UNIVERSITÄTSMEDIZIN MANNHEIM
MEDICAL FACULTY MANNHEIM OF HEIDELBERG UNIVERSITY
Master of Science in Biomedical Engineering

Automated Radiotherapy Treatment Planning Using Metadata Based Treatment Outcome Prediction

Author:
Franz David Schaefer

Thesis supervisor:
Prof. Dr. Jürgen Hesser

September 15, 2020

Automated Radiotherapy Treatment Planning Using Metadata Based Treatment Outcome Prediction

This Master thesis has been carried out by
Franz David Schaefer
under the supervision of

Prof. Dr. Jürgen Hesser
Data Analysis and Modelling in Medicine
Mannheim Institute for Intelligent Systems in Medicine

at the
Department of Data Analysis and Modelling in Medicine

EXAMINATION BOARD

Prof. Dr. Jürgen Hesser
Data Analysis and Modelling in Medicine &
Mannheim Institute for Intelligent Systems in Medicine

Dr. Martin Polednik
Medical Physics and Central Radiation Protection
Medical Faculty Mannheim

DECLARATION

This thesis is the result of my independent investigation under supervision. Where my work is indebted to the work or ideas of others, for example from the literature or the internet, I have acknowledged this within the thesis.

I declare that this study has not already been accepted for any other degree, nor is it currently being submitted in candidature for any other degree.

I am aware that a false declaration could have legal implications.

Mannheim, 15 September

ABSTRACT

Radiotherapy treatment planning has seen the introduction of deep learning in various ways in the last few years. As the technology becomes more and more accessible, there have been novel implementations in segmentation, dose prediction, texture mapping and even automated treatment planning solutions. This thesis builds upon these previous advancements by implementing a fully connected neural network designed to predict the outcome of radiotherapy treatments by integrating the patient metadata. Additionally, we integrated our network into a radiotherapy treatment optimizer as a regularisation feature, allowing the automated creation of patient specific, personalised treatment plans, based on their individual metadata.

This thesis's primary aim was developing the prediction neural network, this was done by training the network on patient metadata such as age, sex and organ geometry and radiotherapy treatment planning metrics such as monitor units and dose volume histograms. This involved acquiring and extracting opensource DICOM data sets, followed by extraction and standardisation of the required features. We developed a score-based treatment outcome assessment metric to rate the quality of the applied plans. Autoencoding of the larger feature sets was used to avoid overwhelming the prediction network by reducing the number of input features, with high quality reconstruction and acceptable loss values. With the final prediction network having a stable prediction accuracy of 80%, with 80% in the training set, and 78% in the validation set.

The secondary aim of the thesis is was the integration of our prediction network into an existing radiotherapy planning software. This was done by using the prediction network as a feedback system, within the software's optimisation function. By integrating it into the objective and gradient functions our prediction network acted as a regularisation function and applied minor changes to the optimisation process to acquire the best possible prediction while still accounting for the physician prescriptions.

After implementation we observed patient specific manipulations of the treatment plan, when comparing plans generated with and without the prediction model regularisation, with changes in the D-95/50 and V-60/30 values depending on the patient's metadata. Furthermore, we measured significant improvements in the treatment score, especially in cases where the original treatment outcome was subpar, seeing up to 20-30% score improvement in some cases.

TABLE OF CONTENTS

ABBREVIATIONS

DVH	Dose Volume Histogram
ROI	Region of Interest
OAR	Organ at Risk
PTV	Planning Target Volume
CTV	Clinical Target Volume
GTV	Gross Tumour Volume
DL	Deep Learning
NN	Neural Network
FCNN	Fully Connected Neural Network
UI	User Interface
RT	Radiotherapy
CSV	Comma Separated Values
MU	Monitor Units
AoE	Area of Exposure
RELU	Rectified Linear Unit
GELU	Gaussian Error Linear Unit
SGD	Stochastic gradient descent
MLC	Multi-leaf Collimator
GPU	Graphical Processing Unit
DICOM	Digital Imaging and Communication in Medicine
CUDA	Compute Unified Device Architecture
Gy	Gray
CT	Computed Tomography
DAO	Direct Aperture Optimisation
VMAT	Volumetric modulated arc therapy
IMRT	Intensity-modulated radiation therapy
LINAC	Linear Accelerator
CV	Cross validation
GPU	Graphics Processing Unit

1 INTRODUCTION

1.1 Motivation

In the last decade there has be a concerted push towards the personalisation of medicine, promoting the concept of using in depth diagnostics to determine the ideal treatment plan for the individual patient. This data is combined with the patients' medical history and underlying circumstances, enabling the development of targeted treatments and prevention plans (PMC, 2020). Radiotherapy has been one of the fields most heavily impacted by this effort, as standard treatments could cause serious harm if not adjusted for patient and condition circumstances.

