Predicting high order Reduced Order Model coefficients for proton therapy dose approximations using deep learning

Denis Bougrimov (4602269)

Delft University of Technology
Department of Radiation Science and Technology
Applied Physics

03-07-2020
Bachelor End Project
Supervisor: Dr. Zoltán Perko

Contents

Abstract

The advantage of using protons to irradiate a tumour in cancer treatment, is that the energy can be delivered very precisely to the tumour without irradiating much of the surrounding tissue. The disadvantage of this is that small displacements of a patient can result in large deviations in the planned dose delivery according to the treatment plan. Therefore, many different dose distributions corresponding to different patient displacements have to be calculated to set up a robust treatment plan, which is computationally expensive. The solution to this problem is to use a combination of reduced order modelling and deep learning to perform computationally cheaper approximations of dose distributions corresponding to patient displacements. In previous research it has been shown that shallow and narrow neural networks were unable to predict the higher order modes of the reduced order model, which are required for the desired accuracy. Therefore, in this research it was studied how a feedforward neural network's performance in predicting the higher order expansion coefficients of a reduced order model and the dose distributions depend on the neural network architecture and training approaches.

First, a dose distribution was obtained by setting up a liver tumour patient treatment plan using treatment planning software matRad. Afterwards, dose distributions corresponding to simulated 1D, 2D and 3D displacements were generated using the optimized treatment plan. The amount of training samples was set to 1000, 2000 and 3000 and the amount of test samples was set to 200, 200 and 300 for the 1D, 2D and 3D case respectively. By using the singular value decomposition on the generated dose distributions of the training set, a reduced order model could be extracted and provided the expansion coefficients that represent the behaviour of the modes within the reduced order model. Afterwards, the expansion coefficients were inspected as a function of displacement to determine which mode showed strong non-linear behaviour. This mode was called the target mode or high order mode and all modes up until the target mode were included in the reduced basis that was used to approximate the dose distributions. Mode 18, 24 and 51 were chosen as the target modes for the 1D, 2D and 3D case respectively. Subsequently, the target accuracies of the dose predictions within the test set in terms of the average voxel fraction acceptances were calculated using a threshold of 1% difference between the true voxel dose and approximated dose, relative to the maximum dose of the true dose distribution. The target accuracies of the dose predictions within the test set were 99.86%, 99.64% and 99.46% for the 1D, 2D and 3D case respectively.

In experiment 1, neural networks with varying amounts of hidden layers, neurons per hidden layer and loss functions were trained on the full set of training samples and, once trained, were used to predict the modes and dose distributions using the displacements from the test set. The amount of hidden layers and neurons per hidden layer were set to: 2, 3, 4 and 5 hidden layers with 40, 80, 120 and 160 neurons per hidden layer for the 1D-case; 2, 4 and 6 hidden layers with 100, 200, 300 and 400 neurons per hidden layer for the 2D-case; 2, 4 and 6 hidden layers with 150, 300, 450 and 600 neurons per hidden layer for the 3D-case. The chosen loss functions were the mean squared error and mean absolute error functions. Each hidden layer used the Rectified Linear Unit activation function and the Adamax optimizer was used. The highest achieved accuracies were 99.79%, 99.50% and 98.84% for the 1D, 2D and 3D case respectively. Experiment 2 was similar to experiment 1, but the networks for the 1D and 2D case were trained with half the amount of training samples. The neural networks of experiment 1 and 2 were trained with the displacements as input variables and all the corresponding expansion coefficients of the modes up until the target mode as output variables. In experiment 3, the neural networks of the 3D case were trained only with the target mode expansion coefficient instead of all expansion coefficients simultaneously. Finally, in experiment 4, a very wide but shallow neural network and a very deep but narrow neural network were compared with and without applying L2 regularization. The regularization parameter was set to 10^{-3}, 10^{-4} and 10^{-5}.

It was concluded that increasing the amount of hidden layers and neurons, up to a certain amount can improve the prediction of both the high order modes and dose distributions, after which the performance starts to deteriorate. The amount of hidden layers and neurons per layer at which this occurs is dependent on the loss function and the amount of training data. More precisely,

the performance of neural networks that utilize the mean absolute error function deteriorate faster than networks utilizing the mean squared error loss function as the amount of hidden neurons and hidden layers increases. On the other hand, the amount of training data is also important to fully represent the features of the highly non-linear modes that deeper and wider neural networks are able to learn. Deeper networks trained longer than wider networks and regularization did not improve the performance of the neural networks within the trained amount of epochs and chosen amount of hidden layers and neurons per layer.

1. Introduction

Proton therapy treatment is a type of cancer treatment that makes use of protons instead of photons to irradiate a tumour. One of the first proton treatments were performed at Berkeley radiation laboratory by using a particle accelerator that was initially intended for physics research in 1954 [1]. The advantage of using protons is that the energy can be delivered very precisely to the tumour without damaging much of the surrounding tissue. However, a disadvantage of this preciseness is that small displacements of a patient can result in large deviations in the planned dose delivery according to the treatment plan. To set up a robust treatment plan that takes all the error scenarios into account, many different dose distributions have to be calculated that correspond to different patient displacements, which is computationally expensive and thus a problem.

A method to perform computationally cheaper approximations to dose distributions corresponding to the error scenarios, is by using a combination of reduced order modelling and deep learning. Reduced order modelling can be utilized to reduce the computational complexity of a mathematical model. On the other hand, deep learning utilizes deep neural networks to find an underlying model within a dataset and, once trained, can be used to make new predictions with data that it has never seen before. Accordingly, the neural network can be trained on data from the reduced order model and, once trained, can be utilized to approximate proton dose distributions using patient displacement data it has never seen before. Previous research has been done on this topic by M. Spadaro [2], in which feedforward neural networks were utilized to predict the modes of a reduced order model which could be used to effectively approximate the dose distributions. However, the shallow and narrow neural networks that were used, were unable to predict the non-linear higher order modes, which are required for a higher accuracy in the dose approximations. Therefore, the question this research will try to answer is how a feedforward neural network's performance in predicting the higher order expansion coefficients of a reduced order model and the dose distributions depend on the neural network architecture and training approaches.

First, a treatment plan was set up which provided the initial proton dose distribution. For this research the sample data of a liver tumour patient was used to set up the treatment plan. Afterwards, dose distributions with simulated one, two and three dimensional displacements were generated using the optimized treatment plan. A reduced order model could then be extracted by using the singular value decomposition on the generated dose distributions. The reduced order model provided the expansion coefficients that represent the behaviour of the modes within the reduced order model. Moreover, the expansion coefficients were inspected as a function of displacement to determine the first mode that showed strong non-linear behaviour. This mode was called the target mode or high-order mode and all modes up until the target mode were included in the reduced basis of the reduced order model. Subsequently, the maximally achievable or target accuracy of the prediction of the doses were calculated. These steps were repeated for each displacement case.

In the first experiment, neural networks with varying depth, width and loss functions were trained on the full set of training samples and were then used to predict the higher order modes and dose distributions. This was done for each displacement case. In the second experiment, only the networks in the 1D and 2D case were trained with half the amount of training samples. The neural networks of the first and second experiment were trained with all the expansion coefficients of the modes up until the target mode. On the contrary, in the third experiment, the neural networks of the 3D case were trained only with the target mode expansion coefficient instead of all expansion coefficients simultaneously. Finally, in the fourth experiment, a wide but shallow and deep but narrow neural network were compared with and without applying regularization.

The report is structured in the following way: Chapter 2 comprises all background information and needed theory. Subsequently in Chapter 3, the data preparation and different experiments are explained. Afterwards, results are shown and discussed in Chapters 4 and 5 respectively. In Chapter 6 a conclusion is made which will be the end of the report. References are included in the last page.

2. Theory

2.1 Proton therapy treatment planning

A proton therapy treatment plan can be set up using a three dimensional computed tomography scan (CT) of a patient, in which the tumour and organs are visible. Using this CT scan, physicians delineate different volumes that are important for the proton dose calculations. First of all, the visible volume that encloses the tumour is delineated, which is called the gross target volume (GTV). Subsequently, microscopic tumours that are not visible in the CT scan are taken into account by delineating the clinical target volume (CTV), which is a slightly larger volume than the GTV. Afterwards, uncertainties concerning the movement of the tumour are factored in by delineating the planning target volume (PTV) which encloses both the GTV, CTV and a volume that is slightly larger than the CTV. Finally, volumes that represent the organs or tissue in which tumours are absent, but are important for the dose calculations, are delineated. These volumes are referred to as the organs at risk (OAR) and proton doses within these volumes should be kept as low as possible to prevent unnecessary damage to healthy organs or tissue.

After delineating all the necessary volumes, treatment planning software can be used to calculate a proton dose distribution according to the treatment plan. For this project, the treatment planning software matRad was used, which is written in Matlab and can be used within research to develop and optimize radiotherapy treatment plans [3]. MatRad already includes different sample patient data that can be used to set up a treatment plan. In the matRad graphical user interface (GUI) it is possible to load in patient data, which includes a CT scan. The CT scan can be viewed from different angles and planes and the delineation for the different volumes that are present in the CT can be turned on or off.

Proton doses can be applied by setting up the proton beams in the GUI by specifying the gantry and couch angles. The beam that is used to irradiate the tumour, is appointed an isocentre plane or grid consisting of smaller individual proton beams or beamlets [4]. Each beamlet in the grid is characterized by a weight q_j which can be used to calculate the dose in a voxel d_i via the summation given by:

$$d_i = \sum_{j=1}^{N_j} B_{ij} q_j. \tag{1}$$

N_j in Equation (1) represents the amount of beamlets and the matrix B_{ij} consists of elements of the dose that each beamlet delivers. It should be noted that a voxel is analogous to a 3D pixel. Afterwards, objective and constraint parameters can be set up in the matRad GUI to regulate the amount of dose that the different volumes receive. More specifically, the objective parameters are used to penalize over- or under dosing. The penalties are functions of the beamlet weights and are denoted by $\{p_i(\mathbf{q})\}_{i=1}^{N_p}$, where $\mathbf{q}$ represents a vector comprising all beamlet weights and N_p represents the amount of penalty functions. The constraint parameters on the other hand determine the absolute dose constraints for a particular volume and act as boundary conditions for the loss function. In the end, a treatment plan is developed by minimizing the loss function in Equation (2) given by:

$$f_{loss}(\mathbf{q}) = \sum_{i=1}^{N_p} (p_i(\mathbf{q})). \tag{2}$$

By adjusting the weights of the individual beamlets, thus changing the intensities of the beamlets, matRad minimizes the loss function. Eventually, after an optimal set of weights has been achieved, the dose volume histogram is inspected to see which volumes receive a certain dose. Judging from this dose volume histogram, objective and constraint parameters can be changed to improve the treatment plan. This process is iterated until a treatment plan is achieved in which the dose in the PTV is high enough and the dose in the various OAR volumes is as low as possible.

2.2 Singular value decomposition and the reduced basis

The singular value decomposition is a matrix decomposition method that can be used to decompose any arbitrary matrix $A \in \mathbb{R}^{m\times n}$ into a product of matrices $U \in \mathbb{R}^{m\times n}$, $\Sigma \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{n\times n}$ given by Equation (3) [6][7].

$$A = U\Sigma V^T \tag{3}$$

The columns of matrix U in Equation (3), denoted by $\{\mathbf{u_i}\}_{i=1}^{n}$, represent the orthogonal basis of the columns $\{\mathbf{a_i}\}_{i=1}^{n}$ of matrix A and Σ represents a diagonal matrix with non negative values $\sigma_1, \ldots, \sigma_n$ which are called the singular values. Moreover, the singular values are ordered in decreasing values across the diagonal. Therefore, A can be approximated most accurately using the first k basis vectors of U instead of the last k basis vectors. Correspondingly, lower order modes contribute more to approximating A correctly than the higher order modes. Subsequently, the reduced basis, or the reduced order model (ROM), is constructed by using only the first k basis vectors from U, with $k < n$ which results in the orthogonal matrix $U_k \in \mathbb{R}^{m\times k}$. This reduced basis can then be used to calculate the expansion coefficients using Equation (4).

$$U_k^T A = C \in \mathbb{R}^{k\times n} \tag{4}$$

U_k^T in Equation (4) represents the transpose of the matrix U_k. The matrix C represents the expansion coefficient matrix that consists of columns of expansion coefficients that are required to recover an approximation of any column of A through a linear combination with the column basis vectors of U_k. This linear combination is given in Equation (5).

$$\mathbf{a_i} \approx \sum_{j=1}^{k} c_{ij}\mathbf{u_j} \tag{5}$$

c_{ij} in Equation (5) represents an expansion coefficient that corresponds to the j^{th} basis vector required to recover an approximation of the i^{th} column of A. Using the expansion coefficient matrix C, an approximation of A can be made using Equation (6).

$$A_k = U_k C = U_k U_k^T A \tag{6}$$

A_k in Equation (6) is an approximation of A which is calculated using the reduced basis U_k. Since the reduced order model contains less basis vectors, calculations using this model become easier and less complex. The amount of basis vectors that are included in U_k determines how small the Froebenius norm of the error $A - A_k$ becomes. Therefore, the approximation of A gets worse as less basis vectors are included in the basis.

2.3 The basics of neural networks

Artificial Neural networks (NN) are computing systems that learn from experience, without having to be programmed according to task-specific rules. These networks are constructed to replicate an underlying model of a given dataset they are being trained on. Consequently, the trained network can be used to predict output parameters using possible input parameters within the true model that the network has never seen before. The working principle of this network is a collection of neurons that are connected to each other to transmit information analogously to a biological neural network [8]. There are many different ways in which the neurons can be connected to each other which determines how information is directed in the network. Popular examples of these topologies include: the feedforward, convolutional and recurrent neural networks, which are suitable for different tasks.

Each neuron in a neural network is connected to a set of N_s sending and N_r receiving neurons which will be denoted by $\{s_i\}_i^{N_s}$ and $\{r_i\}_i^{N_r}$. The connections, called synapses, have their own weights $w_{i,j}$ that determine how information is processed. $w_{i,j}$ represents the weight of the synapse connecting the i^{th} sending neuron with the j^{th} receiving neuron. The information from the set of sending neurons is transmitted to a single receiving neuron via the propagation function denoted by F_{prop}. This function maps the vector containing the outputs or activation values of the sending

neurons and the vector containing all the weights of the synapses between the sending and a single receiving neuron to a single scalar value z_s through:

$$z_s = F_{prop}(\mathbf{a_s}, \mathbf{w_j}, b_j) = \mathbf{w_j}^T \mathbf{a_s} + b_j. \tag{7}$$

$\mathbf{w_j} \in \mathbb{R}^{N_s}$ in Equation (7) represents the weight vector containing all the weights of the synapses between the sending neurons and the single receiving neuron denoted by $[w_{1j}, \ldots, w_{N_s j}]^T$. $\mathbf{a_s} \in \mathbb{R}^{N_s}$ represents the output vector or activation vector containing all the activation values of the sending neurons denoted by $[a_1, \ldots, a_{N_s}]^T$. b_j is a bias value that belongs to the j^{th} receiving neuron which is a value that determines how high $\mathbf{w_j}^T \mathbf{a_s}$ should be before the neuron is able to activate or output a value. The receiving neuron receives a single scalar input z_s, which is subsequently converted to an output or activation of the receiving neuron. Consequently, the receiving neuron becomes a sending neuron. This is done via an activation function F_{act}, which determines how high the activation of the neuron will be with a given input z_s. The higher the input value of the activation function, the higher the output value of the neuron. There are different activation functions like the Rectified Linear Unit, sigmoid and hyperbolic tangent functions that are more suitable for a neural network depending on the given task. The activation function is given by:

$$F_{act}(z_s) = y_{r_j} = a_{r_j}. \tag{8}$$

y_{r_j} and a_{r_j} in Equation (8) represent the output or activation of the receiving neuron. Afterwards, the output is sent to the next set of receiving neurons via a new set of weighted synapses. This process of sending and receiving information begins at the input neurons of the network and is repeated until the information has arrived at the output neurons.

2.4 Feedforward neural networks

The topology that is used in this research is called the feedforward neural network. This topology is one of the most basic topologies as the information only travels in one direction from the input to the output of the network. A schematic overview of how such a network might look like is given in Figure 1.

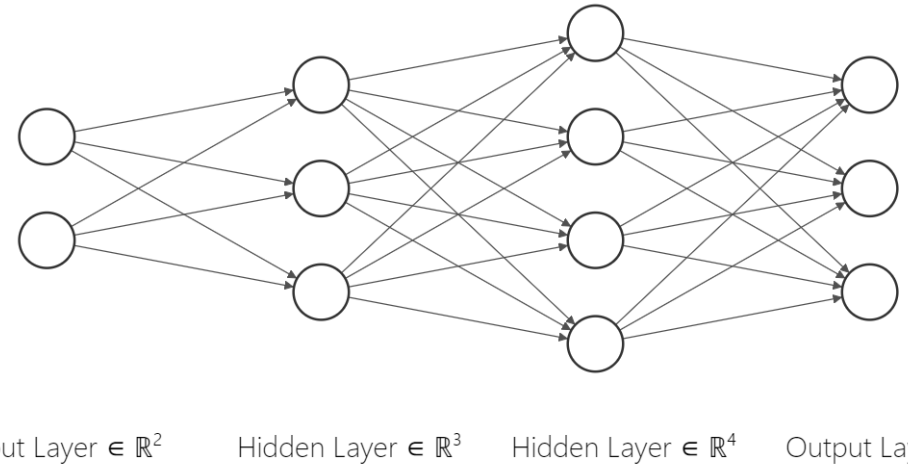

Figure 1: A schematic of a feedforward neural network with 2 hidden layers with different amounts of hidden neurons per layer. Namely, the input layer has 2 neurons, the 2 hidden layers have 3 and 4 neurons respectively and the output layer has 3 neurons. The direction of the information is given by the arrows.

