels where C is the number of channels and G is the number of groups. This is showcased in Figure 2.5.

2.3.4. Self-attention

If we have a sequence $\boldsymbol{z} \in \mathbb{R}^{L \times D}$ with L tokens, where tokens are one dimensional vectors with size D(= embedding dimension, see section 3.2.1), the self-attention (SA) operation transforms each i^{th} token $\boldsymbol{z}_i$ into a query vector $\boldsymbol{q}_i \in \mathbb{R}^{1 \times D_h}$, a key vector $\boldsymbol{k}_i \in \mathbb{R}^{1 \times D_h}$ and a value vector $\boldsymbol{v}_j \in \mathbb{R}^{1 \times D_h}$, where D_h is the dimensionality of these vectors. The query vector represents information that is to be gathered from other elements of $\boldsymbol{z}$, the key vector contains the type of information that is to be shared to other elements of $\boldsymbol{z}$ and the value vector is used to transform $\boldsymbol{z}_i$ into $\boldsymbol{z'}_i$ by applying Equation 2.6.

$$\boldsymbol{z'}_i = \sum_{j=1}^{L} w_j \boldsymbol{v}_j, \tag{2.6}$$

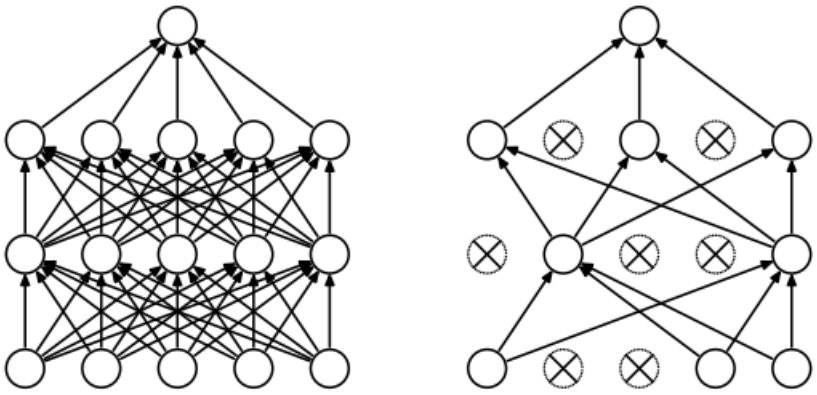
where the weight w_j denotes the similarity between the i^{th} query and the other keys in $\boldsymbol{z}$ and is defined as $w_j = \boldsymbol{q}_i^T \boldsymbol{k}_j$. The self-attention function is now defined by equation 2.7:

$$\boldsymbol{z'} = SA(\boldsymbol{z}), \tag{2.7}$$

where the $SA(\boldsymbol{z})$ function denotes what happens to the sequence $\boldsymbol{z}$ after self-attention.

In our DoTA model, we will also make use of a variant of the SA operation called multi-head self-attention (MSA). This variant runs N_h parallel SA operations which focus on different features and inter-dependencies of the data. If we set the size of $\boldsymbol{q}_i$, $\boldsymbol{k}_i$ and $\boldsymbol{v}_j$ (D_h) to D, the outputs of the parallel SA operations, called attention heads, are first concatenated and then linearly projected with weights $\boldsymbol{W}_h \in \mathbb{R}^{N_h D_h \times D}$ as

$$MSA(\boldsymbol{z}) = concat[SA_h(\boldsymbol{z})]\boldsymbol{W}_h \tag{2.8}$$



(a) Standard Neural Net (b) After applying dropout.

Figure 2.6: This figure showcases the effect of dropout on a neural network. When dropout is applied, random neurons in the neural network are deactivated. When a neuron is deactivated, it can't pass on its information to the following layer. Image retrieved from [26].

Because our DoTA model trains for proton trajectories, we use the causal variants of these SA and MSA operations. This means that the tokens can only attend to previous tokens and not to future tokens [21].

2.3.5. Dropout

With dropout [26], Equation 2.1 becomes:

$$\sum_{z=1}^{N} w_z \tilde{x}_{z,k} + b_{k+1}, \tag{2.9}$$

with

$$\tilde{x}_{z,k} = r_{z,k} * x_{z,k}, \tag{2.10}$$

where $r_{z,k}$ is a Bernoulli random variable (0 or 1). Dropout is used to prevent a neural network from overfitting. Overfitting happens when a neural network learns random fluctuations and noise in the training data as something that it should apply to input data, which makes the output inaccurate [3]. Figure 2.6 showcases what a network looks like before and after dropout.

2.4. Gamma analysis

Gamma analysis is used to compare calculated dose distributions to a ground truth dose distribution (in our case this is the Monte Carlo calculated dose distribution). With gamma analysis, the dose of voxels (volume elements) in the ground truth dose grid gets compared to the dose of neighbouring voxels in the predicted dose grid by using Equation 2.11:

$$\gamma(\mathbf{a}) = \min_{\mathbf{b}} \{\Gamma_{\mathbf{a},\mathbf{b}}(\delta, \Delta)\}, \tag{2.11}$$

with $\mathbf{a}$ being the coordinates of a voxel in the ground truth grid of which its dose will be compared to voxels in the predicted dose grid with coordinates $\mathbf{b}$, $\gamma(\mathbf{a})$ being the so called gamma value and $\Gamma_{\mathbf{a},\mathbf{b}}(\delta, \Delta)$ being given by Equation 2.12:

$$\Gamma_{\mathbf{a},\mathbf{b}}(\delta, \Delta) = \sqrt{\frac{|\mathbf{a}-\mathbf{b}|^2}{\delta^2} + \frac{|y_{\mathbf{a}} - y_{\mathbf{b}}|^2}{\Delta^2}}, \tag{2.12}$$

with $y_{\mathbf{a}}$ and $y_{\mathbf{b}}$ being the dose at coordinates $\mathbf{a}$ and $\mathbf{b}$ respectively, δ being the distance-to-agreement and Δ being the dose difference criterion. The distance-to-agreement is the radius of the sphere around a voxel in

which it is compared to neighbouring voxels and the dose difference criterion is the maximum error between a predicted dose value for a voxel and the reference value for the same voxel for these dose values to be seen as similar. The dose difference criterion is given asa percentage of the reference dose.

When a voxel has a gamma value $\gamma(\mathbf{a}) < 1$, it passes the gamma analysis. The gamma pass rate is the percentage of voxels that have passed the gamma analysis. A gamma pass rate is often followed by the dose difference criterion and the distance-to-agreement like this: gamma pass rate value in percentage (Δ, δ).

3

Experimental Method

3.1. Monte Carlo generated dataset

First the Monte Carlo data generation (Chapter 2.2) that is used to generate the training data for the DoTA model will be discussed. To generate the data an open-source MATLAB code called matRad [1] is used. MatRad is an open-source code that covers radiation planning for different types of energy beams (photon,proton,etc...). For the dose calculation of protons, matRad uses another open-source code called MCsquare [12].