Personalisation in radiotherapy takes many forms, examples include: diagnostically locating the tumour for proper treatment coverage, tailoring dose escalation (Barrett, et al., 2018) or even identifying the type of mutation present in the tumour and using proven treatment methods accordingly (Ma, et al., 2017). However when it comes to taking into account the patients circumstances, and individual metadata in the treatment planning, analysis has only been done on broad biological features (Forker, et al., 2015), as the technology to adequately examine individual patient plans has not been easily accessible.

Machine learning has become an increasingly valuable tool in understanding pattern relationships, with applications in fields such as machine vision, voice recognition and prediction modelling. With open source solutions like Pytorch and TensorFlow being released in 2016 & 2015 respectively, machine learning technology has become more accessible. Enabling deeper modelling of complex features and identifying subtle but consequential patterns.

The primary aim of this thesis is to use machine learning to develop a deep learning prediction model, which uses patient metadata and radiotherapy planning data to predict the treatment outcome. Thereby allowing the physician to model their plans and adjust them according to the model's feedback, leading to the personalisation of plans according to patient metadata.

We intend to train our model using open source cancer treatment data from online sources, which include the patient clinical metadata. Then after extraction and standardisation, create a robust treatment outcome scoring mechanism to better rank and rate how well the treatment plans were personalised to the patient. Finally, we intend to use the Pytorch ecosystem to design and train our neural network model on our processed data, to identify what makes each plan ideal or unideal for the patients' circumstances and condition.

Our secondary aim was to integrate the prediction model into an existing radiotherapy treatment planning software. If successful this would allow the user to input a patients CT and segmentation information, apply appropriate prescripts and set the metadata. Then the software would optimise the plan with the assistance of our neural network model, automatically personalising the plan to the patient. As this is a secondary aim, and would be a proof of concept, we have no specific requirements regarding performance, and will have to be judged via radiotherapy metrics and dose volume histogram comparison. We only expect minor changes to the treatment plans, as the network would not be designed for complete plan creation, instead as an adjustment method to better fit the plan to what the patient's circumstances require.

1.2 Treatment Planning

1.2.1 Radiotherapy

Radiotherapy, or Radiation therapy, is the treatment of disease with the use of ionising radiation. The concept is to use the radiation to damage and destroy tumorous cells, while sparing the healthy tissue, thereby allowing the body to ideally overcome the disease. There are two methods of applying the radiation, externally (for example VMAT) or internally (for example Brachytherapy), this thesis will be focusing on the former. The radiation is usually applied in daily fractions to avoid overexposing the skin and surrounding tissues and giving the body a chance to repair damage, while not giving enough time for the highly exposed tumour cells to repair as well. (Mayles, et al., 2007).

Dose & Monitor Units

Absorbed dose is measured in Gray (Gy) and depends on a variety of factors, however this only describes the energy imparted on a material or tissue (see Equation (1.1). The pre-treatment calculations are usually done via either a collapsed cone or pencil beam or in the cases where long treatment planning times are acceptable via a probability based statistical method known as Monte Carlo (MC), Biological dose must also be considered for radiotherapy as different tissues react in different ways.

$$D(Gy) = 1.602 \cdot 10^{-9} \cdot N\left(\frac{1}{cm^2}\right) \cdot LET\left(\frac{keV}{\mu m}\right) \cdot \frac{1}{\rho}\left(\frac{cm^3}{g}\right) \quad (1.1)$$

Absorbed Dose Equation, particle number (N), linear energy transfer equation (LET) & density of the material (□)

Monitor units (MU) is the method in which the machine dose is measured, allowing the physicians to track the amount of dose imparted into the patient/measurement device.

$$\mathrm{MU} = \frac{\mathrm{cGy.\,SSD}}{\mathrm{CF} \times (\mathrm{OF} \times \mathrm{PDD} \times \mathrm{WF})} \quad (1.2)$$

Monitor Unit Equation: Beam Energy (cGy), Source Surface Distance (SSD), Tissue-Maximum Ratio (TPR), Percentage Depth Dose (PDD), Output Factor (OF), Wedge Factor (WF), Calibration Factor (CF)

Fractionation

Fractionation is the process of dividing a radiotherapy plan into separate application sessions. This spares normal tissues by allowing repair of sublethal damage between dose fractions and allows Repopulation of cells. Simultaneously dividing a dose into several fractions increases damage to the tumour by reoxygenating the tumour environment and reassorting the cells into radiosensitive phases of the cell cycle.

$$\text{Biologically effective dose (BED)} = \mathrm{n} \times \mathrm{d}\left(1 + \frac{\mathrm{d}}{\alpha/\beta}\right) \quad (1.3)$$

Number of fractions (n), dose per fraction (d), linear component (α), quadratic component (β).