The network consists of interconnected layers of neurons and each layer can have different amounts of neurons. The first layer is called the input layer where an input vector $\mathbf{x} \in \mathbb{R}^{N_i}$ is provided. N_i represents the amount of entries in the input vector or amount of input neurons. The consecutive layers are a concatenation of interconnected layers called hidden layers that can

possess different amounts of hidden neurons. The last layer is the output layer where the output vector $\mathbf{y} \in \mathbb{R}^{N_o}$ is delivered. N_o represents the amount of entries in the output vector or amount of output neurons. This simple topology is the reason why these neural networks excel at function approximation tasks, as the network can carry out an arbitrary mapping of the functions input space to the functions output space given by:

$$\mathbb{R}^{N_i} \rightarrow \mathbb{R}^{N_o}. \tag{9}$$

The amounts of hidden layers and hidden neurons per layer of the feedfoward neural network represent the depth and width respectively. An advantage of a deep network is that it is able to capture details at different levels of complexity or abstraction and therefore is able to generalize well for all the data as it learns the underlying model within the data. For instance, a network with 1 hidden layer can only approximate continuous functions, while a network with 2 hidden layers can approximate any function [5]. The reason for this is that the 2 hidden layer can represent more complex function than the 1 hidden layer network. On the other hand, the advantage of a wide network is that it has a higher capacity to memorize all the output values for a specific input value within the true model. Consequently, a deep network might need less hidden neurons per layer to represent a non linear function as it can represent more complex functions, whereas a shallow neural network needs a lot of hidden neurons per layer, as it can only represent simpler functions. To summarize, there are different methods to fit a function to a non linear behaving dataset and one method might be more accurate and efficient than the other. Additionally, it is the architecture of a neural network that will determine with what kind of functions the fit will be made. Therefore, one architecture might be better than the other, which is what will be studied in this research. A schematic of a so called deep feedforward network is given in Figure 2.

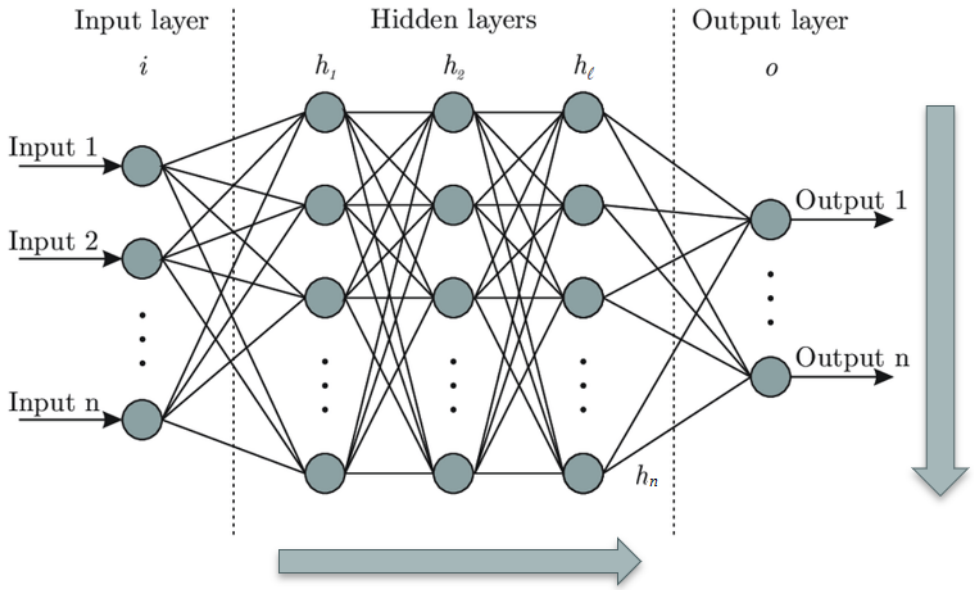

Figure 2: A schematic of a deep feedforward neural network in which the depth and width of the network depend on the amount of hidden layers and hidden nodes respectively[9].

In Figure 2, h_l represents the amount of hidden layers and h_n represents the amount of neurons per hidden layer.

2.5 Training feedforward neural networks

The goal of the neural network in the training phase is to accurately extract the underlying model that is present in a dataset. This is done by supervised learning which means that the network

is trained with a set of pregenerated training input data and their corresponding pregenerated training output data. The input and output data are denoted by $\{x_i\}_{i=1}^{N_{train}}$ and $\{y_i\}_{i=1}^{N_{train}}$ in which N_{train} represents the amount of unique training samples that is available in the training phase. The amount of training samples used per training iteration is also refered to as the batch size. With each training iteration, called an epoch, the training algorithm tries to minimize a cost function J given by:

$$J = \epsilon = \frac{1}{N_{train}} \left[\sum_{i=1}^{N_{train}} F_{loss}(y_i, \hat{y}_i(\mathbf{w}, \mathbf{b})) \right]. \tag{10}$$

ϵ in Equation (10) represents a loss function, y_i represents the expected pregenerated output and $\hat{y}_i$ represents the output generated from the neural network. $\mathbf{w}$ and $\mathbf{b}$ represent the vectors containing all the weights and biases of the neural network respectively. From Equation (10), it can be noted that the weights and biases of a neural network are the free parameters that have to be tuned to minimize the loss function and thus improve the model. There are different loss functions that can be used in a neural network and each loss function might behave differently during the training phase depending on the problem the neural networks is used for. In this research, the mean squared error (MSE) loss function was used which is given in Equation (11).

$$\epsilon_{MSE} = \frac{1}{N_{train}} \left[\sum_{i=1}^{N_{train}} (y_i - \hat{y}_i(\mathbf{w}, \mathbf{b}))^2 \right] \tag{11}$$

Another loss function that was used, is the mean absolute error (MAE) function which is given in Equation (12).

$$\epsilon_{MAE} = \frac{1}{N_{train}} \left[\sum_{i=1}^{N_{train}} |y_i - \hat{y}_i(\mathbf{w}, \mathbf{b})| \right] \tag{12}$$

A loss function indicates with each iteration how the model, captured by the neural network, performs at making predictions for a given set of new input data. The loss functions have different curves and gradients and the slope of the curves indicate how the weights and biases have to be updated to make the model more accurate. Namely, an optimization algorithm, called gradient descent, is used to minimize the loss function by moving to the minimum of the loss function as a function of the weights and biases following the steepest descent as defined by the negative of the gradient. There are different optimization algorithms like RMSProp and Adam that use different methods to update the weights and biases in a way that could speed up the gradient decent in finding the optimal weights and biases.

2.5.1 Forward propagation

In the forward propagation phase, the goal is to propagate the input through the network and eventually calculate the loss function: The algorithm that is used for forward propagation is given in Algorithm 1 [10].

Algorithm 1: Forward propagation

1 $A^{[0]} = X$ With $X \in \mathbb{R}^{N_i \times N_{train}}$ being the input sample matrix;
2 **for** $i = 1$ *to* L **do**
3 $\quad Z^{[i]} = W^{[i]} A^{[i-1]} + b^{[i]}$;
4 $\quad A^{[i]} = F_{act}^{[i]}(Z^{[i]})$ for the last layer L: $A^{[L]} = Z^{[L]}$;
5 **end**
6 $\hat{Y} = Z^{[L]} = A^{[L]}$;
7 The cost function is calculated, for example MSE: $J = \frac{1}{N_{train}} [\sum_{j=1}^{N_o} \sum_{i=1}^{N_{train}} (Y_{i,j} - \hat{Y}_{i,j})^2]$

In Algorithm 1 the superscripts indicate the number of the layer the different matrices belong to and the subscripts indicate the entries of the matrices. In line 1 an input matrix $X \in \mathbb{R}^{N_i \times N_{train}}$ is forwarded to the input layer. As the activation function of the input layer is linear, X is directly forwarded to the output of the input neuron resulting in an output or activation matrix $A^{[0]}$.

Subsequently in line 3, the activation matrix $A^{[0]}$ gets multiplied with the first weight matrix $W^{[1]} \in \mathbb{R}^{Nr \times N_i}$, after which the bias matrix is added and the resulting matrix $Z^{[1]} \in \mathbb{R}^{Nr \times N_{train}}$ is forwarded to the input of the receiving layer. It should be noted that $b^{[1]}$ is not a bias vector but a bias matrix that consists of 1 bias vector repeated N_{train} times, therefore $b^{[1]} \in \mathbb{R}^{Nr \times N_{train}}$. The input of the hidden layer then goes through the activation function after which the calculated activation is forwarded to the next hidden layer. These steps are repeated for the L layers after the input layer after which the information arrives at the output layer. As the activation of the output layer is again linear, the input of the output layer $Z^{[L]}$ equates to $A^{[L]}$ which becomes the output of the neural network denoted by $\hat{Y} \in \mathbb{R}^{N_o \times N_{train}}$ shown in line 6. In the end, the errors between the expected output $Y \in \mathbb{R}^{N_o \times N_{train}}$ and the output of the neural network denoted by $\hat{Y} \in \mathbb{R}^{N_o \times N_{train}}$ at the output neurons are added up and the average error is taken over all the training samples in line 7.

2.5.2 Backward propagation

In the backward propagation phase, the goal is to find how the weights and biases have to change in order to reduce the cost function. The steps are carried out starting from the output layer and ends at the input layer. The algorithm that is used for backward propagation is given in Algorithm 2.

Algorithm 2: Backward propagation

1 $dZ^{[L]} = dA^{[L]}$ as $Z^{[L]} = A^{[L]}$ in the output layer then $dZ^{[L]} = Y - \hat{Y}$;
2 $dW^{[L]} = \frac{1}{N_{train}}[dZ^{[L]}(A^{[L-1]})^T]$;
3 $db^{[L]} = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} dZ^{[L]}$;
4 **for** $i = L - 1$ *to* 1 **do**
5 $\quad dA^{[i]} = (W^{[i+1]})^T dZ^{[i+1]}$;
6 $\quad dZ^{[i]} = dA^{[i]} \odot F'^{[i]}_{act}(Z^{[i]})$;
7 $\quad dW^{[i]} = \frac{1}{N_{train}}[dZ^{[i]}(A^{[i-1]})^T]$;
8 $\quad db^{[i]} = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} dZ^{[i]}$;
9 **end**
10 Weights are updated using an optimizer

The first step is to calculate the differential activation matrix $dA^{[L]}$ resulting from a reduction of the loss function. This is done by taking the derivative of the cost function with respect to $A^{[L]}$ which is done in line 1. As the activation function of the output layer is linear, $dA^{[L]}$ is equal to the differential matrix $dZ^{[L]}$. Afterwards, the differential weight matrix $dW^{[L]}$ and differential bias matrix $db^{[L]}$ resulting from $dZ^{[L]}$ are calculated in lines 2 and 3. These are calculated by taking the derivatives of the Equation in line 3 of Algorithm 1.

Subsequently, the differential activation matrix of the next layer $dA^{[L-1]}$ resulting from $dZ^{[L]}$ is calculated in line 5. For this calculation the derivative of $A^{[L]}$ with respect to $Z^{[L]}$ is used. Calculating $dZ^{[L-1]}$ in layer $L-1$ is more complicated as the activation function is not linear anymore. This can be seen in line 6 where the change $dA^{[L-1]}$ is multiplied element-wise with the derivative of the activation function F'_{act}. The $\odot$ symbol represents the operator of the Hadamard product, which is an element-wise multiplication of matrices [11]. For example, the Hadamard product of a 2×2 matrix A with a 2×2 matrix B is given by:

$$A \odot B = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \odot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}\ b_{11} & a_{12}\ b_{12} \\ a_{21}\ b_{21} & a_{22}\ b_{22} \end{pmatrix}. \tag{13}$$

Subsequently, the differential weight matrix $dW^{[i]}$ and differential bias matrix $db^{[i]}$ resulting from $dZ^{[i]}$ is calculated in lines 7 and 8. This process is repeated until the input layer. In the end, if all differential weight and bias matrices have been calculated, optimizers are used to update the weight matrices and bias matrices.

2.5.3 Regularization

A common problem in training neural networks is over-fitting, which occurs when a trained model has adopted a high variance [12]. Moreover, over-fitting tends to happen in neural networks that have too many neurons and thus have free parameters that do not represent the features in the underlying model in the data, but instead over represent the noise in the data, resulting in unnecessary large weights. Over-fitting can also occur when neural networks train too long and weights get too large which make the neural network unstable. The infuence in the network of these large unwanted weights can be reduced by utilizing regularization, which is a training method that can penalize the weights or biases that become too large. There are different types of regularization like L1 and L2 regularization. L2 regularization, also called ridge regression, penalises the weight matrices from being too large by adding an extra term to the cost function in the following way:

$$J = \epsilon + \frac{\lambda}{2N_{train}} \cdot \sum_{i=1}^{L} ||W^{[i]}||^2. \tag{14}$$

In Equation (14), λ is the regularization parameter which can be tuned to determine how much the model must be penalised and $W^{[i]}$ represent the weight matrices in the neural network. The higher the regularization parameter, the lower the weight matrices will have to become in order to reduce the cost function. This will result in reducing the impact of multiple neurons in the network which makes the network much simpler. This can result in the network learning the model with a lower variance but in turn also with a higher bias.

3. Experimental Method

3.1 Generating a proton treatment plan with matRad

For this research, sample data of a liver tumour patient within matRad was utilized to set up a treatment plan. In the first step of setting up the treatment plan, different beams were configured in the matRad GUI by fixing the gantry and couch angles. Additionally, the target dose for the PTV and a penalty were specified. Subsequently, the dose distribution was calculated by matRad using Equation (1) and optimized by minimizing Equation (2). In the second step, the dose volume histogram and the physical dose distribution in an axial plane were inspected to analyze in which OAR the dose was too high. Analyzing the physical dose distribution and dose volume histogram was done in each consecutive step. Afterwards, objective parameters for the doses that the OAR volumes received, were specified, in which the penalties and squared overdosing values were fixed. In the third step, a new beam was set up with another set of couch and gantry angles, as the target dose in the PTV became too low due to the objective parameters. In the last step, constraint parameters were specified to further reduce the dose in the OAR volumes that might have received too much as a result of the added beam in the third step. In the end, a dose distribution vector was obtained that corresponded to the constructed treatment plan. Each of the steps could have been repeated to optimize the treatment plan parameters to get an even better treatment plan. However, for this research the process was terminated once an acceptable treatment plan was obtained. The dose distributions and dose volume histograms of the first and final step have been provided in the results.

3.2 Data preparation and determining the target modes

The obtained dose distribution vector from the treatment plan contained N_{vox} entries that represented the proton doses that each individual voxel received, which is denoted by $\mathbf{d}^x \in \mathbb{R}^{N_{vox}}$. Afterwards, dose distributions corresponding to extreme error scenarios in each dimension were calculated. The voxels within these dose distributions that received a dose smaller than 0.1 Gy were used to set up a dose mask. Subsequently multiple dose distribution samples were generated corresponding to simulated 1D, 2D and 3D displacements of a patient using the initial dose distribution vector. The displacement values were generated using the Latin hypercube sampling (LHS) method which arbitrarily and uniformly picks values within a certain range. The standard deviation was set at 3 mm. The dose mask was then applied to the dose distributions that removed some of the voxels that received a lower dose than 0.1 Gy. However, some voxels with a value less than the specified 0.1 Gy were not removed as they could have played an important role in the dose calculations for the extreme error scenarios and thus were not removed by the dose mask. The resulting dose distribution vectors are denoted by $\mathbf{d}^x \in \mathbb{R}^{N_{mask}}$, in which N_{mask} is the amount of voxels after applying the dose mask.

For the case of displacements in the x-direction (1D), a total of 1200 samples were generated resulting in a displacement vector $\mathbf{x}^x \in \mathbb{R}^{1200}$ and 1200 dose distribution vectors denoted by $\{\mathbf{d}_i^x\}_{i=1}^{1200}$. Each entry of the displacement vector corresponds to a single dose distribution vector $\mathbf{d}^x$. For the case of 2D and 2D displacements, a total of 2200 and 3300 samples were generated respectively. Subsequently, the entries of the displacement vectors and the dose distribution vectors were split into a training and test set and the dose distribution vectors were added as column vectors into a 2 dose distribution matrices D^{train} and D^{test}. For the 1D case, the displacement vector and dose distribution matrix of the training set is given by $\mathbf{x}^{x,train} \in \mathbb{R}^{1000}$ and $D^{x,train} \in \mathbb{R}^{N_{mask} \times 1000}$ respectively. For the test set they are denoted by: $\mathbf{x}^{x,test} \in \mathbb{R}^{200}$ and $D^{x,test} \in \mathbb{R}^{N_{mask} \times 200}$. This was repeated for the 2D and 3D cases as well, in which 2000 out of 2200 and 3000 out of 3300 samples were reserved for the training phase in the 2D and 3D case respectively.