With these codes, we calculate the dose distribution for 20 3D CT scans (10 CT scans of lung cancer patients and 10 CT scans of head neck (H&N) cancer patients). Because of the diameter of proton beams being about 25 mm and the travel distance of the proton beams in the patient geometry being not more than 300 mm, we reduce the CT scans to blocks of dimensions Y × X × Z = 150 × 24 × 24 with a 2 mm isotropic grid resolution. This means that the blocks cover a volume of 300 × 48 × 48 mm^3. The proton beam travels in the Y direction. We obtain multiple blocks per patient by rotating the CT sampling block in steps of 10° and shifting it laterally in steps of 10 mm. For every block, the dose distribution is calculated for a random beam energy in the range of 70 to 140 MeV and a random 2D Gaussian beam shape. The random 2D Gaussian beam shape was added to the data generation during this project. In the previous model, the beam shape was a 2D Gaussian with the same variance for both dimensions (circle). For the random 2D Gaussians, the variance of both dimensions are random and are not necessarily the same (ellipses). The random 2D Gaussian beam shape was added to the MC code by randomly changing the SpotSize1x and SpotSize1y beam parameters in the BDL file [6] of the MCsquare code and then calculating a 2D Gaussian beam shape for every block by using the square of SpotSize1x and SpotSize1y as the variances of the X and Z dimensions in the 2D Gaussian. The BDL file is a file that has the beam parameters for different beam energies between 70 and 225 MeV in steps of 5 MeV. For every block, SpotSize1x and SpotSize1y are extracted from the BDL file for a random beam energy between 70 and 140 MeV. We use the data of 1 patient of the lung cancer patient set and 1 patient of the H&N patient set as test data and the data of the other 18 patients as training data. This gives us 3276 test blocks and 36288 training blocks. We will use these training blocks to train our DoTA model.

3.2. Description of the DoTA model

We will first describe how the DoTA model calculates the dose distribution from geometry, energy and beam shape input. In Figure 3.1 the architecture of the DoTA model without the implementation of the beam shape variable is presented [21]. The DoTA model processes the input 3D $L \times H \times W = 150 \times 24 \times 24$ CT voxel grid (block) as L 2D slices with dimensions $H \times W$, with L being the depth (direction of proton beam propagation), H being the height and W being the width of the block. The user defined batch size determines how many input blocks the DoTA model processes at the same time.

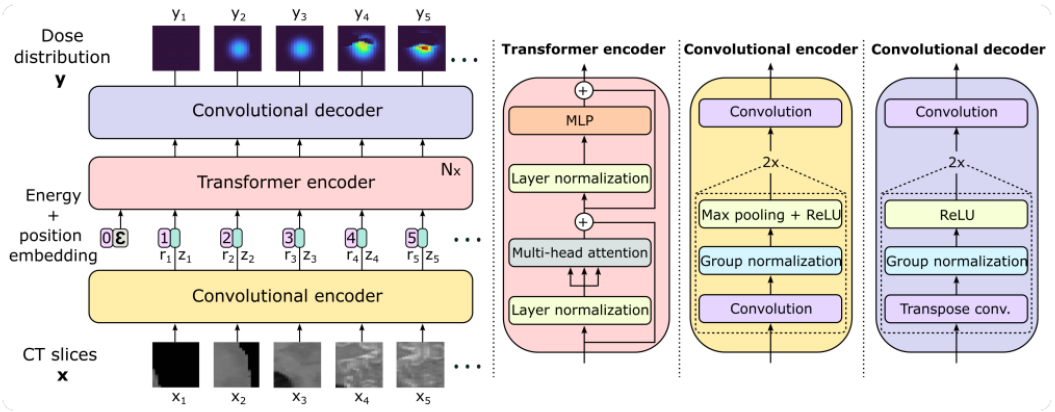


Figure 3.1: This figure showcases the model architecture of the DoTA model before the beam shape was implemented as a token. The 3D CT input is treated as a set of 2D slices which are passed through the convolutional encoder (together with a 2D beam shape slice in the new DoTA model). The convolutional encoder extracts important information from the 2D input such as tissue boundaries and material contrasts and saves this into a vector. The beam energy is added as a token at the beginning of the sequence resulting from the convolutional encoder. The transformer encoder with causal self-attention combines information from all the different elements in the sequence. The convolutional decoder transforms the output vectors of the transformer encoder into 2D dose slices. Image retrieved from [21].

3.2.1. Convolutional encoder

As can be seen in Figure 3.1 the convolutional encoder passes all L 2D slices separately through a convolutional block twice. This convolutional block contains a down-sampling convolution (Chapter 2.3.1), a Group Normalization (Chapter 2.3.3) and a max pooling (Chapter 2.3.2) layer that is followed by ReLU activation (Equation 2.2). This convolutional block extracts important features like tissue boundaries and material contrasts from the input slices. The output that is produced after the second convolutional block is passed through a final convolution with K_f convolutional filters (kernels), this flattens the output into a vector of embedding dimension $D = H' \times W' \times K_f$ where H' and W' are the reduced height and width after the max pooling operations. The process that was just described is applied to all L 2D slices of the input CT image, so after the convolutional encoder, we are left with L 1-dimensional vectors of size D. We call these vectors tokens. In Figure 3.1 the beam shape wasn't implemented as a variable input yet, but for our DoTA model, there is an extra token for the beam shape that is created by passing a 24×24 image of the beam shape through the convolutional encoder. For our model, there are 151 tokens after the convolutional encoder (L + beam shape token)

3.2.2. Transformer encoder

In the causal transformer encoder, a learnable positional embedding $\boldsymbol{r}$ is added to the tokens produced by the convolutional encoder. The 0^{th} position embedding $\boldsymbol{r}_0$ is added to a new token that represents the beam energy and the 1^{st} position embedding $\boldsymbol{r}_1$ is added to the beam shape token. We now have 152 tokens (L + beam shape token + beam energy token). The transformer encoder consists of alternating MSA (Equation 2.8) and Multi-layer Perceptron (MLP) layers with residual connections and Layer Normalization (Chapter 2.3.3) is applied before every layer. MLP is a two layer feed-forward network with Dropout (Chapter 2.3.5) and GELU activations (Equation 2.3). Equation 3.1 [21] represents the operations that are carried out by the transformer encoder on the 152 tokens $\boldsymbol{z}$:

$$\boldsymbol{z'} = \boldsymbol{z} + MSA(LN(\boldsymbol{z})) + MLP(LN(\boldsymbol{z} + MSA(LN(\boldsymbol{z})))), \tag{3.1}$$

3.2.3. Convolutional decoder

The convolutional decoder has the same structure as the convolutional encoder except for the convolution and max pooling layers that are replaced with a convolution transpose layer that increases the dimensions of the input of the convolutional decoder so that the output of the DoTA model has the same dimensions as the input of the DoTA model [21].

3.3. Training the DoTA model

The DoTA model will be trained using TensorFlow with 8 samples per batch (batch size = 8), a scheduled learning rate starting at 10^{-3} that is divided by 2 after every 4 epochs. We train for 30 epochs and save the weights that minimize the error function (see Chapter 2.1.3). The number of epochs determines the number of complete passes through the training dataset. The learning rate determines how much the weights of the model change after every epoch (how quickly the model "learns").

3.4. Hyperparameter optimization

We perform hyperparameter optimization on the training data set for several values of N, K_f and N_h which represent the number of transformer blocks, the number of convolutional filters in the last convolution of the convolutional encoder and the number of attention heads respectively. We will perform this optimization by looking at the Mean Absolute Error (MAE) of the models with these different parameters.

3.5. Testing the DoTA model

We will test the DoTA model with the best parameters with the test data from the 2 patients mentioned in chapter 3.1 and compare the predicted dose distribution to the Monte Carlo calculated dose distribution by performing gamma analysis (Equation 2.11). For this project, every gamma analysis is performed with $\Delta = 1\%$ and $\delta = 3$ mm.