Regions of Interest

Different parts of the body are segmented into regions of interest (ROI) during treatment planning according to how they need to be irradiated, or in some cases avoided (Burnet, et al. 2004). The following are the major ROI that need to be considered in radiotherapy.

- Gross Tumour Volume (GTV): Extent and location of growth (tumour plus metastasis).
- Clinical Target Volume (CTV): Tissue volume that contains a GTV and disease that is not yet detectable.
- Planning Target Volume (PTV): A ROI concept that considers the effect of all the possible variations and inaccuracies. For example, organ/patient movements, treatment technique and setup errors.
- Organs at Risk (OAR): Normal tissues which are especially sensitive to radiation. And need to be actively avoided during treatment planning if possible.

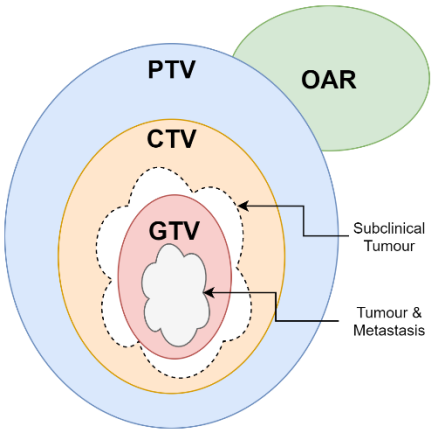

Figure 1.1 – Radiotherapy ROI Visualisation

1.2.2 IMRT & VMAT

Intensity modulated Radiotherapy (IMRT), developed in 2000, is the widely used standard method for radiotherapy planning in the world. This is due to the ability of IMRT to spare OARs by using non-uniform radiation beam intensities and inverse planning methods (Liu, et al. 2004). Specifically, fixed field IMRT, delivered via linear accelerators (LINAC's) fitted with multi-leaf collimators (MLC), has become the most popular modality of IMRT and is considered as the standard technique. However, IMRT is time consuming and delivers more monitor units in comparison to newer methods and has therefore been mostly superseded by Volumetric modulated arc therapy (VMAT) as the preferred treatment planning system.

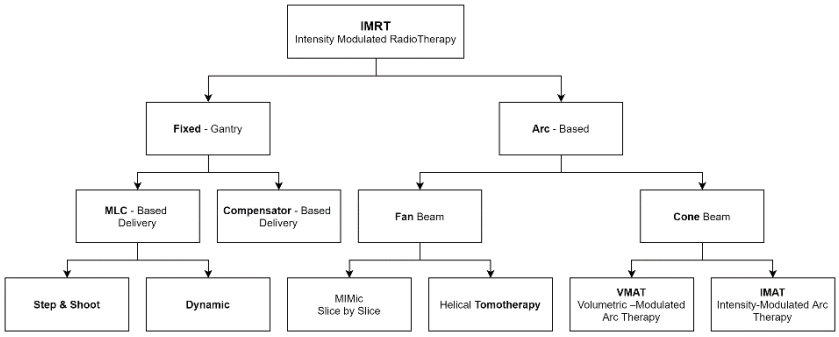

Figure 1.2 – IMRT Hierarchy Diagram

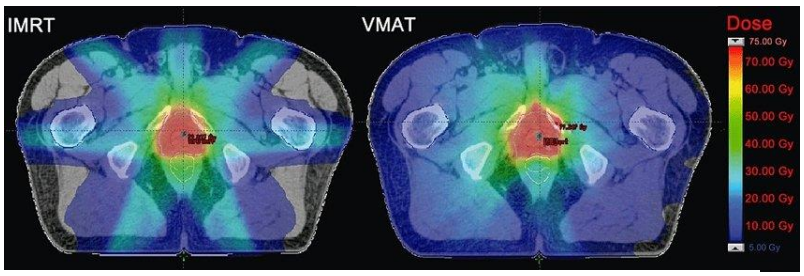

Figure 1.3 – IMRT vs VMAT Comparison for post-prostatectomy radiotherapy. (Nguyen, 2012)