By applying the singular value decomposition from Equation (3) on D^{train}, the orthogonal basis U was calculated. Subsequently, the reduced basis U_k was taken and the projection in

Equation (4) was used to calculate the expansion coefficient matrix C in which the entries in the rows of C corresponded to the expansion coefficients that belonged to a single mode for different displacements. Afterwards, The last k^{th} expansion coefficient row vector was plotted as a function of the displacements and inspected to see if the expansion coefficient acted highly non-linearly in terms of showing strong oscillatory behaviour. This was repeated until the first high order non-linear mode was found and was done for each displacement case. The high order mode is referred to as the target mode or mode k_{target}. There is a reason why the number of modes included in the reduced basis was not based on a certain accuracy to be reached for the dose approximation, but instead was based on which mode started to act strongly non-linear. Namely, it is not recommended to include many high-order modes within your reduced order model, as your neural network might not be able to correctly approximate even higher-order modes. This means that those extra added high order modes that are not approximated accurately by the neural network would induce extra errors within your dose approximations and also results in a more complicated model that your neural network might not be able to extract correctly from the given data.

Subsequently, the reduced basis $U^{train}_{k_{target}}$ with k_{target} basis vectors was taken from the training set and was used to calculate the expansion coefficient matrices of the training set C^{train} and the test set C^{test} using Equation (4). The approximated dose distribution matrix of the test set $D^{test}_{k_{target}}$ was then calculated using Equation (6) and $U^{train}_{k_{target}}$ and the accuracy of the approximated dose distributions was evaluated. The accuracy was determined by using the voxel fraction acceptance which represents the fraction of voxels for which $abs(\mathbf{d}^{test}_i - \mathbf{d}^{test}_{i,k_{target}}) < T \cdot max(\mathbf{d}^{test})$ holds, in which T represents the threshold. The voxel fraction acceptance for a single sample is denoted by α_i. The average voxel fraction acceptances of the approximated dose distributions of the test set were calculated using a threshold T of 1% difference between the approximated dose distribution and the true dose distribution, relative to the maximum dose of the true dose distribution. The voxel fraction acceptance α_i was calculated for each dose distribution sample after which the average was taken over all samples given by:

$$AVFA = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \alpha_i. \tag{15}$$

N_{test} in Equation (15) represents the amount of test samples and AVFA represents the average voxel fraction acceptance. The AVFA was calculated for different amounts of modes to compare the 1D, 2D and 3D case with each other regarding the amount of modes that are needed for a specific accuracy. The input data of the neural networks were the entries of the normalized displacement vector given by Equation (16).

$$\hat{x}_i = \frac{x_i}{max(abs(\mathbf{x}))} \tag{16}$$

Each entry of the displacement vector corresponds to one normalized output expansion coefficient column vector of a sample given by Equation (17).

$$\hat{\mathbf{c}}_\mathbf{i} = \frac{\mathbf{c_i}}{max(abs(\mathbf{C}))} \tag{17}$$

The expansion coefficient column vectors in Equation (17) are normalized by dividing the vectors by the maximum entry of the expansion coefficient matrix C. A neural networks that fits a model to training data that contain large values, tends to have really large weights. A model with such large weights is more unstable which results in worse performance in the training phase and high sensitivity to input values which results in higher generalization errors [13]. Therefore, normalization is applied.

3.3 Description of the experiments

The goal of the first experiment was to evaluate how a neural network with varying depth and width performed in predicting the expansion coefficients of the target mode and the dose distribution.

For the 1D case the amount of hidden layers h_l was set to 2, 3, 4 and 5 with the amount of hidden neurons per layer h_n set to: 40, 80, 120 and 160. Moreover, the amount of hidden neurons is the same in all the hidden layers. For the 2D case h_l was set to 2, 4, and 6 with h_n set to 100, 200, 300 and 400 and for the 3D case h_l was set to: 2,4 and 6 and h_n to 150, 300, 450 and 600 hidden neurons. In the first part of the experiment the MSE loss function from Equation (11) was used and in the second part the MAE loss function was used from Equation (12).

The neural networks were built using the API toolset within Keras [14], which is an open-source neural network library written in Python and runs on top of Tensorflow [15]. Subsequently, the neural networks were trained with the displacements x^{train} as input and expansion coefficient vectors $\{\hat{\mathbf{c}}_i^{train}\}_{i=1}^{N_{train}}$ as output of the training set and then used to make a prediction with the displacements of the test set x^{test} as input which resulted in the predicted expansion coefficient vectors $\{\hat{\mathbf{c}}_i^{test,NN}\}_{i=1}^{N_{test}}$. These vectors were then denormalized resulting in vectors $\{\mathbf{c}_i^{test,NN}\}_{i=1}^{N_{test}}$ and thus matrix expansion coefficient matrix $C^{test,NN}$. Afterwards, the mean absolute relative difference between the last expansion coefficient row vector of $C^{test,NN}$ and C^{test} containing the expansion coefficients of the target mode, was calculated to determine how well the high order expansion coefficients were predicted by the neural network. Subsequently, the average was taken over the mean absolute relative differences of all the modes. In the end, the predicted dose distributions $D_{k_{target}}^{test,NN}$ were calculated using $U_{k_{target}}^{train}$ and the AVFA was calculated using the condition $abs(\mathbf{d}_i^{test} - \mathbf{d}_{i,k_{target}}^{test,NN}) < T \cdot max(\mathbf{d}^{test})$ in which T is again 1%.

In the second experiment, the goal was to evaluate how the same neural networks from experiment 1 of the 1D and 2D case performed in approximating the expansion coefficients of the target mode and dose distributions when the amount of training samples was halved. More precisely, for the 1D and 2D case 500 and 1000 training samples were used instead of 1000 and 2000. A reduced basis was taken from the 500 and 1000 samples containing the same amount of modes as in experiment 1. Moreover, only the error function that showed the best results in the dose predictions in experiment 1 was used. Subsequently, the mean absolute relative difference between the last expansion coefficient row vectors, the average over the mean absolute relative differences of all the modes and the average voxel fraction acceptances were calculated and evaluated similar to experiment 1.

In the third experiment, the same neural networks as experiment 1 and 2 were trained with only the high order target mode instead of all modes simultaneously. Subsequently, only the mean absolute relative difference between the last expansion coefficient row vector of $C^{test,NN}$ and C^{test} containing the expansion coefficients of the target mode, was calculated to determine how well the high order expansion coefficient was predicted by the neural network. This was done for the 3D-case only.

In the fourth and last experiment, a deep but narrow and a wide but shallow neural network were compared. The wide and deep networks contained 2 and 20 hidden layers and 2000 and 200 hidden neurons per layer respectively. These networks were first tested with and without L2 regularization. The regularization parameter was set to 10^{-3}, 10^{-4} and 10^{-5}. The training time without applying regularization and the achieved AVFA values for the cases with and without regularization were evaluated.

To make sure the neural network did not over-train, early stopping was utilized. Early stopping uses a part of the training data as a validation set over which the loss function is evaluated alongside the loss function of the training set. Subsequently, The training phase is terminated when the early stopping condition is met, which is when the loss function over the validation set starts to increase while the loss function over the training set might still decrease. The patience parameter represents the amount of epochs over which the error over the validation set is evaluated. Accordingly, if the error over the validation set does not decrease within that specified amount of epochs, the early stopping condition is met and the training is terminated. The same patience parameter was used for all the neural networks within a displacement case. For the validation set, 25% of the training samples was used. The batch size was set to the amount of training samples that was available, which was in this case 75%. Each hidden layer used the Rectified Linear Unit (ReLu) activation function and Adamax was chosen as the optimizer.

4. Results

4.1 Treatment plan

A treatment plan was set up using the sample data of a liver tumour patient that matRad provided. In the first step of generating the treatment plan, two proton beams with gantry angles of 200° and 300° and couch angles of 0° and 0° were set up. The physical dose distribution of the first step in generating the treatment plan is given in Figure 3.

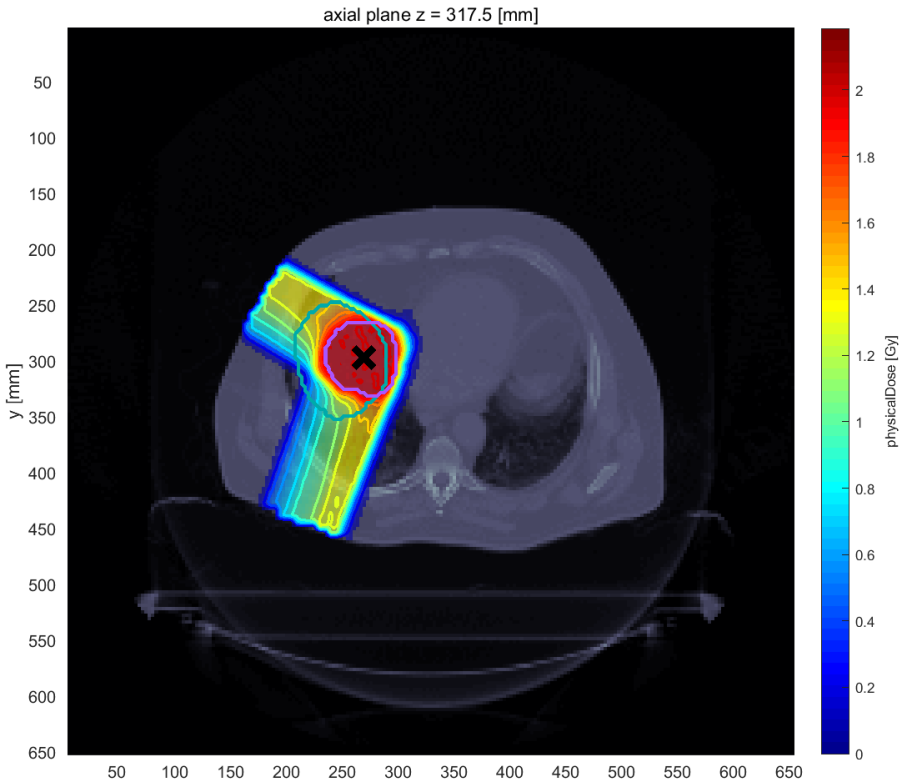

Figure 3: The physical dose distribution in an axial plane of the first step in setting up the treatment plan of a sample liver tumour patient from matRad. The green and purple delineation mark the liver and PTV respectively.

It can be observed in Figure 3 that the highest dose is applied to the PTV, which is expected. However, the applied dose outside of the PTV is still too high. This can also be deduced from the dose volume histogram given in Figure 4.

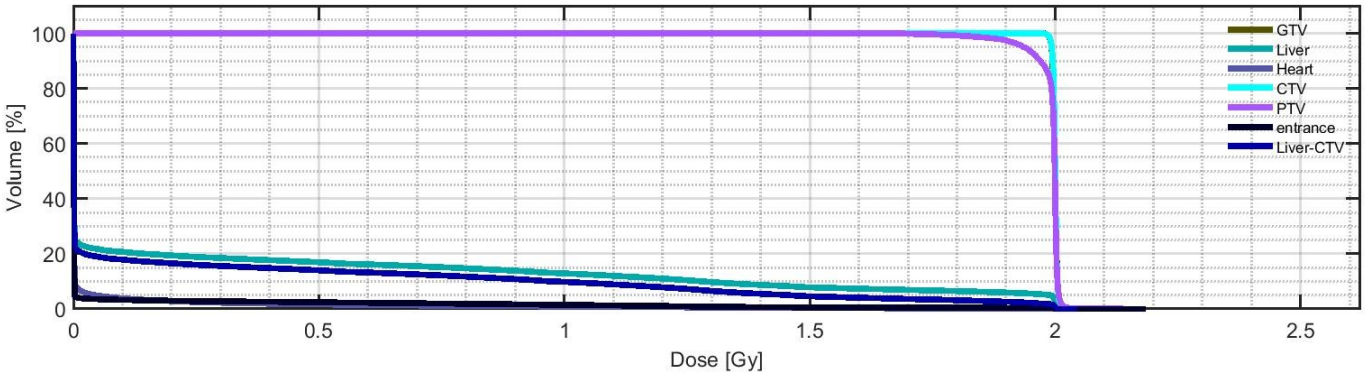

Figure 4: The dose volume histogram of the first step in setting up the treatment plan of a samples liver tumour patient from matRad showing the dose values in different volumes.

The dose volume histogram shows that unwanted dose is applied to the liver - CTV and heart which represent the OAR volumes in this case. The liver - CTV is the liver volume minus the clinical target volume. Subsequently, objective parameters were added to reduce the dose in these OAR volumes. Afterwards, a third beam with a gantry angle of 28° and a couch angle of 240° was added to increase the dose in the PTV. In the last step, constraint parameters were added to further reduce the dose in the OAR volume. The target dose, objective parameters and constraint parameters are shown in Table 1.

Table 1: The target dose, objective and constraint parameters that were used to set up the treatment plan for the sample liver tumour patient from matRad.

Volume	type	OP	objective/constraint	penalty	D(Gy)
PTV	target	1	squared deviation	1000	60
Heart	OAR	2	max mean dose constraint	---	34
Heart	OAR	2	squared overdosing	200	20
Liver - CTV	OAR	2	squared overdosing	200	32
Right kidney	OAR	2	max mean dose constraint	---	18

In the final dose distribution in Figure 5, it can be observed that the dose in the surrounding tissue has substantially reduced compared to Figure 3.

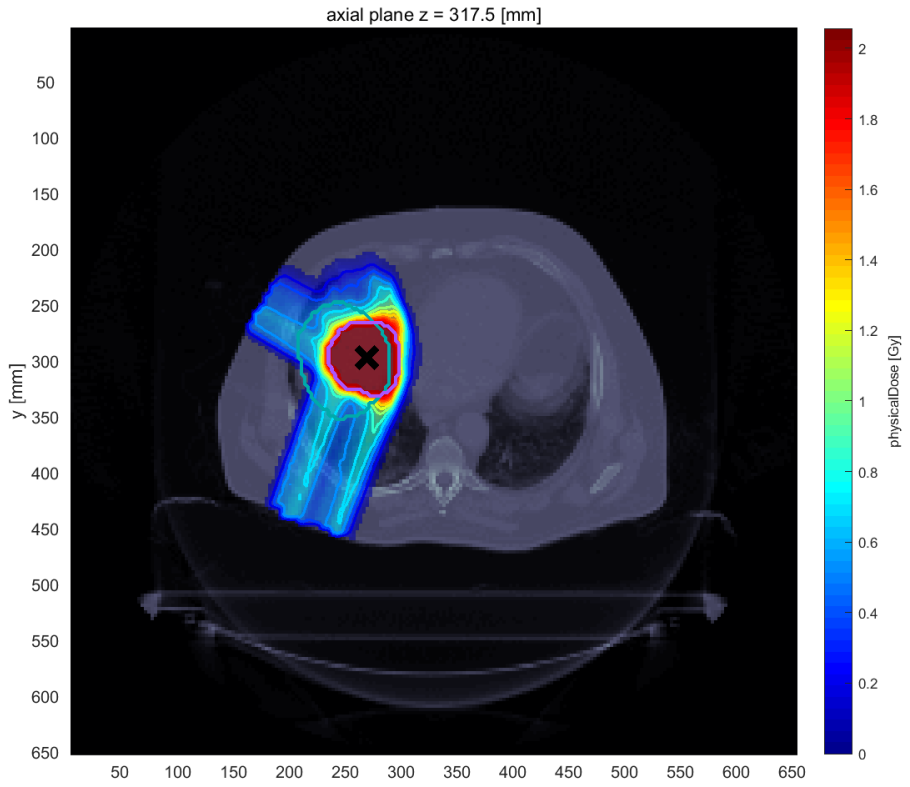

Figure 5: The physical dose distribution in an axial plane of the final step in setting up the treatment plan of a sample liver tumour patient from matRad. The green and purple delineation mark the liver and PTV respectively.

The reduction of proton dose in the surrounding tissue can also be verified using the final dose volume histogram given in Figure 6.

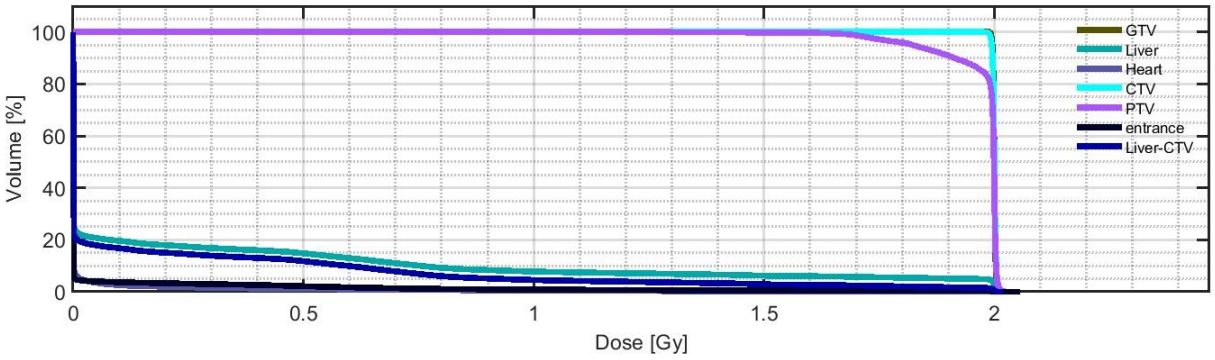

Figure 6: The dose volume histogram of the final step in setting up the treatment plan of a samples liver tumour patient from matRad showing the dose values in different volumes.