4

Results

4.1. Hyperparameter optimization

From Table 4.1, we see that set 2 gives the smallest MAE and thus we will analyse the results of the DoTA model that is trained with the hyperparameters of set 2.

4.2. Gamma evaluation of the DoTA model

We test the model with the data of 2 patients that is not used during training, giving 3276 output blocks. In Figure 4.1 a histogram is presented with the gamma pass rates of the DoTA model. By analysing the gamma pass rate, we compare the predicted dose to the target Monte Carlo calculated dose. In Figure 4.1 we can see that most of the output blocks have gamma pass rate between 90 and 100%. Some important values of the gamma passrate of this model are: Mean = 98.45%, Standard deviation = 2.60%, Minimum = 44.13% and Maximum = 100%. The dose difference criterion (Δ) is equal to 1 % and the distance-to-agreement (δ) is equal to 3 mm for every gamma analysis in this section.

Because the minimum passrate is very low, we will look at the dose distribution plots of some of the worst gamma pass rate blocks and try to find a pattern.

4.3. Analysing the worst performing test blocks

4.3.1. Worst performing test block

In Figure 4.2 10 first consecutive slices around the middle of the W-axis of the worst performing block, which has a gamma pass rate of 44.13%, are presented. In Figure 4.3 10 consecutive slices around the point on the L-axis where the proton beam stops propagating of the same block are presented. From Figure 4.3 we can clearly see that the beam shape was predicted very inaccurately. In the first slice in Figure 4.3, where the beam shape

Table 4.1: The DoTA model is trained with the training data for 3 sets of hyperparameters N, K_f and N_h. These values have been given in this table together with the MAE these sets produce.

	Transformer blocks (N)	Convolutional filters (K_f)	Attention heads (N_h)	MAE ($\frac{Gy}{10^9 part.}$)
Set 1	1	16	16	0.0266
Set 2	1	12	16	0.0261
Set 3	1	8	16	0.0299

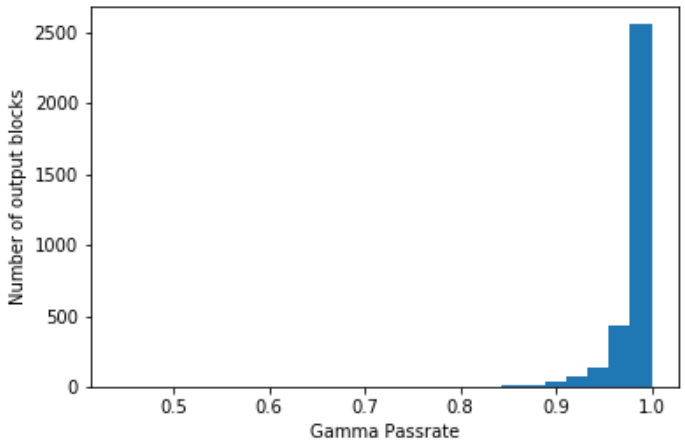


Figure 4.1: The histogram that presents the gamma pass rates of the DoTA model that is presented in section 3.2 and has been trained with the following hyperparameter set: $N = 1$, $K_f = 12$ and $N_h = 16$

still looks unaltered due to complexities in the material density, it can be seen that the target beam shape is a normal 2D Gaussian (circle) and the predicted beam shape is an ellipse that has its major axis along the W axis of the block and its minor axis along the H axis of the block. This looks like the main reason why the gamma pass rate of this block is so low.

4.3.2. Second and third worst performing test blocks

Figure 4.4 displays a slice in the middle of the W-axis for the second worst performing block, which has a gamma pass rate of 76.35%. Figure 4.5 displays a slice in the middle of the L-axis of this same block.

From Figure 4.4 we can see that the dose distribution of the second worst predicted block is way closer to the target dose distribution than the worst predicted block was. Figure 4.5 shows us that the beam shape was predicted well, as almost no voxels have a gamma value greater than 1. This confirms that the beam shape not adjusting only happens in one special case (for the worst performing block). If the beam shape didn't adjust for any other blocks, their gamma pass rate would be similar to that of the worst block.

Figure 4.6 displays a slice in the middle of the W-axis for the third worst performing block, which has a gamma pass rate of 76.73%. Figure 4.7 displays a slice in the middle of the L-axis of this same block. Figure 4.6 is plotted to look for a pattern that may cause disturbances in the predicted dose. We see that the error of this block is near the end of the block (on the L-axis/beam direction axis), right after the last part of dense tissue in the patient geometry. Since both the error of Figure 4.4 and Figure 4.6 happen in air (black area), these errors might be due to a sudden change in density that disrupts the dose calculation. Figure 4.7 shows that the beam shape was again predicted well, in the showcased slice there is not a single voxel that has a gamma value greater than 1. This again confirms that the beam shape not adjusting only happens in the case of the worst performing block.

4.4. Plots for a block with mean gamma pass rate

In this last section, plots will be shown for a block with a gamma pass rate that is close to the mean gamma pass rate of the model. This block represents the average performance of our model.

In Figure 4.8 10 consecutive slices around the middle of the W-axis of a block with a gamma pass rate of 98.69% are presented. This gamma pass rate is close to the mean gamma pass rate of 98.45%. In Figure 4.9 10 consecutive slices on the L-axis of the same block, around the position where the proton beam stops propagating, are

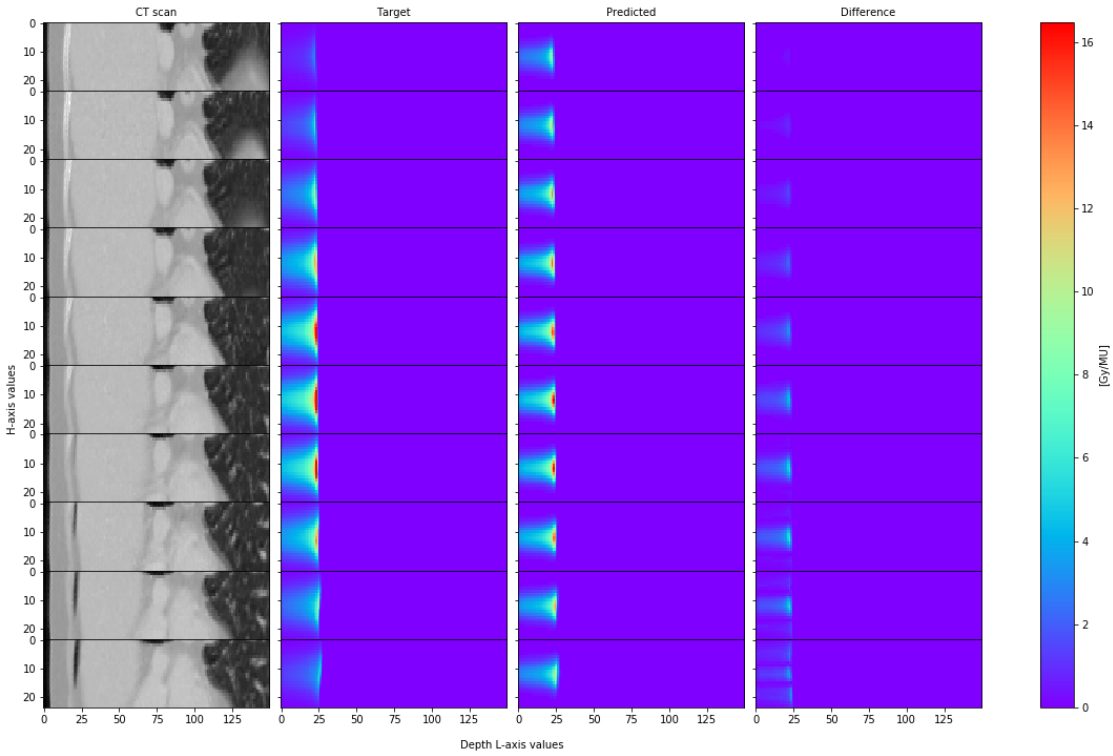


Figure 4.2: This figure showcases 10 consecutive slices around the middle of the W-axis of the worst performing block. In the first column a 2D slice of the patient geometry CT scan is showcased, in the second column the Monte Carlo calculated target proton dose distribution is showcased, in the third column the DoTA predicted proton dose distribution is showcased and in the fourth column, the absolute difference between the target dose and the predicted dose is showcased. These slices are used to analyse how the proton beam travels along the L axis.