VMAT was developed in 2010 as an offshoot from IMRT, is the currently preferred radiotherapy technique for dose delivery. It involves the continuous application of radiation to the patient as the LINAC rotates. Simultaneously it shapes the beam using MLCs, targeting the tumour while sparing the healthy tissue of radiation dosage. The underlying concept is to give the physician more freedom when it comes to planning and administering treatments. VMAT involves simultaneously adapting the Gantry position and speed, the collimator leaf positions and angle, Dose rate and if possible, the non-coplanar beam directions.

1.2.3 Workflow

Unlike the traditional approach to planning, IMRT & VMAT are inverse planning methods. This means that the planning begins with the desired result and calculates how the beam must be programmed to acquire it (Cilla, et al., 2020).

First the ROI's are contoured (The images are usually acquired via CT for accurate dose calculations as it contains information regarding tissue density, and PET for functional localisation of the tumour) Then the treatment fields are specified, including what beam angles are available to the planning software. Finally, one must select the parameters used to run the optimization algorithm, and check if the optimisation successfully acquired an acceptable solution, for both the physician and the initial constraints set.

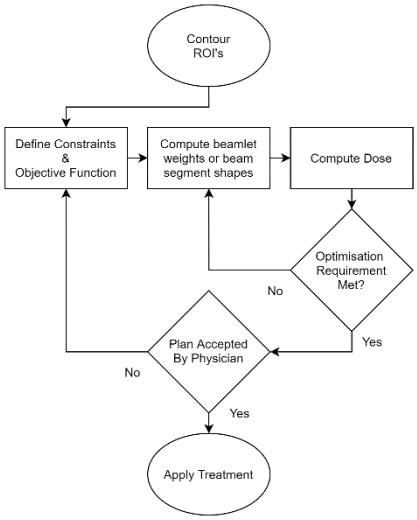

Figure 1.4 – IMRT Treatment Planning Workflow

1.2.4 Optimisation

The optimisation process can be described as: The minimisation of an objective function f(x), subject to inequality and equality constrains $g_i(x) \leq 0$ and $h_i(x) = 0$ respectively. This can be solved by using the conjugate gradient decent method, to take iterative steps towards the ideal minima. By first computing the dose influence in advance the optimiser will be able to adjust weights. Therefore, by taking iterative steps along the gradient of the underlying optimisation function, the optimiser will eventually reach a local minima/optimum.

$$L(\Phi,\lambda) = F(\Phi) + \sum_{k=1}^{N} \lambda_k G_k(\Phi) \qquad F(\Phi) = \frac{1}{n}\sum_{i=n}^{n} e^{-\alpha\Phi_i} \tag{1.4}$$

***Left:** Dose Optimization problem, **Right:** Target Function*
Number of constraints (N), constraint function (G (Φ)), Lagrange multiplier (λ).
Target Number (n) clinical outcome correlator (α).

The objective function when minimised should achieve 4 specific goals. These are as following:

- Goal I: Achieve a sufficient dose in target tissue
- Goal II: Do not exceed acceptable doses in OAR's
- Goal III: Target dose should be conformal and spare healthy tissue (non-Target or OAR)
- Goal IV: No large or excessive hot spots in the target/body

Objective Functions

The objective functions can be used to control the dose in the target volume, however, to avoid being only an increasing force and to avoid hot spots they often come in pairs. The most common example of an objective function in radiotherapy is the Overdose/Underdose Function, the two functions work together to create a parabolic minimum for the objective function to identify and optimise.

$$F_{over} = \frac{1}{N}\sum_{i=1}^{N}(D_i - D_0)^2\,\theta(D_i - D_0)$$
$$F_{under} = \frac{1}{N}\sum_{i=1}^{N}(D_i - D_0)^2\,\theta(D_0 - D_i) \qquad (1.5)$$

Over/Underdose objective function equation pair.
Number of Voxel in ROI (N), the current dose (D_0), prescribed dose (D_i) Biological factor (θ)

Conformality Function

This function is used to shape the dose gradient closest to the target volume, ideally sparing generic tissue around the target, while getting as homogeneous dose within the structure. Aiming towards a square function, where dose instantly increases within the target area and is 0 outside.

$$G(D) = \frac{1}{n}\sum_{i=1}^{n} f\left(\frac{D_i}{D_{i,0}}\right) \qquad (1.6)$$