The dose in the liver-CTV and heart are slightly reduced compared to the first step as a smaller percentage of the volume received a dose higher than 0.75 Gy compared the dose volume histogram of the first step in Figure 4. It should also be noted that the dose in the PTV also decreased, which is caused by adding the objective and constraint parameters.

4.2 Determining target modes and target accuracies.

The target mode was chosen by analyzing the expansion coefficients as a function of the displacements. The first highly non-linear mode for the 1D case was mode 18 for which the expansion coefficients as a function of the displacements are given in Figure 7.

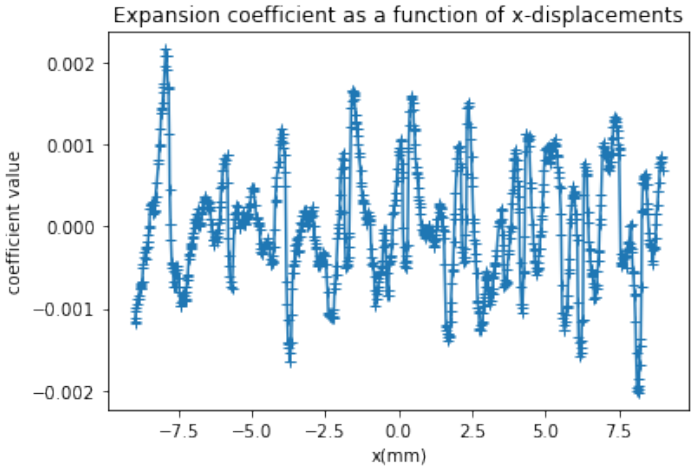

Figure 7: Expansion coefficients 18 as a function of x-displacements. The 1000 LHS sampled x-displacements from the train set have been plotted.

In Figure 7 it can be observed that the expansion coefficients of the target mode is highly non-linear as a function of displacements. Subsequently, the AVFA of the test set as a function of modes was calculated using a threshold value $T = 1\%$. The AVFA as a function of modes for the 1D case is given in Figure 8.

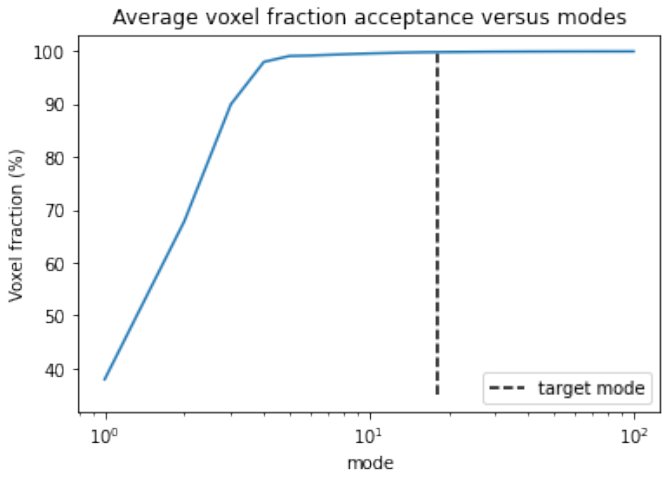

Figure 8: The average voxel fraction acceptance as a function of modes using a threshold of $T = 1\%$ for the 1D case. The test set corresponding to the 200 Latin hypercube sampled x-displacements with a standard deviation of 3 mm was used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using all modes up until the target mode with the 200 samples of the test is 99.86%.

It can be noted that more modes are required in order to achieve a higher AVFA. Moreover, the lower order modes contribute more to the AVFA than the higher order modes which was expected from the theory. Likewise, the first highly non-linear modes for the 2D and 3D case were mode 24 and 51 for which the expansion coefficients as a function of the displacements are given in Figures 9 and 11 respectively. The AVFA as a function of modes for the 2D and 3D case are plotted in Figures 10 and 12.

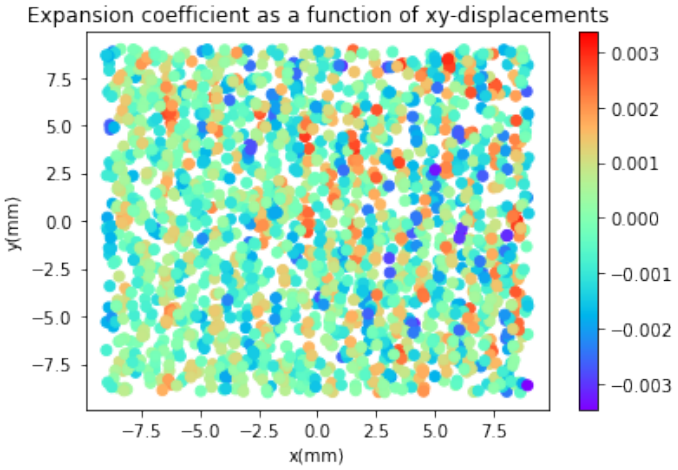

Figure 9: Expansion coefficients 24 as a function of x and y-displacements. The 2000 LHS sampled x and y-displacements from the train set have been plotted.

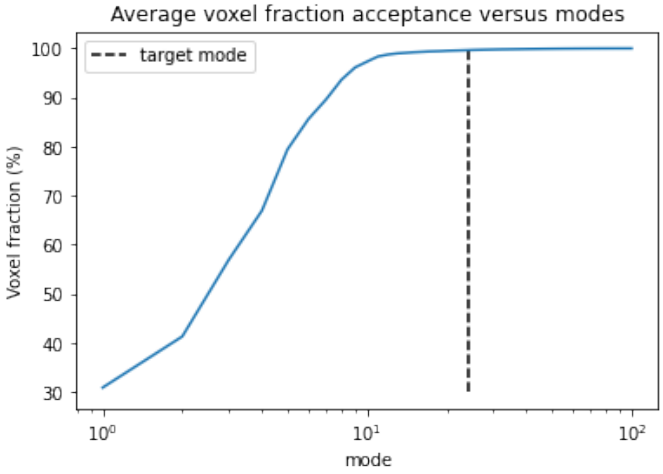

Figure 10: The average voxel fraction acceptance as a function of modes using a threshold of $T = 1\%$ for the 2D case. The test set corresponding to the 200 Latin hypercube sampled x and y-displacements with a standard deviation of 3 mm was used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using all modes up until the target mode with the 200 samples of the test is 99.64%.

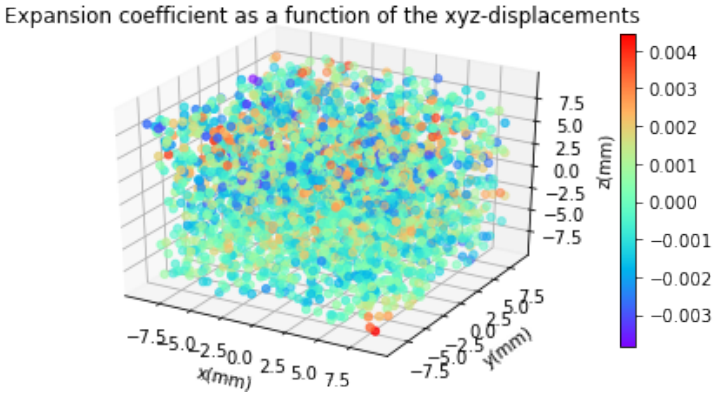

Figure 11: Expansion coefficients 51 as a function of the x,y and z-displacements. The 3000 LHS sampled x, y and z-displacements from the train set have been plotted.

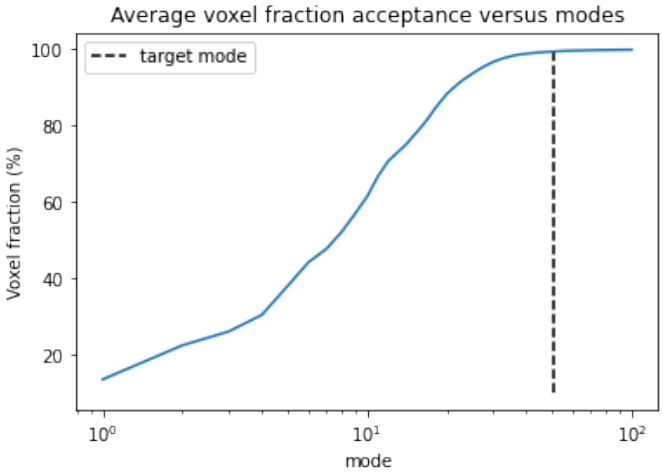

Figure 12: The average voxel fraction acceptance as a function of modes using a threshold of $T = 1\%$ for the 3D case. The test set corresponding to the 300 Latin hypercube sampled x, y and z-displacements with a standard deviation of 3 mm was used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using all modes up until the target mode with the 300 samples of the test is 99.46%.

It can be noted that more modes in the 3D case are needed for a certain AVFA compared to the 1D and 2D case. For example, for the 1D case only 18 modes are needed to reach an AVFA of 99.86% while for the 3D case 51 modes are needed for an AVFA of 99.46%. To summarize, all target modes and achievable AVFA values for the different displacement cases have been provided in Table 2.

Table 2: The target modes, amounts of samples and achievable average voxel fraction acceptance values for the one, two and three dimensional cases are given. The average voxel fraction acceptances were calculated using a threshold T of 1% difference between the approximated dose distribution and the true dose distribution, relative to the maximum dose of the true dose distribution.

Dimension	k_{target}	N^{train}	N^{test}	$AVFA_{max}(\%)$
1D	18	1000	200	99.86
2D	24	2000	200	99.64
3D	51	3000	300	99.46

4.3 Experiment 1: Neural networks with varying length, depth and loss functions.

The goal of experiment 1 was to evaluate how a neural network with varying depth, width and loss functions performed in predicting the expansion coefficients of the target mode and the dose distributions. This was done for the 1D, 2D and 3D case. First, the prediction of the expansion coefficients of the target modes are evaluated for each neural network setup. Afterwards, in order to determine how the neural network performed in predicting the dose distributions, the AVFA values of the predicted set are analyzed as well.

4.3.1 Results of the 1D-case

The most accurate prediction of the expansion coefficients of the target mode (mode 18) from a network that used the MSE and MAE function are given in Figures 13 and 14 respectively.

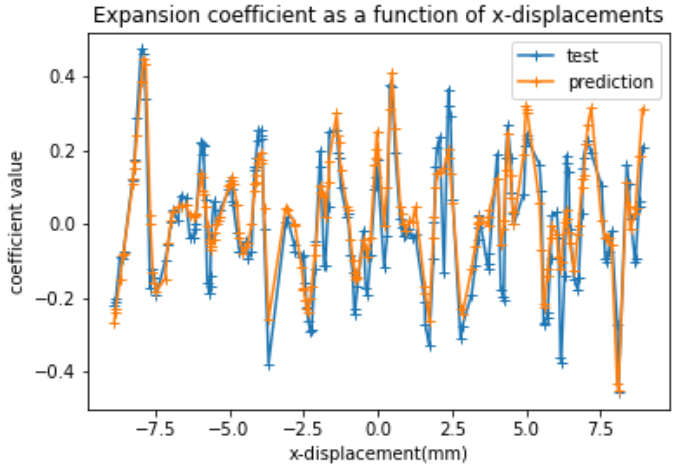

Figure 13: The best prediction of the expansion coefficients of the target mode as a function of x-displacements. The network contained 4 hidden layers and 160 hidden neurons and used the mean squared error function. The mean absolute relative difference between the test set and predicted set is 14.4%. The networks trained with all 18 expansion coefficients and used 1000 samples in the training phase. The test set and predicted set included 200 samples.

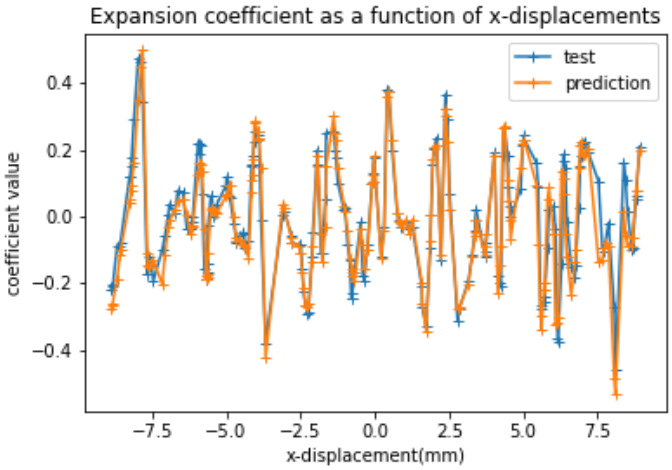

Figure 14: The best prediction of the expansion coefficients of the target mode as a function of x-displacements. The network contained 5 hidden layers and 120 hidden neurons and used the mean absolute error function. The mean absolute relative difference between the test set and predicted set is 9.5%. The networks trained with all 18 expansion coefficients and used 1000 samples in the training phase. The test set and predicted set included 200 samples.

It can be observed in Figures 13 and 14 that the prediction made by the network that utilized the MAE loss function is more accurate than the network that utilized the MSE loss function. It should be noted that the networks that used MAE required 30% less epochs on average for the early stopping condition to be met. The mean of the absolute relative differences of the target mode for each neural network are given in Figures 15 and 16.

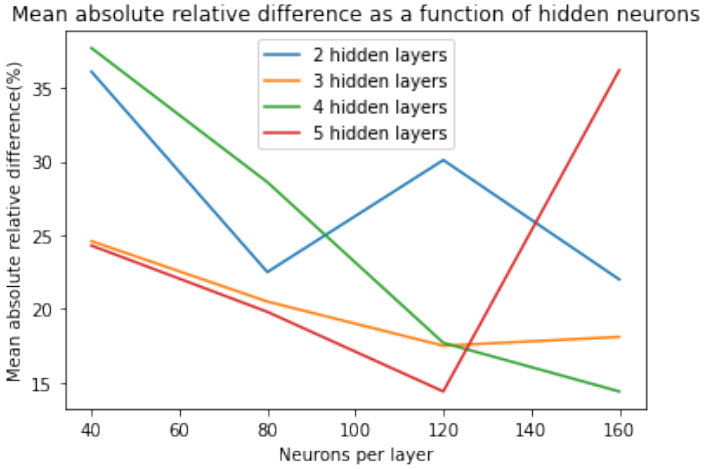

Figure 15: The mean of the absolute relative differences between the expansion coefficients of the test set and predicted set for high order mode 18 as a function of hidden neurons for the 1D-case. The networks used the mean squared error function and trained with all 18 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

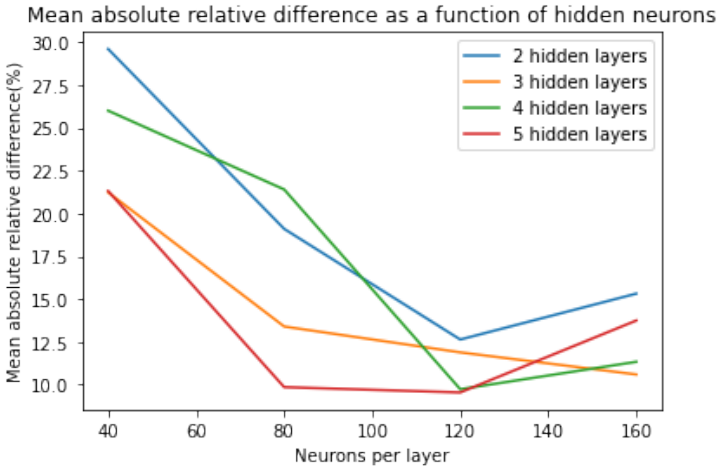

Figure 16: The mean of the absolute relative differences between the expansion coefficients of the test set and predicted set for high order mode 18 as a function of hidden neurons for the 1D-case. The networks used the mean absolute error function and trained with all 18 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

It can be observed that the networks utilizing MAE predicted the target mode more accurately than the networks that utilized MSE. Moreover, the prediction made by networks containing 3, 4 and 5 hidden layers, does get more accurate as more neurons are added up until 120 neurons per layer after which the prediction gets worse. Furthermore, it is noticeable that the accuracy of the prediction increases when the amount of hidden layers increases from 2 to 3 and 2 to 5 but is less consistent when comparing the networks with 2 and 4 hidden layers. Subsequently, the average over all the mean absolute relative differences of all the modes for all networks have been plotted in Figures 17 and 18.