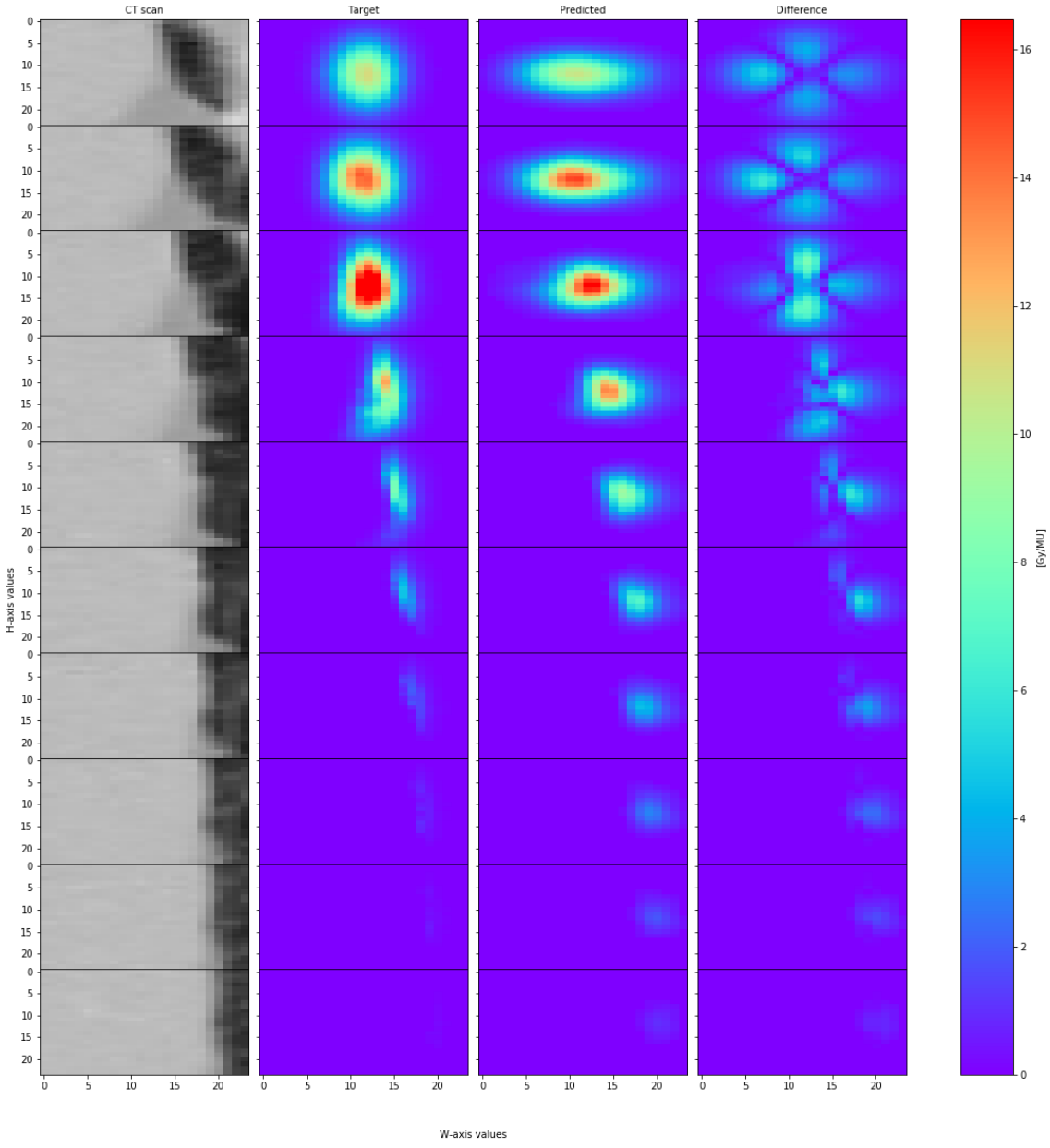


Figure 4.3: This figure showcases 10 consecutive slices on the L-axis of the worst performing block around the point where the proton beam stops propagating. In the first column a 2D slice of the patient geometry CT scan is showcased, in the second column the Monte Carlo calculated target proton dose distribution is showcased, in the third column the DoTA predicted proton dose distribution is showcased and in the fourth column, the absolute difference between the target dose and the predicted dose is showcased. These slices are used to analyse the beam shape of the proton beam at certain values on the L axis.