Conformality Constraint Function
Voxels outside of ROI (n), distance to target volume (f(x)), the current dose (D_0), prescribed dose (D_i)

Cost Functions

Cost functions, or biological constraint functions are designed to spare the OAR's. There are two types of cost functions that are generally implemented. The parallel constraint which acts to smooth the dose gradient, and the serial constraint which avoids hot spots and max dose within the ROI. The cost functions are similar to objective functions however to encourage avoidance a power law exponent is added (k) to cause an exponential increase in loss if the dose limits are violated.

$$G_{Parallel}(D) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{1+(D_0/D_i)^k}$$
$$G_{Serial}(D) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{D_i}{D_{ref}}\right)^k \qquad (1.7)$$

Cost/Biological Constraint Functions
Power law exponent (k), Number of Voxel in ROI (n), the current dose (D_0), prescribed dose (D_i)

1.3 Machine Learning

1.3.1 Neurons

Deep learning employs the use of artificial neurons (Figure 1.5) as its basic computational unit, which parallel the biological neurons in their function (Bonaccorso, 2017). The artificial neuron integrates weighted inputs and activates the summed input through an activation function. This function converts the inputted values through applied weights, into a single output which can then be passed on to the next layer of neurons. The activation function is therefore the key to how the neuron and network behaves and can be generally differentiated between linear and non-linear.

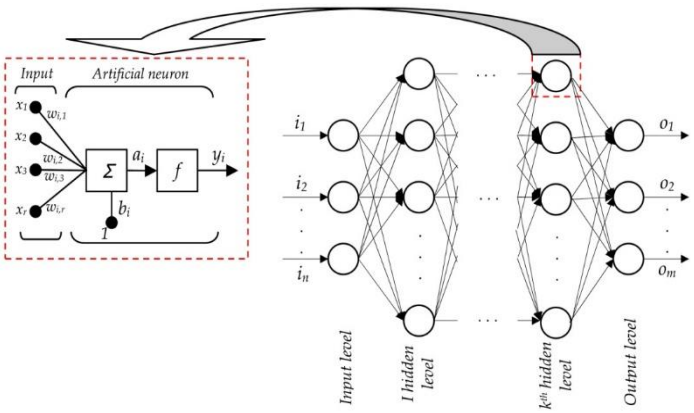

Figure 1.5 – An expansion of an artificial neuron within a network layer. The neuron integrates inputs from the neurons x1; x2 etc. The result is returned after activating with function 'f' (Tanikić, 2012)

Some common activation functions include the following:

Rectified linear unit (ReLU)

The ReLU is non-linear activation function, its output is the maximum value between zero and the input value. It is the simplest and one of the cheapest functions available and therefore is widely used. There is also a leaky- ReLU version which allows for a small nonzero negative gradient.

$$f(x) = \begin{Bmatrix} 0 & x < 0 \\ x & x \geq 0 \end{Bmatrix} \xrightarrow{\text{Derivative}} f'(x) = \begin{Bmatrix} 0 & x < 0 \\ 1 & x \geq 0 \end{Bmatrix} \quad (1.8)$$

ReLU + Derivation

Sigmoid function (SIGM)

The sigmoid function is non-linear, with all inputs being converted to an output ranging between 0 and 1. Therefore it works very well as a scaling function, when you wish to have a specific range out outputs returned.

$$f(x) = \frac{1}{1 + e^{-x}} \xrightarrow{\text{Derivative}} f'(x) = f(x)\big(1 - f(x)\big) \quad (1.9)$$

Sigmoid + Derivation

Gaussian Error Linear Units (GELU)

The GELU nonlinearity weights all inputs by their value, instead of gating inputs as in the case of ReLU. This function results in better performance and so it good for long term training. (Hendrycks & Gimpel, 2016)a

$$f(x) = x \cdot \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \approx 0.5x\left(1 + \tanh\left(\sqrt{2/\pi}\,(x + 0.044x^3)\right)\right) \quad (1.10)$$

$$f'(x) \approx 0.5\tanh(0.035x^3 + 0.79x) + (0.053x^3 + 0.39x)\operatorname{sech}^2(0.036x^3 + 0.79x) + 0.5$$

GELU function Sigmoid + Derivation

Exponential Linear Unit (ELU)

The ELU function behaves similarly to Relu in the positive space, however in the negative, instead of being a sharp smoothing function, it behaves much smoother (in an exponential fashion). However, it can also blow up when the inputs become larger.