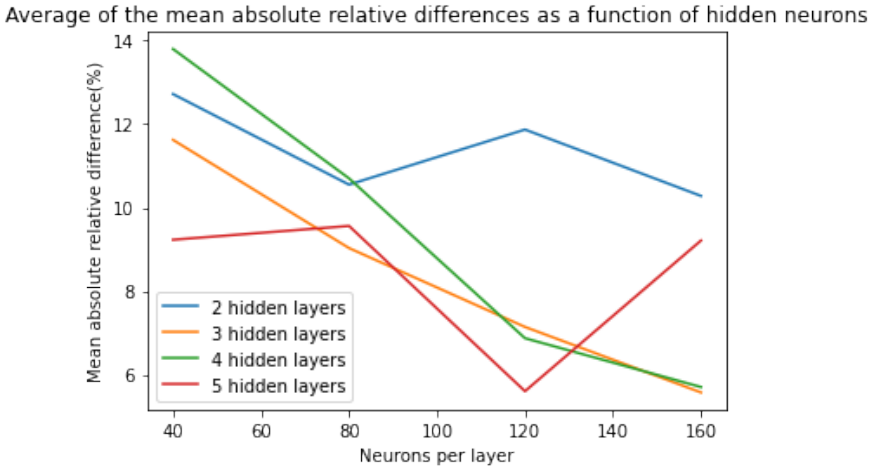

Figure 17: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 1D-case. The networks used the mean squared error function and trained with all 18 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

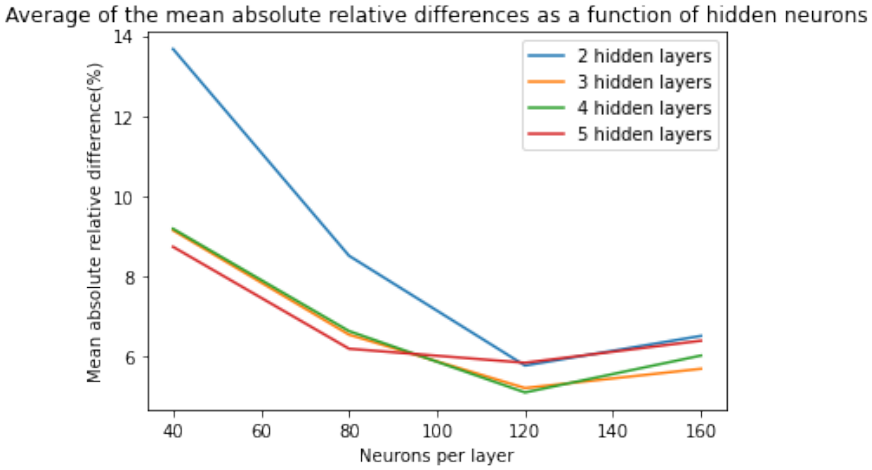

Figure 18: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 1D-case. The networks used the mean absolute error function and trained with all 18 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

It can be noted that the average prediction of the expansion coefficients improves as more neurons or layers are added to the network up until 120 neurons per layer, just as the prediction of the higher order mode in Figures 15 and 16. However, the average values are almost equal on average for networks using MSE and MAE, as opposed to the mean absolute relative difference of the higher order mode. Afterwards, the voxel fraction acceptances are given in Figures 19 and 20.

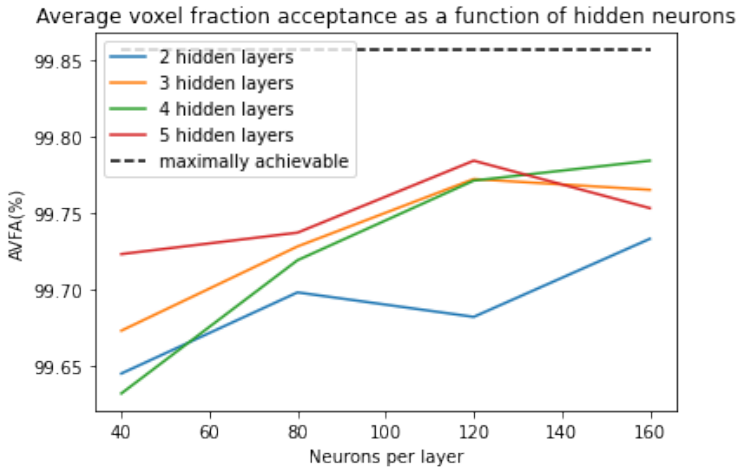

Figure 19: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 1D-case. The 200 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 18 modes with 200 test samples is 99.86%. The networks used the mean squared error function and trained with all 18 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

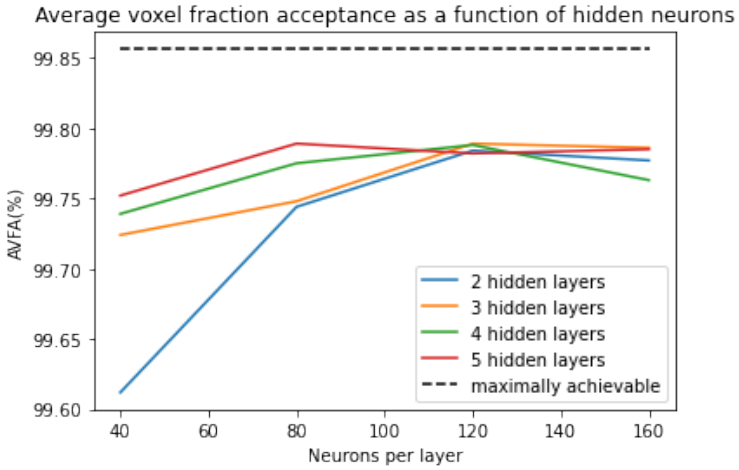

Figure 20: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 1D-case. The 200 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 18 modes with 200 test samples is 99.86%. The networks used the mean absolute error function and trained with all 18 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

First, it can be noted from Figures 19 and 20 that the differences in the achieved AVFA between the networks that used the MSE and MAE functions are small. Namely, the highest achieved AVFA of a network that used MSE is 99.78% which is lower than the achieved AVFA of a network that used MAE of 99.79%. These values were achieved with neural networks containing 4 and 5 hidden layers, and 160 and 80 neurons respectively. Another observation is that small networks utilizing MAE predict the dose more accurately than small networks utilizing MSE. As an example, the AVFA values achieved by the networks utilizing MAE with only 40 and 80 hidden neurons per layer are higher than the AVFA values achieved by networks utilizing MSE. Furthermore, it can be observed that neural networks also predict the dose more accurately as the amount of hidden layers and neurons per layer increases up until 120 neurons per layer. This can be related to overfitting.

4.3.2 Results of the 2D-case

The most accurate predictions of the expansion coefficients of the target mode (mode 24) from a network that used the MSE and MAE function are given in Figure 21 and 22 respectively.

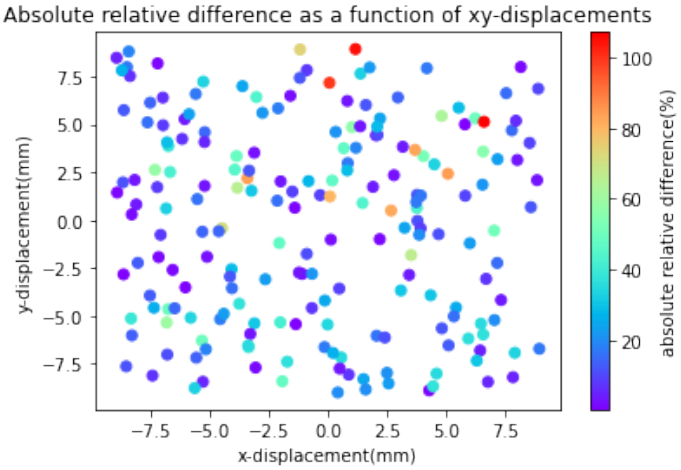

Figure 21: The best prediction in terms of the absolute relative difference, of expansion coefficients 24 as a function of x and y-displacements. The network contained 4 hidden layers and 300 hidden neurons per layer and used the mean squared error function. The mean absolute relative difference between the test set and predicted set is 23.3% . The networks trained with all 24 expansion coefficients and used 2000 samples in the training phase. The test set and predicted set included 200 samples.

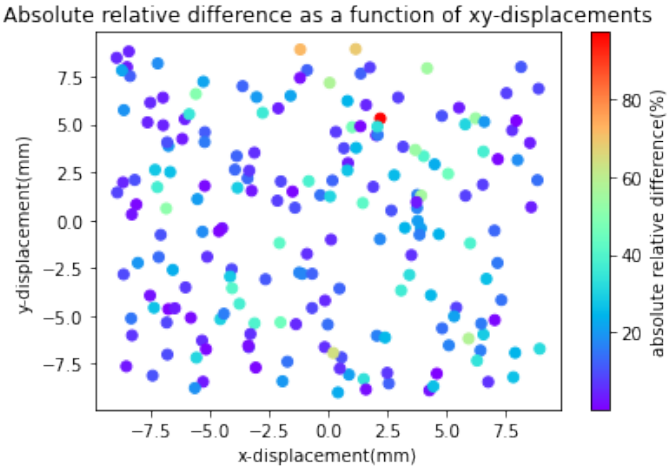

Figure 22: The best prediction in terms of the absolute relative difference, of expansion coefficients 24 as a function of x and y-displacements. The network contained 4 hidden layers and 300 hidden neurons per layer and used the mean absolute error function. The mean absolute relative difference between the test set and predicted set is 17.9%. The networks trained with all 24 expansion coefficients and used 2000 samples in the training phase. The test set and predicted set included 200 samples.

In Figures 21 and 22 it can be seen that the prediction of the target modes is not as accurate as the prediction of the target mode in the 1D-case, as the mean absolute relative differences given in Figures 21 and 22 are higher than in Figures 13 and 14. The mean absolute relative differences of the target mode as a function of hidden layers and neurons are given in Figures 23 and 24.

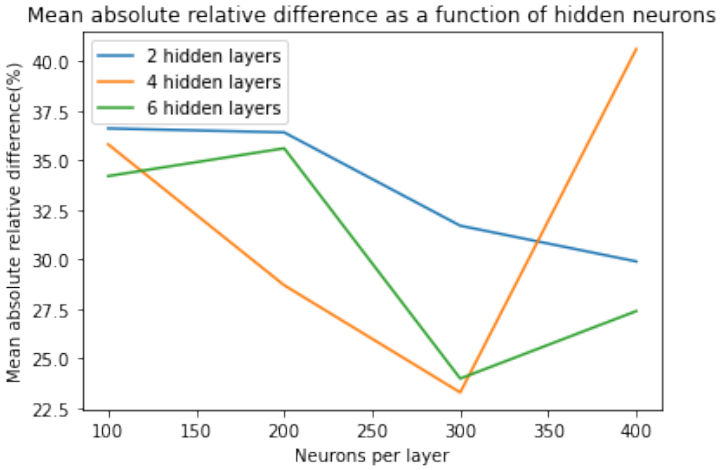

Figure 23: The mean of the absolute relative differences between expansion coefficients of the test set and predicted set for the high order mode 24 as a function of hidden neurons using for the 2D-case. The networks used the mean squared error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

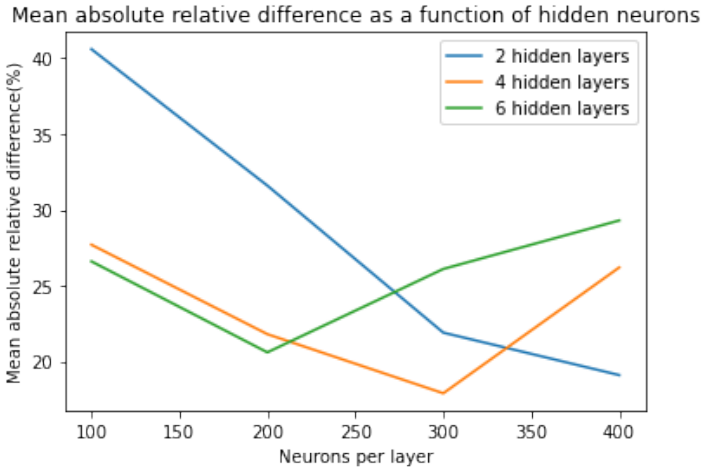

Figure 24: The mean of the absolute relative differences between expansion coefficients of the test set and predicted set for the high order mode 24 as a function of hidden neurons for the 2D-case. The networks used the mean absolute error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

It can be observed that the networks utilizing MAE predicted the target mode more accurately than the networks that utilized MSE. Furthermore, in Figure 23 it can be seen that the prediction made by networks that used MSE containing 4 and 6 hidden layers are more accurate than the predictions being made by the networks containing 2 hidden layers up until 300 neurons per layer. Moreover, the prediction made by the 4 hidden layer network that used MAE in Figure 24 starts to deteriorate at 300 hidden neurons while the prediction made by networks containing 6 hidden layers deteriorates at 200 hidden neurons. Since the target mode was not predicted sufficiently well, lower order modes were also analyzed to see which mode was correctly predicted. As an example, the mean absolute relative differences of lower order mode 22 are given in Figures 25 and 26.

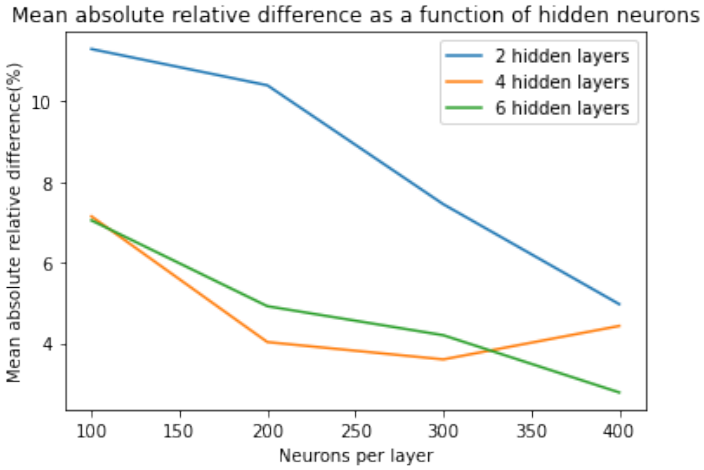

Figure 25: The mean of the absolute relative differences between expansion coefficients of the test set and predicted set for the low order mode 22 as a function of hidden neurons for the 2D-case. The networks used the mean squared error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

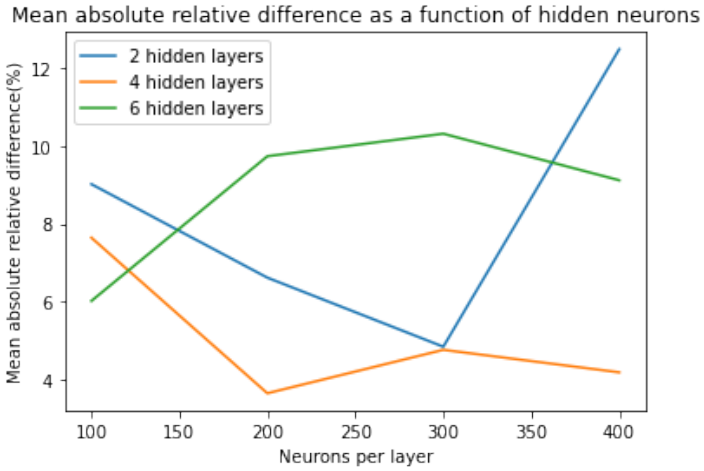

Figure 26: The mean of the absolute relative differences between expansion coefficients of the test set and predicted set for the low order mode 22 as a function of hidden neurons for the 2D-case. The networks used the mean absolute error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

In Figures 25 and 26 it can be observed that the prediction of the lower order mode is more accurate than the prediction of the higher order mode in Figures 23 and 24. Furthermore, it can be seen in Figure 25 that the prediction made by the neural networks containing 4 and 6 hidden layers using MSE are more accurate than the prediction of the 2 hidden layer network as the mean absolute relative differences are lower. On the other hand, it can be seen in Figure 26 that the prediction of the expansion coefficients of the lower order mode made by the neural networks that used MAE get less accurate as the amount of hidden layers increases from 4 to 6, just as in the prediction of the higher order mode in Figure 24. Subsequently, the average over all the mean absolute relative differences of all the modes as a function of hidden neurons has been plotted in Figures 27 and 28.

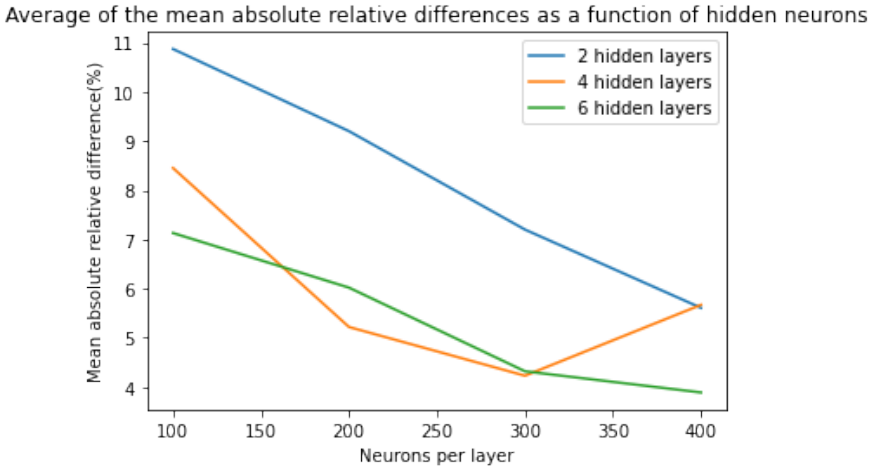

Figure 27: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 2D-case. The networks used the mean squared error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

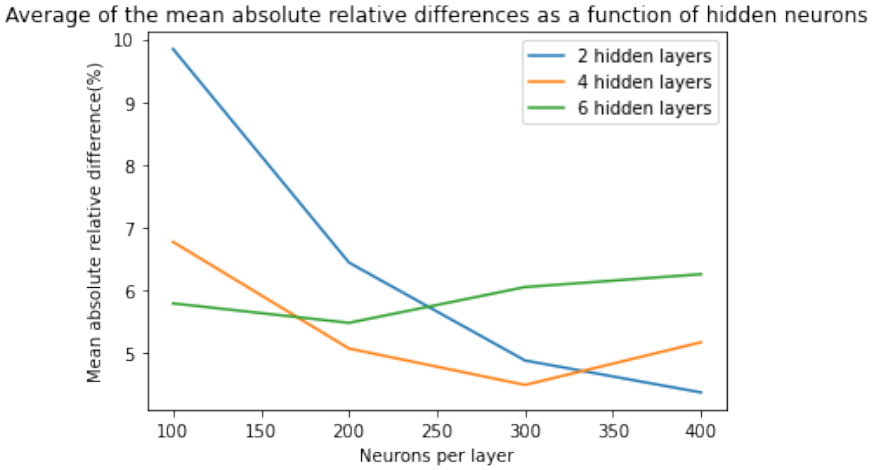

Figure 28: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 2D-case. The networks used the mean absolute error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

Similar behaviour as in the prediction of the higher and lower order modes can be seen in the average over all the mean absolute relative differences in Figure 27 and 28. Namely, networks using MSE with 4 and 6 hidden layers predict the expansion coeffiients more accurately than networks with 2 hidden layers systematically up until 300 neurons per layer. Moreover, it can be seen that the predictions of the expansion coefficients of networks that used MAE get less accurate as the amount of hidden neurons increases up until a certain amount, depending on the amount of hidden layers. This is also similar to the prediction of the higher and lower order mode. Subsequently, the AVFA values are shown in Figures 29 and 30.