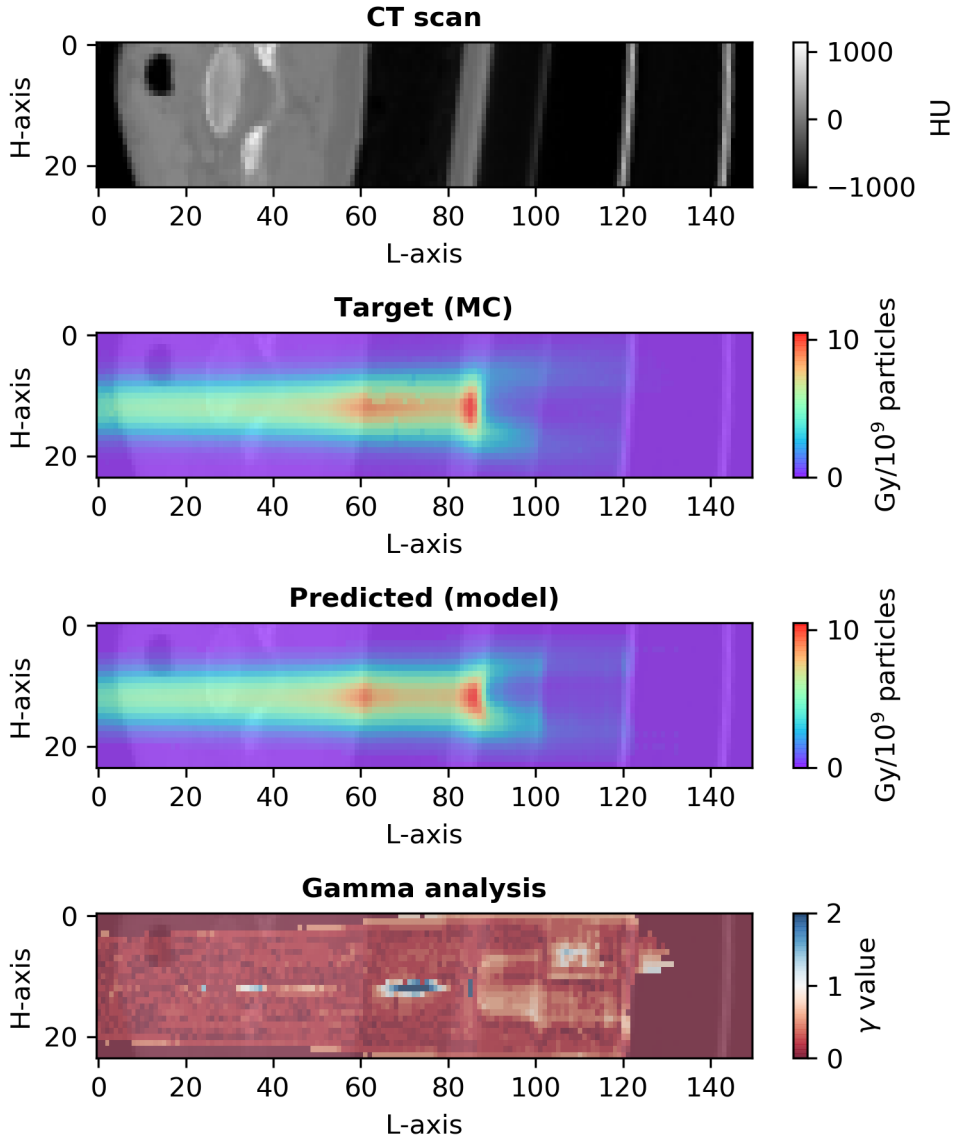


Figure 4.4: This figure showcases a slice in the middle of the W-axis of the second worst performing block. In the first (upper) image a 2D slice of the patient geometry CT scan is showcased, in the second image the Monte Carlo calculated target proton dose distribution is showcased, in the third image the DoTA predicted proton dose distribution is showcased and in the fourth image, the gamma values of the voxels of this slice are showcased. This slice is used to analyse how the proton beam travels along the L axis.

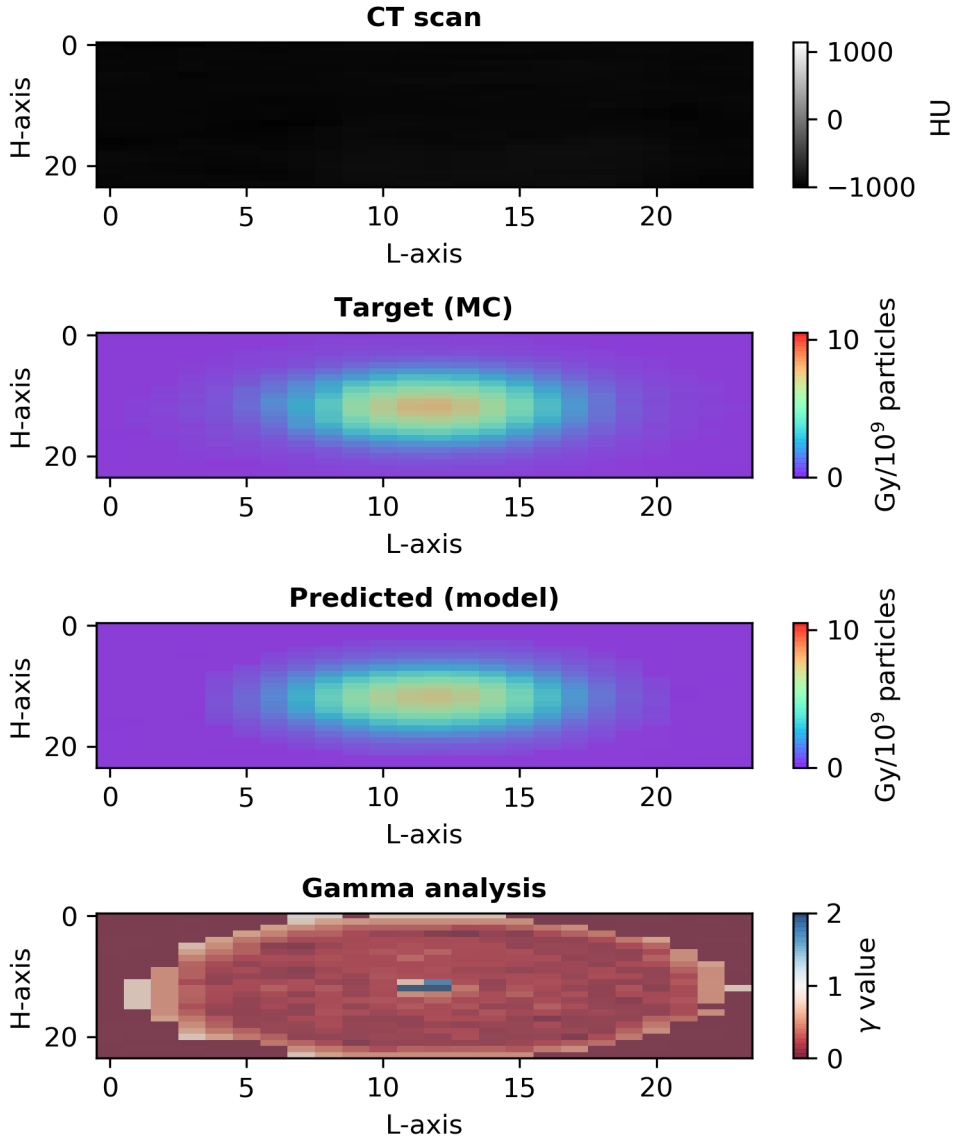


Figure 4.5: This figure showcases a slice in the middle of the L-axis of the second worst performing block. In the first (upper) image a 2D slice of the patient geometry CT scan is showcased, in the second image the Monte Carlo calculated target proton dose distribution is showcased, in the third image the DoTA predicted proton dose distribution is showcased and in the fourth image, the gamma values of the voxels of this slice are showcased. This slice is used to analyse the beam shape of the proton beam in the middle of the L axis.