$$f(x) = \begin{cases} x & x > 0 \\ \alpha(e^z - 1) & x \leq 0 \end{cases} \xrightarrow{\text{Derivative}} f'(x) = \begin{cases} 1 & x > 0 \\ \alpha e^z & x \leq 0 \end{cases} \quad (1.11)$$

ELU + Derivation

1.3.2 Neural Networks

A collection of neurons is a neural network. The neurons are connected in a way that they form layers. Where every neuron in one layer is connected to all neurons in the subsequent layer. A neural network takes an input, passes it through multiple layers of hidden neurons via synapses. These are connections that pass the outputs forwards to the next layer of neurons, with a unique weight applied to each (Shanmugamani, 2017).

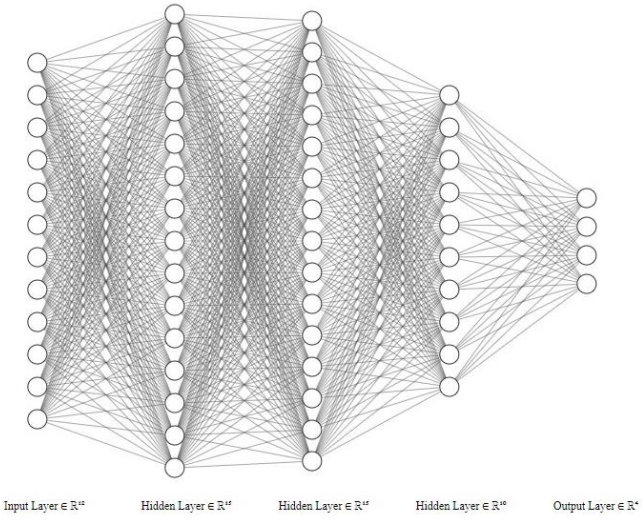

Figure 1.6 - Example of Neural Network Structure

NNs can be of two forms: feedforward or recurrent. In feedforward NN, connections between the neurons are strictly in the forward direction, therefore avoiding recurrent cycles. A NN is divided into 3 significant layers: Input Layer (where the inputs are fed and stored), Hidden Layers (where the relationship between data/ and output is computed and learned) and the output Layer (where the final output is extracted from the previous layers).

$$y_j = f\left(\sum_i w_{ij}x_i + b_j\right) = f(y_j^0) \quad (1.12)$$

Example Neuron Activation Function
Activation function (f ()), weight (w), input (x), bias (b)

As an example, for a hidden layer, the neuron (n_j) receives an input (x_i) from each neuron in the previous layer. The neuron then takes the input values and weights them by the strength of the synapse (w_{ij}), adding a bias (b_j). After which it activates the value using the activation function (f) to produce an output (y_j) for the next layer.

1.3.3 Loss and Backpropagation

Backpropagation is the concept of working through a network in reverse compensating for incorrect steps taken. This requires a loss function, which calculates the divergence or Loss of the network output to the correct answer. Through repeated backpropagation the goal is to minimise this loss function, which in cases of nonspecific goals is defined as the objective function.

The loss functions relevant for this thesis include the following:

L1 Loss
A loss function that measures the mean absolute error between each element in the input and target. This loss function is a good baseline and first start to any problem as it is simple to use and understand. It is based on the classical objective function used in classical inverse problem mathematics.

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i}^{n}|y_i - \hat{y}_i| \qquad (1.13)$$

L1 Loss Function
Each feature (n), True Value (ŷ), Predicted Value (y)

MSE Loss
A loss function that measures the mean squared error between each element in the input and target. This function is one of the more popular to use, as its robust and widely applicable, especially in autoencoder situations. It generally acts more strongly against outliers, making it ideal for achieving a clean prediction of constant quality.

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i}^{n}|y_i - \hat{y}_i|^2 \qquad (1.14)$$

MSE Loss Function
Each feature (n), True Value (ŷ), Predicted Value (y)

Cross Entropy Loss
This loss metric measures the quality of a classification model. It returns a probability value between 0 and 1 for each class thereby describing the networks best guess. Cross-entropy loss increases as the predicted probability diverges from the actual label and decreases as the network's predictions become more accurate. Classification is when the model decides whether the input is best described by a certain class/type of output, and the cross-entropy loss describes how well the network chose.

$$L(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \qquad (1.15)$$

Cross Entropy Loss Function
Each class (n), True Value (ŷ), Predicted Value (y)

1.3.4 Optimisers

Stochastic gradient descent (SGD)

This optimizer finds the mean loss over all n samples in the dataset, for each updating step. This, however, becomes very inefficient if the number of samples n increases. A solution to this problem is to batch the data set and only input a random portion of samples in each updating step. The loss varies stochastically depending on the used batch, thereby the name: stochastic gradient descent (Monro, et al., 1951 & Amari, 1993).