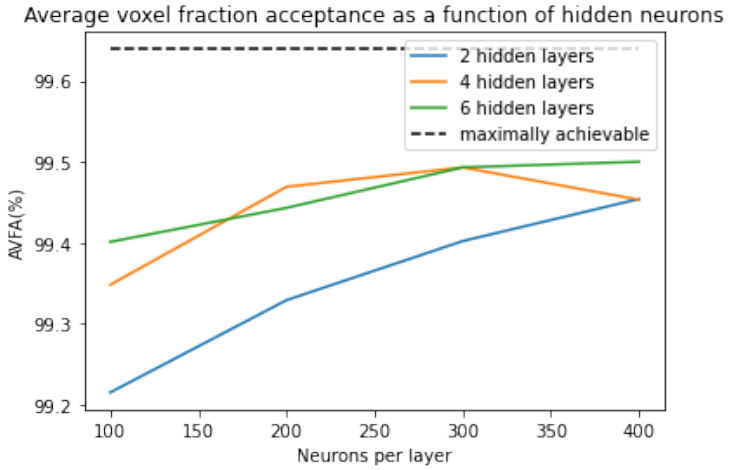

Figure 29: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 2D case. The 200 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 24 modes with 200 test samples is 99.64%. The networks used the mean squared error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

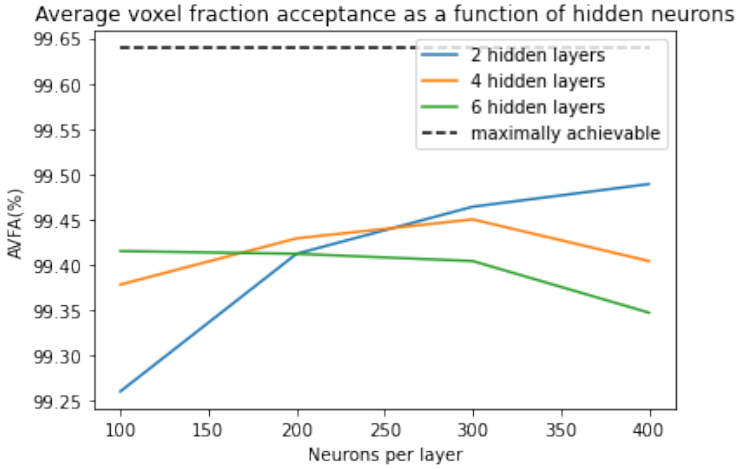

Figure 30: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 2D case. The 200 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 24 modes with 200 test samples is 99.64%. The networks used the mean absolute error function and trained with all 24 expansion coefficients and 2000 samples in the training phase. The test set and predicted set included 200 samples.

The differences in the AVFA values in Figures 29 and 30 between the case where MSE and MAE were used as loss functions are small, similar to the 1D-case. Namely, the highest achieved AVFA using MSE is 99.50% which is slightly higher than the achieved AVFA of 99.49% when using MAE. These AVFA values were achieved by networks containing 6 and 2 hidden layers and 400 hidden neurons respectively. Another observation is that the dose prediction made by the neural networks using MSE containing 4 and 6 hidden layers is again more accurate than the dose predictions made by networks using 2 hidden layers, just as in the prediction of the expansion coefficients. Furthermore, it can be observed that networks that used MAE perform worse in predicting the dose as the amount of hidden layers and neurons per layer increase, similar to the prediction of the expansion coefficients.

4.3.3 Results of the 3D-case

The mean absolute relative differences of the target mode as a function of hidden layers and neurons are given in Figures 31 and 32.

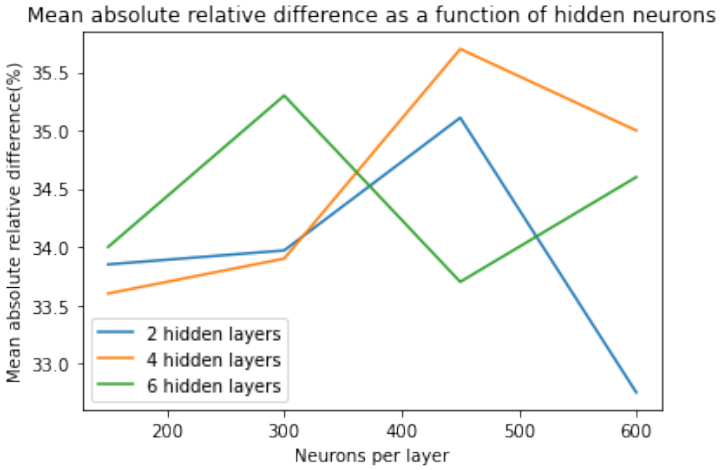

Figure 31: The mean absolute relative differences between expansion coefficients of the test set and predicted set for the high order mode 51 as a function of hidden neurons for the 3D-case. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

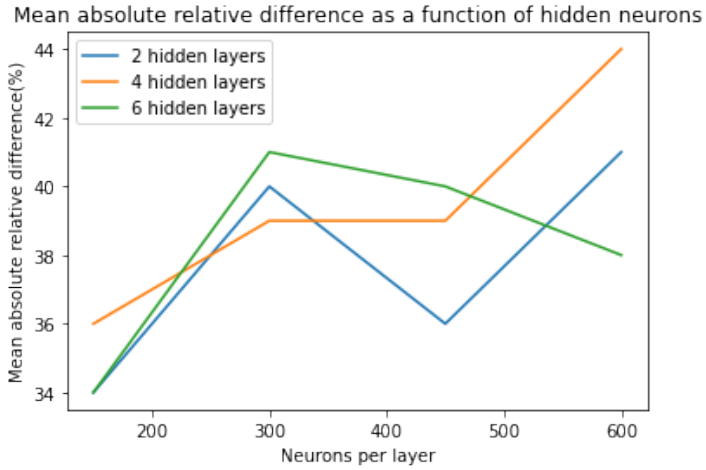

Figure 32: The mean absolute relative differences between expansion coefficients of the test set and predicted set for the high order mode 51 as a function of hidden neurons for the 3D-case. The networks used the mean absolute error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

In Figures 31 and 32 it can be seen that all chosen networks act inconsistently and failed in predicting the target mode. Another observation is that the higher order mode was predicted more accurately by networks using the MSE function. Since the target mode was not predicted sufficiently well, lower order modes were also analyzed to see which mode was correctly predicted. As an example, The mean absolute relative differences of lower order mode 48 is given in Figures 33 and 34 using MSE and MAE respectively.

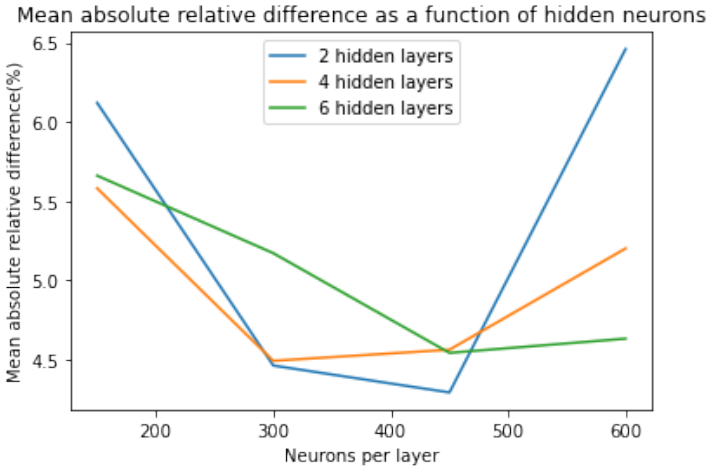

Figure 33: The mean absolute relative differences between expansion coefficients of the test set and predicted set for the low order mode 48 as a function of hidden neurons for the 3D-case. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

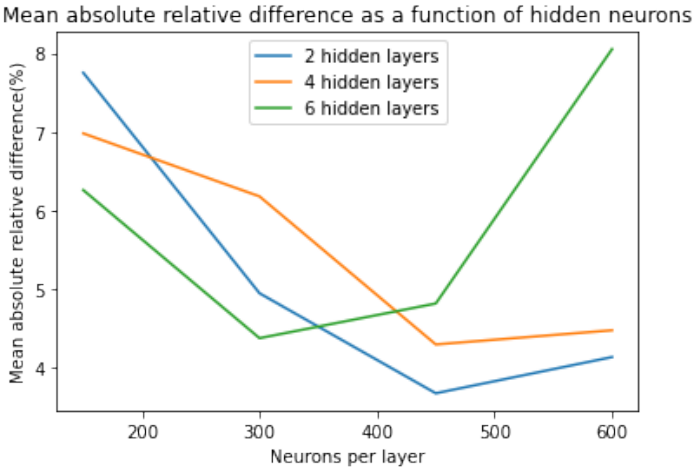

Figure 34: The mean absolute relative differences between expansion coefficients of the test set and predicted set for the low order mode 48 as a function of hidden neurons for the 3D-case. The networks used the mean absolute error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

It can be observed that the prediction of the lower order modes in Figures 33 and 34 are both more accurate than the predictions of the higher order mode in Figures 31 and 32. Moreover, it can be observed in Figure 34 that the performance of networks using MAE deteriorates as the amount of hidden layers increases from 2 to 4 and 6 hidden layers. It is difficult to state the same for the networks using MSE, as the predictions are not consistent in Figure 33. Furthermore, it can be noted that the 2 hidden layer networks achieved the best prediction of the lower order mode, which is an unexpected result compared to the 1D and 2D case in which deeper networks gave the most accurate predictions. Subsequently, the average over all the mean absolute relative differences of all the modes as a function of hidden neurons has been plotted in Figures 35 and 36.

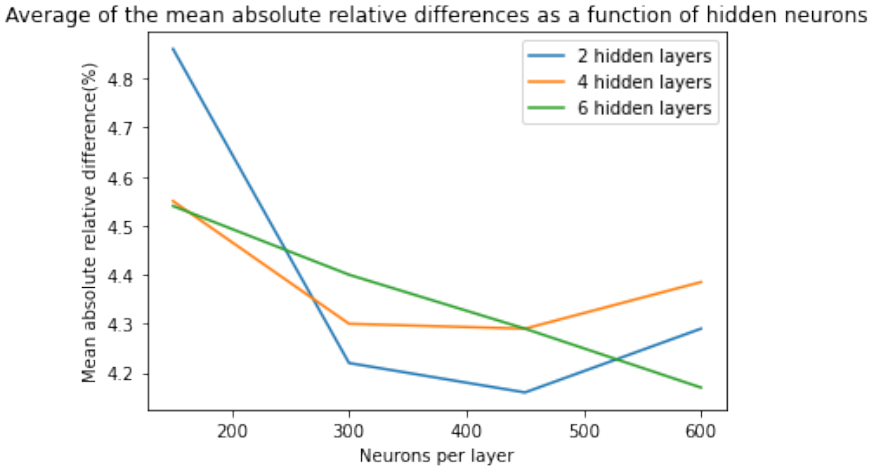

Figure 35: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 3D-case. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

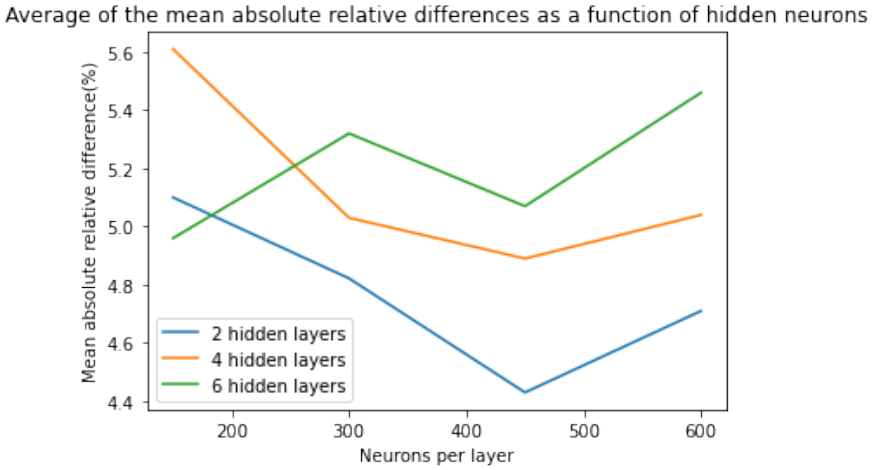

Figure 36: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 3D-case. The networks used the mean absolute error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

Similar behaviour as in the prediction of the lower order mode can be seen in the average over all the mean absolute relative differences in Figures 35 and 36. Namely, it can be seen that the prediction of the expansion coefficients made by networks using MAE gets less accurate as the amount of hidden layers increase. More precisely, networks with 2 hidden layers performed better in predicting the expansion coefficients than networks with 4 and 6 hidden layers. This is again hard to say for networks using MSE. However, it can be stated that the predictions of the expansion coefficients are done more accurately by networks using the MSE function compared to networks using MAE. Subsequently, the AVFA values are shown in Figures 37 and 38.

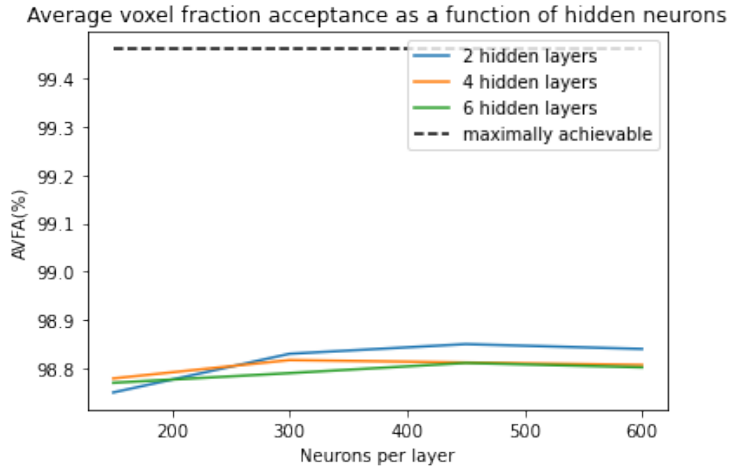

Figure 37: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 3D case. The 300 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 51 modes with 200 test samples is 99.46%. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

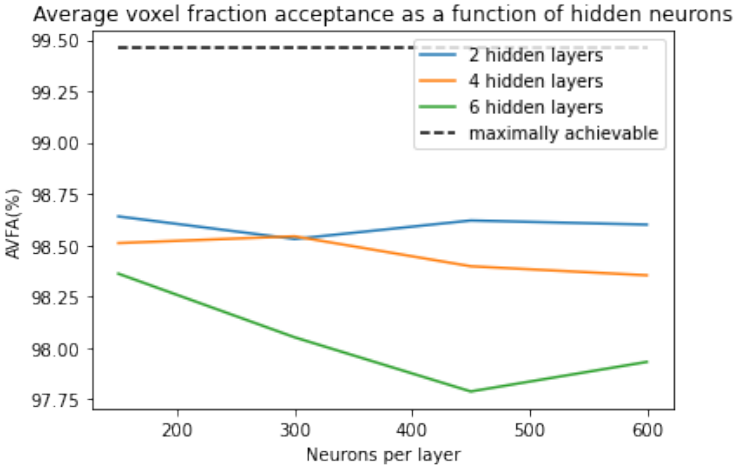

Figure 38: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 3D case. The 300 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 51 modes with 200 test samples is 99.46%. The networks used the mean absolute error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

The difference in the AVFA values in Figures 37 and 38 between the case where MSE and MAE were used as loss functions, is again small. Namely, the highest achieved AVFA using MSE is 98.84% which is higher than the achieved AVFA of 99.74% when using MAE. These AVFA values were achieved by a network containing 2 hidden layers and 400 and 600 hidden neurons per layer respectively, which is an unexpected result compared to the 1D and 2D case in which deeper neural networks gave the highest dose predictions. This could be related to the amount of training samples. Furthermore, it can be observed that networks that used MAE perform worse in predicting the dose as the amount of hidden layers and neurons increase, similar to the 2D case and comparable to the 1D-case where smaller networks using MAE performed better than smaller networks using MSE.