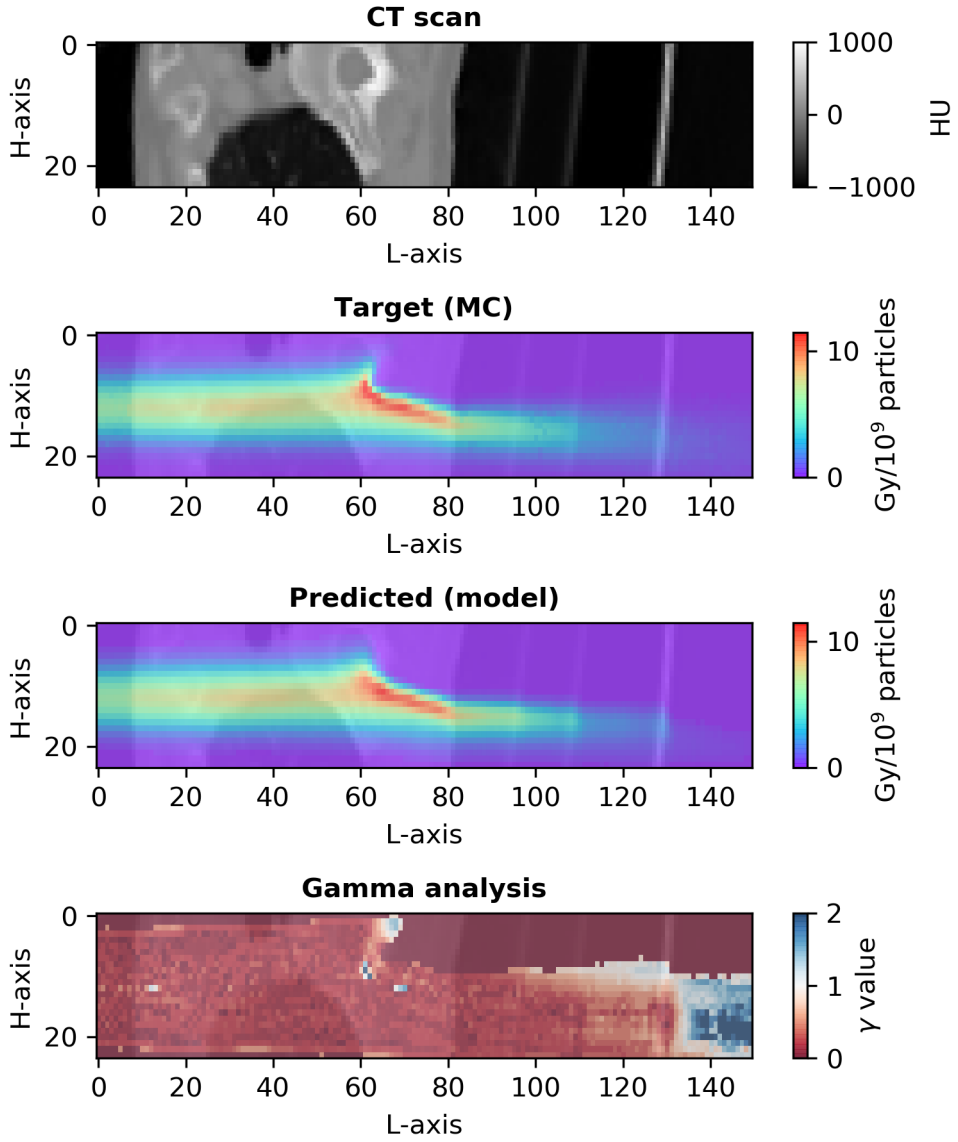


Figure 4.6: This figure showcases a slice in the middle of the W-axis of the third worst performing block. In the first (upper) image a 2D slice of the patient geometry CT scan is showcased, in the second image the Monte Carlo calculated target proton dose distribution is showcased, in the third image the DoTA predicted proton dose distribution is showcased and in the fourth image, the gamma values of the voxels of this slice are showcased. This slice is used to analyse how the proton beam travels along the L axis.

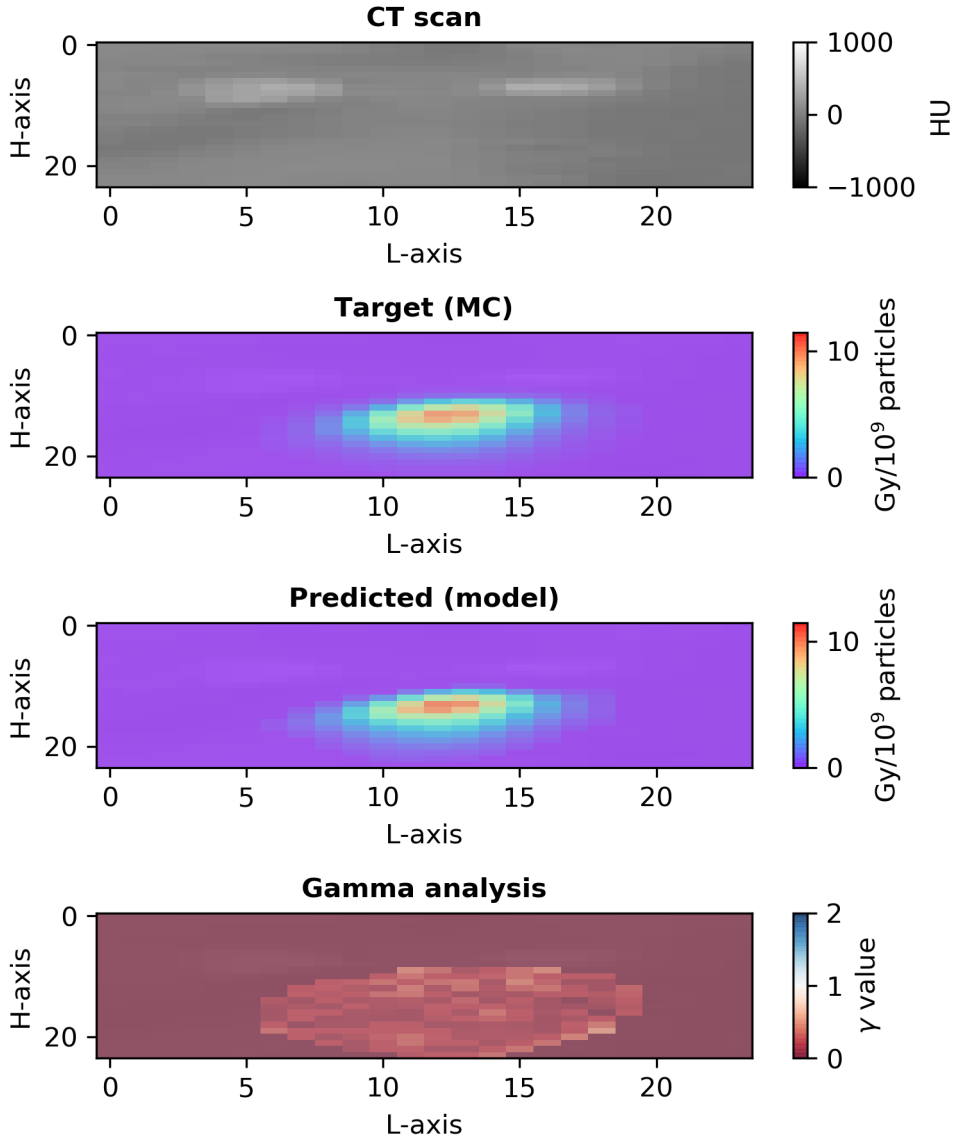


Figure 4.7: This figure showcases a slice in the middle of the L-axis of the third worst performing block. In the first (upper) image a 2D slice of the patient geometry CT scan is showcased, in the second image the Monte Carlo calculated target proton dose distribution is showcased, in the third image the DoTA predicted proton dose distribution is showcased and in the fourth image, the gamma values of the voxels of this slice are showcased. This slice is used to analyse the beam shape of the proton beam in the middle of the L axis.