Adaptive Moment Estimation (Adam)

The Adam optimizer, is an extension on the SGD optimizer by considering gradients from previous training steps, thereby speeding up the convergence (Adam, et al., 2014). This introduces a momentum to the optimisation, (the influence of which can be modified as a parameter), where previous gradients influence the current update step.

The optimizer contains two velocities: The scaled past velocity and gradient & the second-order square velocity and the scaled gradient squared.

$$v_d W = \beta_1 v_d W + (1 - \beta 1)\frac{\partial J}{\partial W}$$

$$s_d W = \beta_2 s_d W + (1 - \beta_2)\left(\frac{\partial J}{\partial W}\right)^2$$

$$\hat{v}_{dW} = \frac{v_d W}{1 - (\beta_1)^t}$$

$$\hat{s}_{dW} = \frac{s_d W}{1 - (\beta 1)^t}$$

$$W = W - \alpha \frac{\hat{v}_{dW}}{\sqrt{\hat{s}_{dW}} + \varepsilon}$$

- $v_d W$ - the exponentially weighted average of past gradients
- $s_d W$ - the exponentially weighted average of past squares of gradients
- β_1- hyperparameter to be tuned
- β_2- hyperparameter to be tuned
- $\frac{\partial J}{\partial W}$- cost gradient with respect to current layer
- W - the weight matrix (parameter to be updated)
- α - the learning rates
- ε - small value to avoid dividing by zero

Equation 1.16 – Adam Optimiser Gradient Equations

1.3.5 Regularization

A common problem in deep learning, is that the network is too keenly trained for the training data, and as a result cannot fit to other data and fails to predict a correct answer. This is called overfitting, and regularisation is a technique to mitigate this effect.

Data Augmentation and Acquisition

The most effective way to avoid overfitting is to simply acquire more training data, and if this is impossible, to create more via data augmentation. However, depending on what the network is training this can be difficult, in the case of medical imaging one often has issues with confidentiality agreements, and the creation of data often requires significant effort or can even require multiple years of time. Augmentation generates new training data from given original dataset, this can be done via multiple methods, be that introducing noise, cutting out data points, etc. Some common publicly available options include adding jitter, using a principal component analysis (PCA) and rotation or flipping the dataset.

Dropout

Dropout randomly sets a fraction of the artificial neurons to zero (See Figure 1.7) in each forward pass during training (Srivastava, et al., 2014). The idea behind this technique is to force the network to retain redundant representations and to prevent the artificial neurons from co-adapting. It can then be deactivated during network testing and evaluation, as outside of training it can cause suboptimal performance.

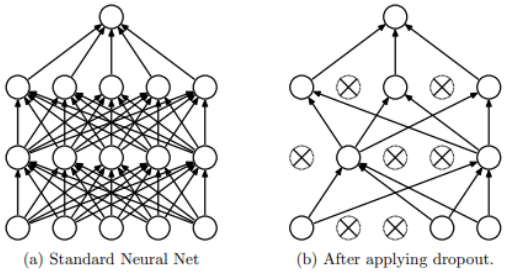

Figure 1.7 - Dropout Neural Network Model (Srivastava, et al., 2014)

Cross Validation

When training a neural network, the dataset must be split into a minimum of 2 sets. The Training and the testing, this is done as we cannot test the networks performance on the training set without having the network simply memorise the test and train dataset. Therefore, all network evaluation must be done via the test set. It is also recommended to use stratification when splitting, to ensure that an even representation of metrics is trained and tested for, and that the network is performing well for all cases.

Cross validation (CV) is the idea of partitioning the available data into three sets. The train and test data, and then splitting the training data again, creating the validation set. CV is ideal for assessing how a model generalizes to the testing set.

The most common technique for cross validation is k-fold cross validation, where the training set is split into k smaller sets. The model is then trained a specified ratio of k-splits and validated on the remaining k-splits. Common settings for k-fold, are 5 and 10 folds where the training set include 80 and 90% of data respectively.