4.4 Experiment 2: Determining the dependence on the samples size of the training set for the 1D and 2D case.

The goal of experiment 2 was similar to experiment 1. However, the amount of training samples that the networks were trained with were halved. For the 1D-case, 500 training samples were used and for 2D-case, 1000 training samples were used to train the network with. MAE will be used for the 1D case, since the best dose prediction was achieved in experiment 1 using that loss function. Similarly, MSE was used for the 2D case instead. The mean absolute relative difference for the target mode is given in Figure 39.

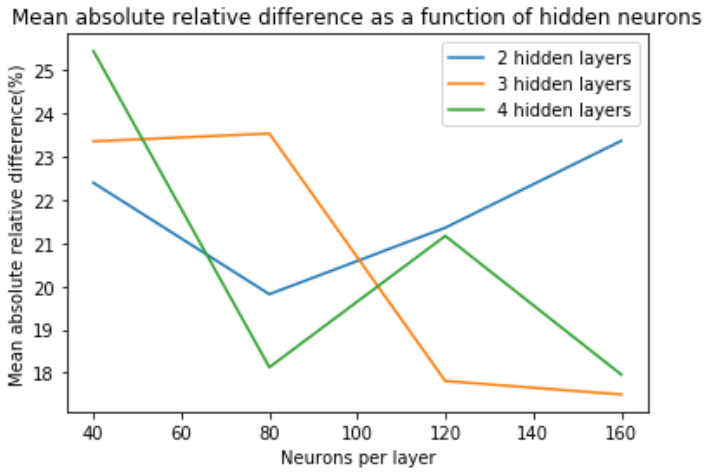

Figure 39: The mean absolute relative differences between expansion coefficients of the test set and predicted set for the high order mode 18 as a function of hidden neurons for the 1D-case. The networks used the mean absolute error function and trained with all 18 expansion coefficients and 500 samples in the training phase. The test set and predicted set included 200 samples.

From Figure 39 it can be deduced that the predictions of the target mode are less accurate and less consistent than the predictions made in experiment 1 in Figure 16. Namely, the mean absolute relative differences are twice as large on average compared to the values in Figure 16. Subsequently, the average over the mean absolute relative differences of all the modes can be seen in Figure 40.

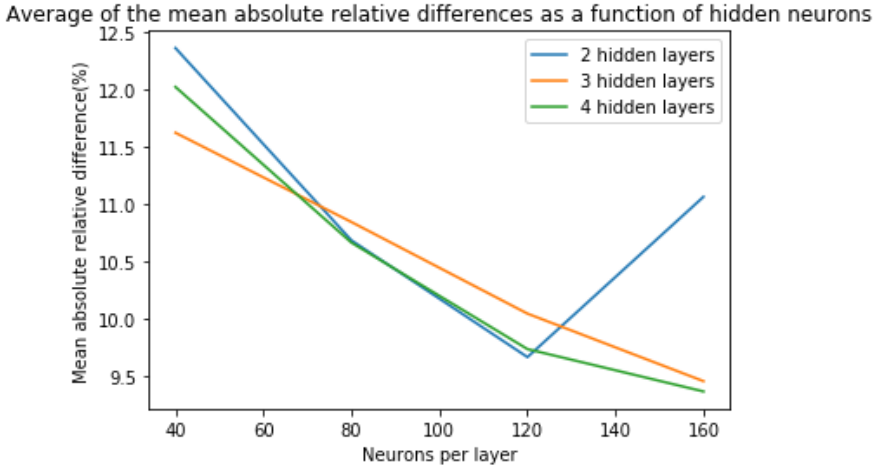

Figure 40: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 1D-case. The networks used the mean absolute error function and trained with all 18 expansion coefficients and 500 samples in the training phase. The test set and predicted set included 200 samples.

In Figure 40 it can be observed that the the prediction of the modes does not improve for deeper networks compared to the results in Figure 18 from experiment 1. Similar behaviour can also be seen in the AVFA which is plotted in Figure 41.

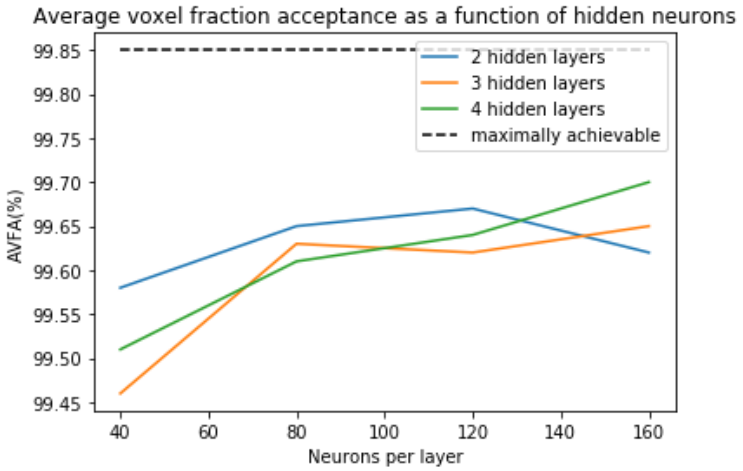

Figure 41: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 1D case. The 200 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 18 modes with 200 test samples is 99.85%. The networks used the mean squared error function and trained with all 18 expansion coefficients and 500 samples in the training phase. The test set and predicted set included 200 samples.

The highest achieved AVFA for the 1D case with 500 training samples is 99.70% which is lower than 99.79% that was achieved in experiment 1. This value was achieved with a neural network with 4 hidden layers and 160 hidden neurons. Moreover, it is notable that increasing the depth of the neural network does not increase the accuracy of the dose prediction, as opposed to the results in Figure 20. For instance in Figure 20 it can be seen that the AVFA values achieved by deeper neural networks are higher, while in Figure 41 the AVFA values achieved by the networks containing 3 and 4 hidden layers are lower than the 2 hidden layer network in general. This result is similar as the result of the 3D-case in experiment 1 in that the dose prediction does not improve as more layers are added to the network. This can again be related to the amount of training samples. Likewise, the mean absolute relative differences of high order mode 24 and lower order mode 22 of the 2D case have been plotted in Figure 42.

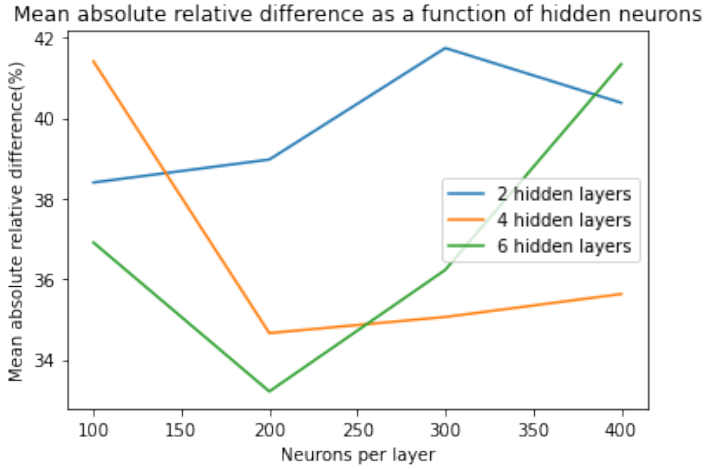

Figure 42: The mean absolute relative differences between expansion coefficients of the test set and predicted set for the high order mode 24 as a function of hidden neurons for the 2D-case. The networks used the mean squared error function and trained with all 24 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

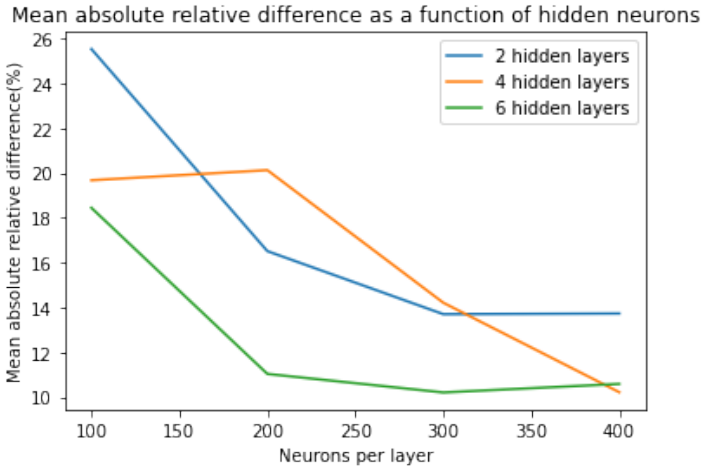

Figure 43: The mean absolute relative differences between expansion coefficients of the test set and predicted set for the low order mode 22 as a function of hidden neurons for the 2D-case. The networks used the mean squared error function and trained with all 24 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

It is clear from Figure 42 that the neural network fails to correctly predict the expansion coefficients of the higher order mode, which was not the case in Figure 23 of experiment 1. Namely, the mean absolute relative differences are twice as large in general as the values in Figure 23. Moreover, it can still be stated that increasing the amount of hidden layers does improve the prediction of both the lower and higher order expansion coefficients. Subsequently, the average over all the mean absolute relative differences have been plotted in Figure 39.

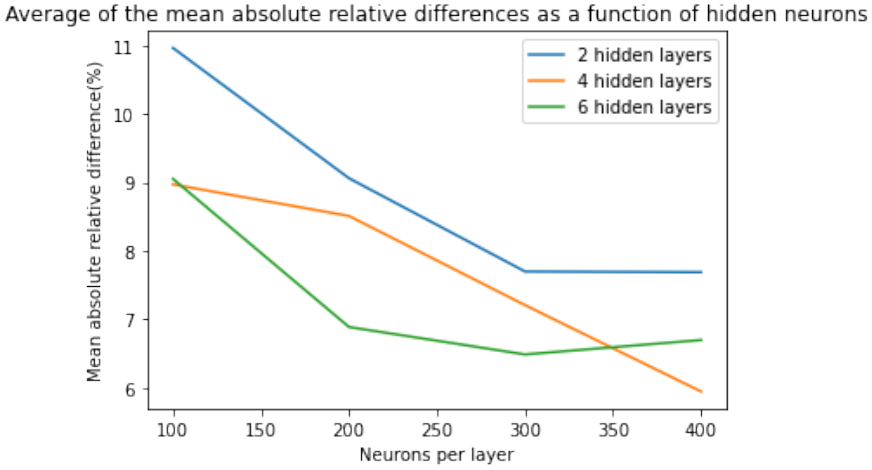

Figure 44: The average over all the mean absolute relative differences of all the modes as a function of hidden neurons for the 2D-case. The networks used the mean squared error function and trained with all 24 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

Similar behaviour as in the prediction of the higher and lower order mode in Figures 42 and 43 can be seen in the average of the mean absolute relative differences over all expansion coefficients in Figure 44. Namely, the predictions of the expansion coefficients of all modes in Figure 44 improve as more layers and neurons are added to the network. Subsequently, the AVFA values have been plotted in Figure 45.

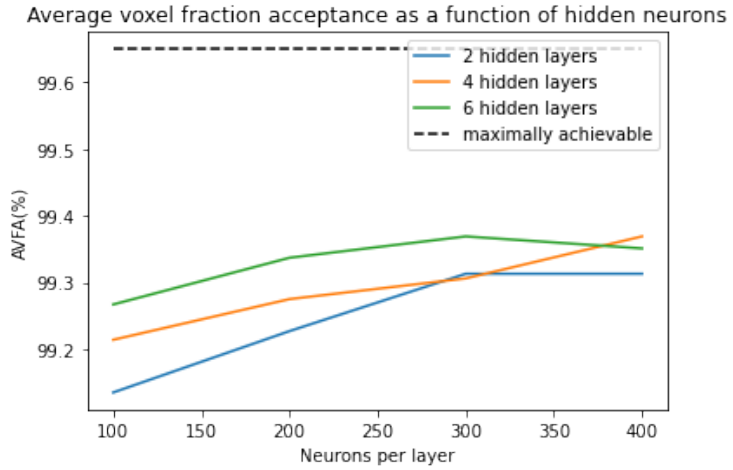

Figure 45: The average voxel fraction acceptance as a function of hidden neurons using a threshold of $T = 1\%$ for the 2D case. The 200 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 24 modes with 200 test samples is 99.65%. The networks used the mean squared error function and trained with all 24 expansion coefficients and 1000 samples in the training phase. The test set and predicted set included 200 samples.

The highest AVFA that was achieved in Figure 45 for the 2D case with 1000 training samples is 99.37% which is again lower than the 99.50% that was achieved in experiment 1. This value was achieved with a neural network containing 4 hidden layers and 400 hidden neurons as opposed to the 6 layers and 400 neurons per layer achieving the highest AVFA in experiment 1. It can be observed that the dose predictions improves as more layers and neurons are added to the network, which is the opposite results compared to the 1D-case. This is again related to the amount of training data.

4.5 Experiment 3: Training the neural networks only with the target mode expansion coefficients for the 3D-case.

In the third experiment, the same neural networks from experiment 1 from the 3D-case have been trained with only the target mode expansion coefficients instead of all 51 expansion coefficients simultaneously. The mean absolute relative differences for the target mode have been given in Figure 46.

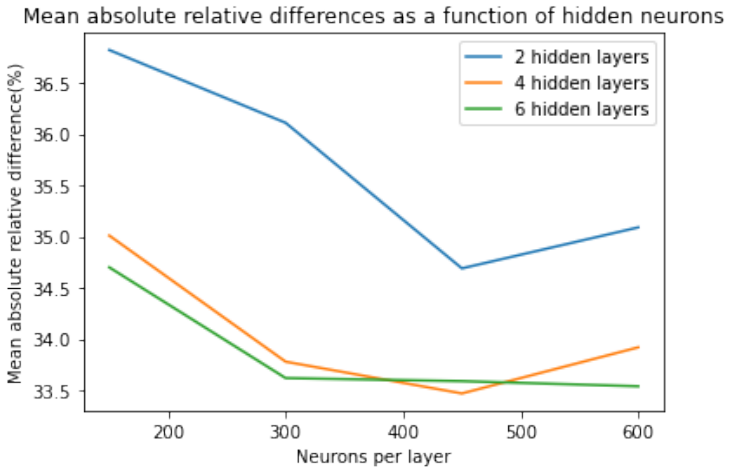

Figure 46: The mean absolute relative differences between expansion coefficients 51 of the test set and predicted set for each neural network setup for the 3D-case. The networks used the mean squared error function and trained solely with the target mode expansion coefficients 51 and 3000 samples in the training phase. The test set and predicted set included 300 samples.

When comparing Figure 46 with Figure 31 from experiment 1, a clear distinction can be seen in the prediction of the target mode. Namely, in Figure 46 it can be observed that the prediction of the target mode does improve as the neural networks get deeper and wider, which is not the case in Figure 31. This results can be related to the the way a network reduces its loss function. Furthermore it can be observed that the prediction is not more accurate compared to experiment 1.

4.6 Experiment 4: Comparing a wide but shallow network with a deep but narrow network.

In the fourth and last experiment, a wide and a deep neural network were tested for different regularization parameter values. The wide and deep neural networks contained 2 and hidden layers and 2000 and 200 hidden neurons respectively. The results for the case without regularization are given in Table 3

Table 3: The results for the wide and deep neural network. The amount of layers, neurons per layer, training time, epochs and the achieved average voxel fraction acceptance values are given for the 3D-case. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

h_l	h_n	$AVFA_{NN,max}$	epochs	training time(s)
2	2000	98.77	33451	1875
20	200	98.71	50127	5806

It can be noted from Table 3 that the achieved AVFA values are not far of from the achieved value in experiment 1, but the achieved AVFA of the wide neural network is Furthermore, it can be observed that the training time of the deep neural network is around 3 times higher than the wide neural network and that the deep neural network achieved a lower AVFA than the wide neural network. Afterwards, the plot of the AVFA as a function of different regularization parameters is given for the wide and deep neural network in Figure 47.

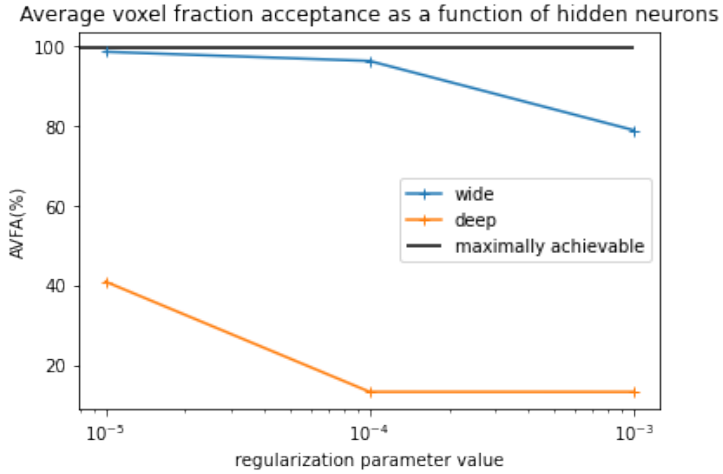

Figure 47: The average voxel fraction acceptance as a function of the regularization parameter using a threshold of $T = 1\%$ for the 3D case. The 300 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 51 modes with 300 test samples is 99.46%. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

It is clear from Figure 47 that as a higher value for the regularization parameter is chosen, the dose prediction deteriorates. Accordingly, it can be stated that using regularization does not improve the prediction of the dose distributions for the wide but shallow and deep but narrow neural networks.