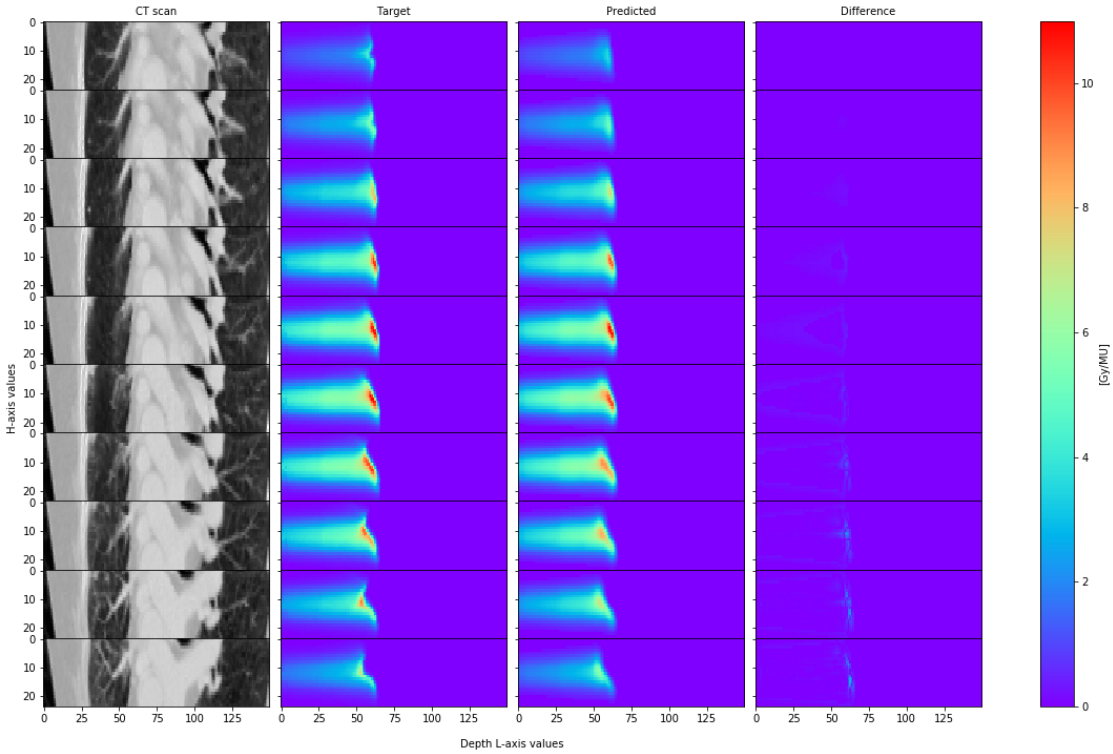


Figure 4.8: This figure showcases 10 consecutive slices around the middle of the W-axis of the block with a gamma pass rate that is close to the mean gamma pass rate. In the first column a 2D slice of the patient geometry CT scan is showcased, in the second column the Monte Carlo calculated target proton dose distribution is showcased, in the third column the DoTA predicted proton dose distribution is showcased and in the fourth column, the absolute difference between the target dose and the predicted dose is showcased. These slices are used to analyse how the proton beam travels along the L axis.

presented.

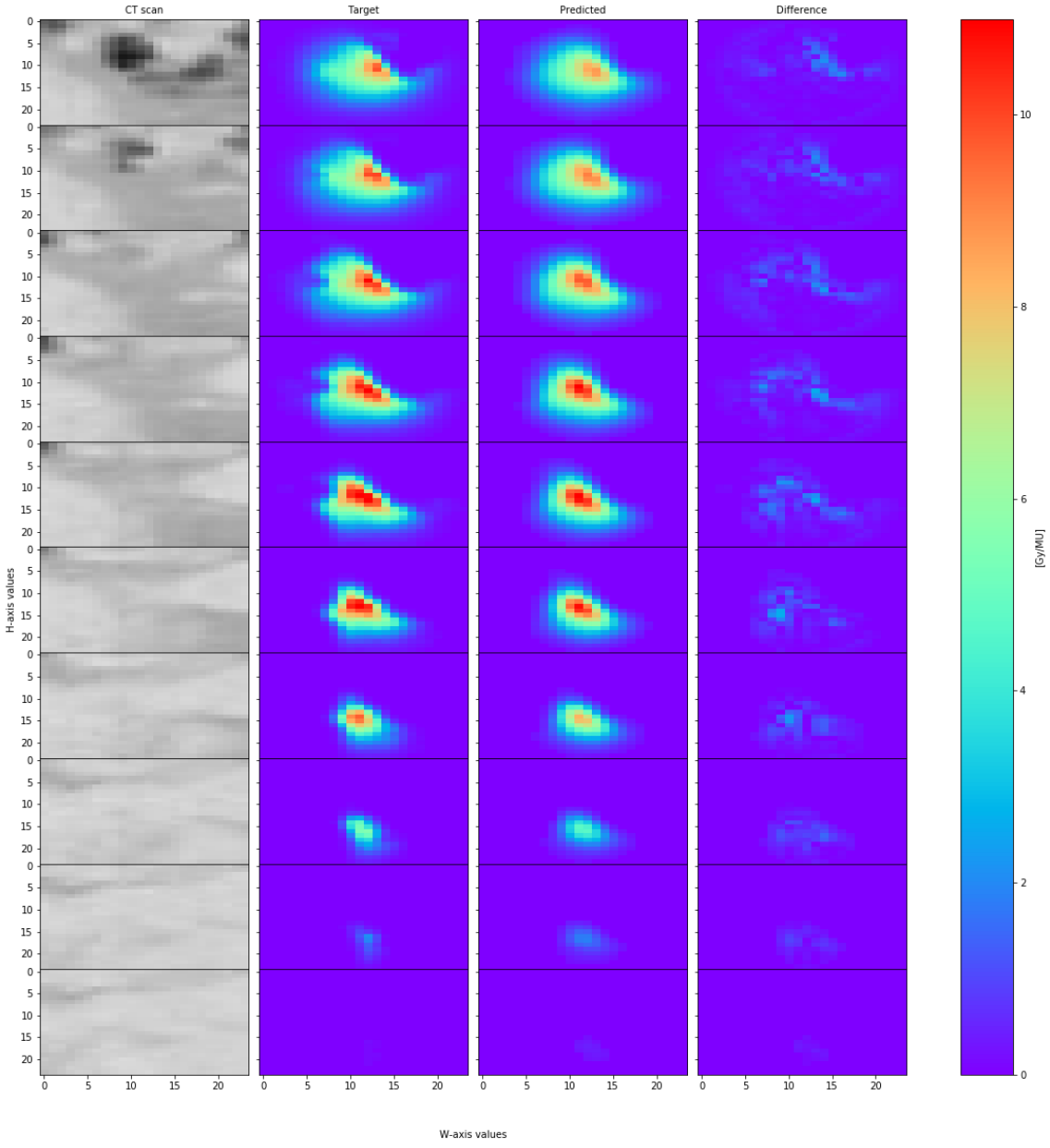


Figure 4.9: This figure showcases 10 consecutive slices on the L-axis of the block with a gamma pass rate that is close to the mean gamma pass rate around the point where the proton beam stops propagating. In the first column a 2D slice of the patient geometry CT scan is showcased, in the second column the Monte Carlo calculated target proton dose distribution is showcased, in the third column the DoTA predicted proton dose distribution is showcased and in the fourth column, the absolute difference between the target dose and the predicted dose is showcased. These slices are used to analyse the beam shape of the proton beam at certain values on the L axis.

5

Discussion

5.1. Overall DoTA model discussion

The results presented show that our DoTA model performs well overall, but struggles a bit with sudden density changes in a few cases and has smaller errors for complex geometries. In the worst test block, the patient geometry is (almost) homogeneous but the gamma pass rate is the lowest of all blocks because the beam shape does not adjust in this specific block. Despite these issues, the mean gamma pass rate is 98.45 ± 2.60 % (1 %, 3 mm) which is still good considering the fact that the beam shape can now be adjusted. The model without the beam shape variable had a multi-site gamma pass rate of 99.37 ± 1.17 % (1 %, 3 mm) [21]. Things that might have lead to a better performing DoTA model are: training with more patients, doing a more accurate hyperparameter optimization by testing more hyperparameter sets and defining a finer grid resolution. The addition of the variable beam shape to the model was successful, except for the worst performing test block, where the beam shape did not adjust.