4.7 A short overview of the results from all the experiments

The highest achieved AVFA values and the respective neural networks of experiment 1 and 2 have been provided in Tables 4 and 5. In Figure 48, the mean absolute relative differences of the target modes from experiment 3 have been provided from the networks that trained solely with the target mode expansion coefficients instead of all expansion coefficients simultaneously. In Table 6 and Figure 49, the achieved average voxel fraction acceptances from experiment 4 have been provided of the wide but shallow and deep but narrow neural networks that trained without and with using regularization respectively.

Table 4: The results of experiment 1 showing the high order target modes, sample amounts, achievable and achieved average voxel fraction acceptances and neural network setups. The average voxel fraction acceptances were calculated using a threshold T of 1% difference between the approximated dose distribution and the true dose distribution, relative to the maximum dose of the true dose distribution. The networks were trained with all k_{target} expansion coefficients.

Dimension	target mode	N_{train}	N_{test}	$AVFA_{max}(\%)$	$AVFA_{NN,max}(\%)$	ϵ	h_l	h_n
1D	18	1000	200	99.86	99.79	MAE	5	80
2D	24	2000	200	99.64	99.50	MSE	6	400
3D	51	3000	300	99.46	98.84	MSE	2	800

Table 5: The results of experiment 2 showing the high order target modes, sample amounts, achievable and achieved average voxel fraction acceptances and neural network setups. The average voxel fraction acceptances were calculated using a threshold T of 1% difference between the approximated dose distribution and the true dose distribution, relative to the maximum dose of the true dose distribution. The networks were trained with all k_{target} expansion coefficients.

Dimension	target mode	N_{train}	N_{test}	$AVFA_{max}(\%)$	$AVFA_{NN,max}(\%)$	ϵ	h_l	h_n
1D	18	500	200	99.85	99.70	MAE	4	160
2D	24	1000	200	99.65	99.37	MSE	4	400

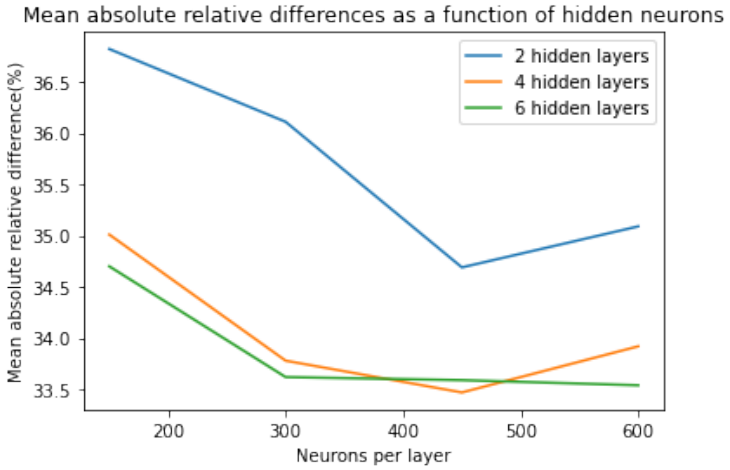

Figure 48: The mean absolute relative differences between expansion coefficients 51 of the test set and predicted set for each neural network setup for the 3D-case. The networks used the mean squared error function and trained solely with the target mode expansion coefficients 51 and 3000 samples in the training phase. The test set and predicted set included 300 samples.

Table 6: The results for the wide and deep neural network. The amount of layers, neurons per layer, training time, epochs and the achieved average voxel fraction acceptance values are given for the 3D-case. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

h_l	h_n	$AVFA_{NN,max}(\%)$	epochs	training time(s)
2	2000	98.77	33451	1875
20	200	98.71	50127	5806

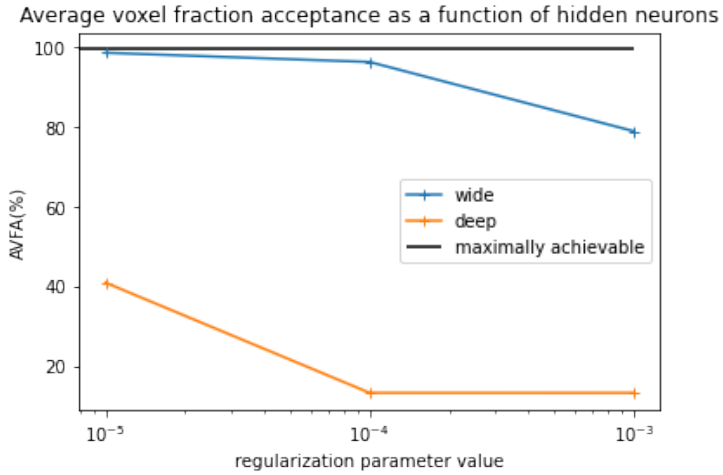

Figure 49: The average voxel fraction acceptance as a function of the regularization parameter using a threshold of $T = 1\%$ for the 3D case. The 300 predicted dose distributions were used to calculate the average voxel fraction acceptance. The maximum achievable average voxel fraction acceptance using 51 modes with 300 test samples is 99.46%. The networks used the mean squared error function and trained with all 51 expansion coefficients and 3000 samples in the training phase. The test set and predicted set included 300 samples.

5. Discussion

5.1 Discussion of the results

In the results of the 2D and 3D case of experiment 1 it was shown that the prediction of the expansion coefficients of the high order mode and dose distributions made by neural networks utilizing the MAE function started to deteriorate faster than networks using MSE as the amount of hidden layers and hidden neurons increased. The reason for this could be that the derivatives that are calculated in the backpropagation step are more complex for MAE than for MSE. Looking at line 5 of the Algorithm 2 in the theory, it can be noted that the derivatives of previous layers are needed to calculate the derivatives of contiguous layers in the backpropagation steps. As the neural networks get deeper, the derivatives get too complicated in the networks using MAE which might not be a good representation of the underlying model in the available data. Consequently, the accuracy of the prediction of the high order mode and dose deteriorated faster as the networks using MAE got deeper than the networks using MSE.

In the results of the 3D-case of experiment 1 and 1D case of experiment 2, it was observed that predictions of the high order modes failed and that more accurate predictions of the dose were achieved by neural networks with 2 hidden layers than networks with more hidden layers. The reason for this could be that the 500 training samples used in the 1D case of experiment 2 and the 3000 samples used in the 3D case of experiment 1 were not enough to represent most of the complex modes. Namely, smaller data sets have a smaller distributed feature representation of the underlying system. This means that the non-linearity in the complex high order modes and even simple lower order modes might not have been fully represented by the given data. The neural networks are constructed to find an underlying distribution or model belonging to a dataset that the network is being trained on. Accordingly, networks with 2 hidden layers were deep and complex enough to sufficiently predict the modes that were fully represented in the data while deeper neural networks over fitted to the data as the complex modes were under represented by the data. This can also be stated about networks that have too many neurons per layer. To summarize, the more training data that is available, the better the representation of the features in the non-linear modes becomes and therefore, the more hidden layers and neurons per hidden layer are required to capture those features. For the 2D-case in experiment 2 on the other hand, the 1000 samples that were used for the training phase were enough to represent lower and higher order modes, meaning that there was the possibility for deeper neural networks to capture more complex features and therefore achieve a better dose prediction.

In experiment 3 in which the networks were only trained with the target mode instead of all the modes up until the target mode, the results from Figure 46 showed that the prediction did improve as more layers were added, which was not the case in experiment 1 in Figure 31. The reason for this could be explained by looking at Algorithm 1. In line 7 it can be seen that the loss is calculated as a sum over all losses at the output neurons. This means that the weights are not updated to minimize the loss function of the target mode, but to minimize the loss over all the output neurons. This is why it is much easier to get more consistent results when the networks are only trained with the target, as there are no other output neurons influencing the loss of the output neuron at which the target mode expansion coefficients are evaluated.

The results from experiment 4 showed that the deep neural network trained much longer than the wide neural network. As was already stated, deep neural networks extract features from a dataset at different levels of abstraction or complexity. More precisely, with each hidden layer in the back propagation steps, the complexity of the calculations increases as more derivatives of previous layers are needed. It could be the case that the weights in those layers are tuned much slower than the weights close to the output layer, as the complex features within the underlying model are much harder to capture than the simpler features. Another observation was that the training time per epoch is twice as high for the deep neural network. The reason for this could be that a low amount of large matrix multiplications are done faster than a large amount of small matrix multiplication

in the propagation steps. Furthermore, the reason why applying regularization did not improve the predictions made by the neural networks could be that the networks did not overtrain within the trained amount of epochs or because the networks were not big enough to overfit to the training data.

To summarize, the highest achieved average voxel fraction acceptances for the 1D, 2D and 3D cases of experiment 1 were 99.79%, 99.50% and 98.84% respectively. 99.79% and 99.50% from the 1D and 2D case are improvements compared to previous research conducted by M. Spadaro of 99.6% and 98%. Moreover, the maximally achieved values are only 0.07% and 0.14% lower than the maximally achievable or target accuracies of 99.86% and 99.64%. This means that there is still some room for improvement but is not significant. Moreover, 98.84% that was achieved for the 3D case is a good start as it has not been tried before, but is 0.62% lower than the achievable 99.46%, which is a much larger difference than the 1D and 2D case. Therefore, there is still much room for improvement in the prediction of the higher order mode and the dose in the 3D case.

5.2 Recommendations

Recommendations for future research regarding the proton therapy treatment planning part would be to set up a more optimized treatment plan than was set up in this research. More precisely, a larger fraction of the total dose should be applied to the PTV instead of the OAR volumes. Consequently, more voxels within the OAR volumes receive a dose that is less than the specified dose of the dose mask, resulting in more voxels being removed from the generated dose distributions corresponding to the patient displacements. The first advantage is that smaller dose distributions corresponding to the patient displacements can be generated faster than larger dose distributions, allowing for more samples to be generated within a certain period of time, which is important as more data is required to represent the higher order modes in the 2D and 3D case to achieve more accurate dose predictions. The second advantage is that smaller dose distributions can be represented by simpler models, which are easier to calculate. Therefore it is recommended that during the treatment planning phase, the amount of voxels in the dose distribution after applying the dose mask is evaluated as well.

Recommendations regarding the deep learning part would be to use more training samples for the 3D case. In this research, 3000 training samples were used, which might be too low to fully represent the high order non-linear modes or even lower order modes that are needed for more accurate dose distributions. Especially for the larger networks that are used for the 3D case, it is recommended to use 10000 samples instead of 3000.

It is also recommended to use k-fold cross validation for the training phase of the neural networks. In summary, the total dataset is split into k amounts of groups of samples, whereafter the networks are trained with k-1 groups and tested with the k^{th} group. Afterwards an evaluation of the model is saved, whereafter the model is trained again using different groups for the train and test set. From the evaluations, the best model can then be picked.

Furthermore, learning rate schedulers could be applied to regulate the learning rate during the training phase. The Adamax optimizer updates any parameter with an individual learning rate within the network, but these individual learning rates are calculated using the initial learning rate that acts as an upper limit. Consequently, using learning rate schedulers could speed up the training process in the first couple of epochs and could help to reduce the loss even further in the last couple of epochs when the loss function is close to its minimum. Consequently, a more accurate model could be captured by the neural network allowing for better dose approximations.

Another recommendation would be to separate the lower order modes and higher order modes over 2 neural networks instead of training one neural network simultaneously with all the modes. This will result in more consistent and accurate approximations of the lower and higher order modes as the losses at output neurons corresponding to the lower order modes do not influence the losses at the output neurons corresponding to the higher order modes.

6. Conclusions

In this research, different feedforward neural networks were tested to determine how the performance in predicting the higher order expansion coefficients of a reduced order model and the dose distributions depend on the neural network architecture and training approaches.

First, a treatment plan for a liver tumour patient was successfully set up which provided the initial dose distribution. Afterwards, multiple dose distribution samples were generated corresponding to patient displacements and a reduced order model could be successfully extracted from the training set of samples using the singular value decomposition. Mode 18, 24 and 51 were chosen as the high order target modes for the 1D, 2D and 3D case respectively and all modes up until the target mode were included in the reduced bases. Subsequently, the target accuracies of the dose approximations using the test set of samples in terms of the average voxel fraction acceptances of the approximated dose distributions were calculated using a threshold of 1% difference between the true voxel dose and approximated dose, relative to the maximum dose of the true dose distribution. The highest achievable average voxel fraction acceptances for the 1D, 2D and 3D cases using the reduced bases are 99.86%, 99.64% and 99.46% respectively. The highest achieved average voxel fraction acceptances in this research for the 1D, 2D and 3D cases are 99.79%, 99.50% and 98.84% respectively. 99.79% and 99.50% achieved in the 1D and 2D cases are sufficiently high compared to the maximally achievable accuracies of 99.86% and 99.64%. The achieved average voxel fraction acceptance of 98.84% for the 3D case is a good start, but is not sufficiently high compared to the maximally achievable accuracy of 99.46%.

It can be concluded that there is still much room for improvement before the neural networks are able to properly approximate the dose distributions of more realistic simulated 3D displacements compared to simulated 1D and 2D displacements. Based on all the results of experiment 1, 2 and 3, it can be concluded that increasing the amount of hidden layers and neurons, up to a certain amount can improve the prediction of both the high order modes and dose distributions, after which the performance starts to deteriorate. The amount of hidden layers and neurons per layer at which this occurs is dependent on the loss function and the amount of training data. More precisely, the performance of neural networks that utilize the MAE function deteriorate faster than networks utilizing the MSE loss function as the amount of hidden neurons and hidden layers increase, which could be caused by the complex derivatives of the MAE loss function in the backpropagation steps. On the other hand, the amount of training data is also important to fully represent the features of the highly non linear modes that deeper and wider neural networks are able to learn. If the data is not able to fully represent those features, there is no use for deeper and wider neural networks. The conclusion that can be drawn from experiment 4 is that deeper networks train longer than wider networks due to the calculations in the backpropagation steps. Furthermore, regularization does not improve the performance of the neural networks within the trained amount of epochs and chosen amount of hidden layers and neurons per layer.

Recommendations for future research would be to evaluate the generated proton therapy treatment plans based on the amount of voxels in the dose distributions after the dose mask is applied. This could reduce the size of the dose distributions allowing for more dose distributions corresponding to patient displacements to be generated within a period of time and resulting in a simpler model to be calculated. 10000 training samples should be used instead of 3000 samples in the 3D case to fully represent the modes. Moreover, k-fold cross validation can be used to capture an even better model. Furthermore, learning rate schedulers could be applied to decrease the training time but also further reduce the loss functions in the last training epochs. Last but not least, splitting low order and high order modes over 2 neural networks could be done to make sure the losses at output neurons corresponding to the lower order modes do not influence the losses at the output neurons corresponding to the higher order modes.

Bibliography

[1] *Proton therapy.* https://en.wikipedia.org/wiki/Proton_therapy/. Retrieved on 1 June 2020.

[2] M. Spadaro. *Reduced order modeling with neural networks for uncertainty quantification in proton therapy.* 14 January 2020.

[3] Hans-Peter Wieser et al. *Development of the open-source dose calculation and optimization toolkit matRad.* Medical Physics 44.6 (2017), pp. 2556 2568.

[4] Uknown. *Investigation into the effectiveness and robustness of proton and photon therapy.* Radiation Science & Technology. Delft university of Technology. March 2019.

[5] G. Cybenko. *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals, and Systems 2.4 (Dec. 1989).

[6] Mengwu Guo, Jan S. Hesthaven, *Data-driven reduced order modeling for time-dependent problems.* Computer Methods in Applied Mechanics and Engineering, Volume 345, 2019, Pages 75-99, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2018.10.029.

[7] Yu, J., Yan, C., Guo, M. (2019). *Non-intrusive reduced-order modeling for fluid problems: A brief review.* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 233(16), 5896–5912. https://doi.org/10.1177/0954410019890721

[8] J.S. Hesthaven, S. Ubbiali, *Non-intrusive reduced order modeling of nonlinear problems using neural networks.* Journal of Computational Physics, Volume 363, 2018, Pages 55-78, ISSN 0021-9991, https://doi.org/10.1016/j.jcp.2018.02.037.

[9] Bre, Facundo Gimenez, Juan Fachinotti, Víctor. (2017). *Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks.* Energy and Buildings. 158. 10.1016/j.enbuild.2017.11.045.

[10] Lui, Hugo Wolf, William. *Construction of Reduced Order Models for Fluid Flows Using Deep Feedforward Neural Networks.* (2019).

[11] Elizabeth Million, *The Hadamard Product* April 12, 2007

[12] Jason Brownlee *Use Weight Regularization to Reduce Overfitting of Deep Learning Models*, Machine Learning Mastery, Available from https://machinelearningmastery.com/machine-learning-with-python/, accessed June 12th, 2020

[13] Jason Brownlee. *How to Normalize and Standardize Time Series Data in Python.* Machine Learning Mastery, Available from https://machinelearningmastery.com/machine-learning-with-python/, accessed June 12th, 2020

[14] Francois Chollet et al. Keras. https://keras.io. 2015.

[15] Martin Abadi et al. TensorFlow: Large-Scale Machine Learning on Hetero- geneous Systems. Software available from tensorflow.org. 2015. url: https://www.tensorflow.org/.