5.2. Model use case

Since the DoTA model calculated the proton dose distribution much faster than a Monte Carlo algorithm (average of 0.3 seconds per proton beam dose distribution calculation for DoTA versus average of 20 seconds for Monte Carlo), the DoTA model can be used to speed up the dose calculation process in a proton therapy treatmant plan. If the DoTA model would run on a system with a great processing power, the calculation times could be small enough so that the model could even be used for real-time dose calculation which could be used to adapt the delivered proton dose based on sudden changes in the patient geometry like intestinal movements, coughing or breathing. This would minimize the irradiation of healthy tissue and thus minimize side effects.

5.3. Limitations

Like any deep learning model, DoTA will produce inaccurate predictions if the input is drastically different from the training data. To minimize this problem one should train the model with many different patient geometries, energies and beam shapes. Since DoTA is not a physics based model that uses physics equations to determine the dose, like Monte Carlo algorithms do, DoTA will always have problems if an input is used that is too different from the training data.

Since DoTA is modeled with training data that is generated by a machine with specific Monte Carlo settings, a DoTA model needs to be made per machine. The same goes for needing to make a model per grid resolution.

5.4. Future work

Since the DoTA model was only trained with geometries from lung cancer and head neck cancer patients, it should be trained with different and more complex patient geometries to broaden the use case of the DoTA model. An example of a more complex patient geometry is a patient with metal implants. The DoTA model can not only be applied to proton therapy, but also to the more commonly practiced photon therapy and other therapies that make use of ionizing radiation.

6

Conclusion

The DoTA model presented in this project has a mean gamma pass rate of 98.45 ± 2.60 % (1 %, 3 mm). During the project, the functionality of the earlier version of the DoTA model was improved by adding the proton beam shape as a variable input. In the previous model, only the proton beam energy and the patient geometry were variable inputs. Used with a Intel(R) Core(TM) i7-8565U CPU, the DoTA model gives an average dose calculation time of around 0.3 seconds per proton beam. The DoTA model is a lot faster than the traditional Monte Carlo dose calculation method which takes around 20 seconds to calculate the dose distribution of one proton beam when used with the same CPU. When used in a system with more processing power (for example GPU instead of CPU), the DoTA model calculation times can be decreased even more. In the paper that describes the previous version of the model [21], the one without variable beam shape implementation, a dose calculation time of 5 ± 4.9 ms was recorded. This calculation time was the result of using the previous version of the DoTA model with a Nvidia Tesla T4® GPU.

A weakness of the DoTA model is its long training time. In most cases, after finding the best hyperparameters, the DoTA model only needs to be trained once, so it isn't a major weakness.

From this conclusion and the discussions in chapter 5, it is clear that the DoTA model has the potential to evolve into a model that can be applied for real-time dose calculation and thus provide even more precision than the models that are currently used for radiotherapy treatment. In this report, the DoTA model was described for proton therapy, but it can also be used for other radiotherapy types.

References

[1] Radiotherapy Optimization Research Group at German Cancer Research Center DKFZ. matrad - an open source multi-modality radiation treatment planning system. https://e0404.github.io/matRad/.

[2] P. Baheti. Activation functions in neural networks [12 types use cases]. https://www.v7labs.com/blog/neural-networks-activation-functions, June 2022. Accessed on 2022-06-29.

[3] J. Brownlee. Overfitting and underfitting with machine learning algorithms. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. Accessed on 2022-07-15.

[4] Cancer.net. Side effects of radiation therapy. https://www.cancer.net/navigating-cancer-care/how-cancer-treated/radiation-therapy/side-effects-radiation-therapy. Accessed on 2022-07-15.

[5] Cooper University Health Care. Questions you may have about radiation therapy. https://www.cooperhealth.org/services/radiation-oncology/questions-you-may-have-about-radiation-therapy. Accessed on 2022-06-27.

[6] Université catholique de Louvain (Belgium). Commissioning procedure for mcsquare. http://www.openmcsquare.org/documentation_commissioning.html. Accessed on 2022-06-30.

[7] Stony Brook Cancer Center. Radiation therapy process. https://cancer.stonybrookmedicine.edu/RadiationTherapyProcess. Accessed on 2022-06-27.

[8] B. Clasie, H. Paganetti, and M.H. Kooy. Dose calculation algorithms. In *Proton Therapy Physics*. CRC Press, 2012.

[9] Mayo Clinic. Proton therapy. https://www.mayoclinic.org/tests-procedures/proton-therapy/about/pac-20384758. Accessed on 2022-07-15.

[10] IBM Cloud Education. Neural networks. https://www.ibm.com/cloud/learn/neural-networks, August 2020. Accessed on 2022-06-29.

[11] Y. Elcim, B. Dirican, and O. Yavas. Dosimetric comparison of pencil beam and monte carlo algorithms in conformal lung radiotherapy. *Journal of applied clinical medical physics*, 19(5):616–624, 2018.

[12] K. Souris et al. Source code of the fast monte carlo dose calculation mcsquare. https://gitlab.com/openmcsquare/MCsquare.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[14] National Cancer Institute. Radiation therapy to treat cancer. https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy, . Accessed on 2022-07-15.

[15] National Cancer Institute. Is proton therapy safer than traditional radiation? https://www.cancer.gov/news-events/cancer-currents-blog/2020/proton-therapy-safety-versus-traditional-radiation, . Accessed on 2022-07-15.

[16] Protom International. The bragg peak. https://www.protominternational.com/2018/06/bragg-peak/. Accessed on 2022-07-25.

[17] The Royal Marsden. Types of radiotherapy. https://www.royalmarsden.nhs.uk/your-care/treatments/radiotherapy/types-radiotherapy. Accessed on 2022-07-15.

[18] Johns Hopkins Medicine. Photons and protons. https://www.hopkinsmedicine.org/news/articles/photons-and-protons. Accessed on 2022-07-15.

[19] C. Misher. Radiation therapy: Which type is right for me? https://www.oncolink.org/cancer-treatment/radiation/introduction-to-radiation-therapy/radiation-therapy-which-type-is-right-for-me. Accessed on 2022-07-15.

[20] H. Paganetti. Monte carlo simulations. In *Proton Therapy Physics*. CRC Press, 2012.

[21] Z. Perkó and O. Pastor-Serrano. Millisecond speed deep learning based proton dose calculation with monte carlo accuracy. *Physics in medicine and biology*, 67(10), 2022.

[22] Connell. P. P.; Hellman. S. Advances in radiotherapy and implications for the next century: A historical perspective. *Cancer Res*, 69(2):383–391, 2009.

[23] B. Schaffner, E. Pedroni, and A. Lomax. Dose calculation models for proton treatment planning using a dynamic beam delivery system: an attempt to include density heterogeneity effects in the analytical dose calculation. *Physics in medicine and biology*, 44(1):27–41, 1999.

[24] M. Soukup and M. Alber. Influence of dose engine accuracy on the optimum dose distribution in intensity-modulated proton therapy treatment plans. *Physics in medicine and biology*, 52(3):725–740, 2007.

[25] M. Soukup, M. Fippel, and M. Alber. A pencil beam algorithm for intensity modulated proton therapy derived from monte carlo simulations. *Physics in medicine and biology*, 50(21):5089–5104, 2005.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[27] M. Tubiana. Wilhelm conrad röntgen et la découverte des rayons x. *Bulletin de l'Academie nationale de medecine*, 180(1):97–108, 1996.

[28] Cancer Research UK. Planning your external radiotherapy. https://www.cancerresearchuk.org/about-cancer/cancer-in-general/treatment/radiotherapy/external/planning/about, November 2020. Accessed on 2022-06-27.

[29] Y. Wu and K. He. Group normalization. https://arxiv.org/abs/1803.08494, 2018.