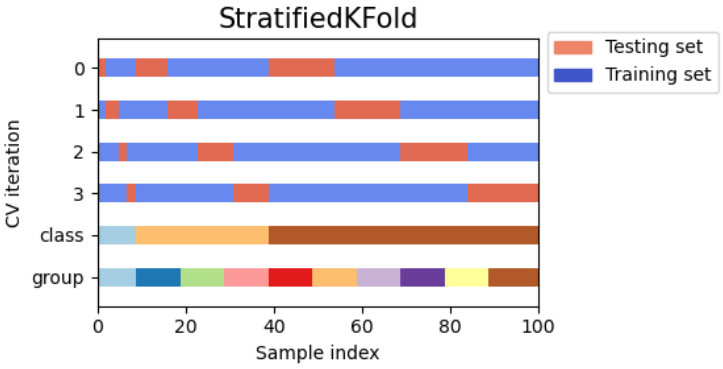

Figure 1.8 - Stratified K Fold Example (scikit-learn, 2020)

1.3.6 Autoencoders

An autoencoder is a special type of neural network, while most attempt to predict or interpret input data, the autoencoder compresses/encodes the input data and then attempts to reconstruct/decode it again. This can be especially useful if one needs to compress data, as the encoding and decoding step can be split, allowing the user to utilise the encoded data version, without needing to worry about losing essential information.

This is because, while the autoencoder will lose some information, since it needs to reconstruct it again, and the loss of the network is measured by the output's similarity to the input. IT can only do this if the key data is present within the encoded data set. Therefore, if the autoencoder network has a low enough loss, one can use the compressed data as an analogue to the original.

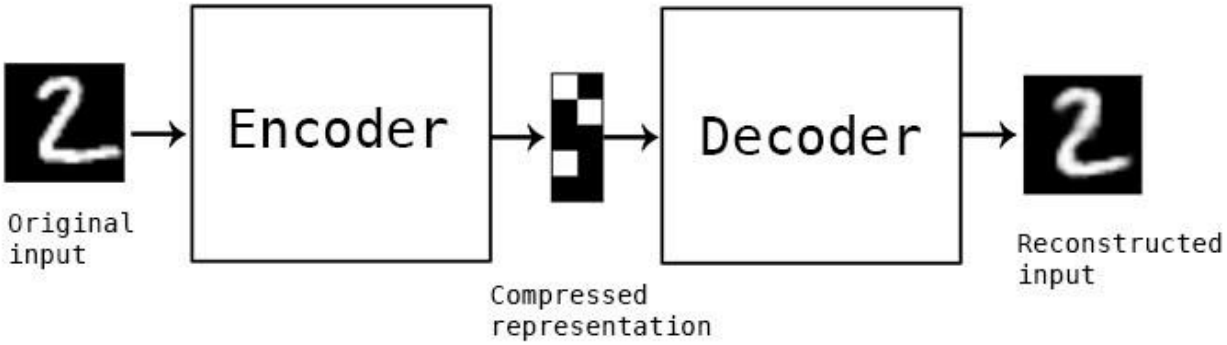

Figure 1.9 – Example of autoencoder network structure and data outputs. (Chollet, 2016)

A secondary benefit to an autoencoder data set is that since the network has been trained to accept a specific type of data, it will tend to automatically remove outliers in its compression, therefore an autoencoder network can be used for data denoising, especially when combined with up sampling techniques.

1.4 Digital Imaging and Communications in Medicine (DICOM)

1.4.1 Origins & Use Cases

DICOM was a file type/method developed in 1993 to manage/transfer and store medical imaging data. It was designed in a way that it would be back compatible to past versions, allowing it to have a unified version system (All versions would be able to work with one another). It has become the most popular and common way of processing medical image data, seeing its use in servers, networks, and image capture devices worldwide (Bidgood, et al., 1997).

DICOM has seen extensive use within the radiological and radiotherapy departments, and part of its innovation was spurred by the requirements of radiologists to keep patient data organised and linked. Thereby DICOM includes standards for multiple image modalities such as: Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and Radiotherapy (RT), as well as many others. It has been especially successful in enabling the transfer of metadata related to images, as all DICOM files include a header to keep the information and images unified.

1.4.2 Data Format

As mentioned previously the main thing that sets DICOM apart is the fact that it groups information into the image/data set. For example, A CT image will contain itself as well as the patient's metadata (name, sex, date of birth, etc), within the DICOM file, thereby avoiding separation or mislabelling. However, a DICOM file consists of several other attributes such as image acquisition information or proposed treatments, but a single DICOM file can have only one attribute containing pixel data. In the case of an MRI, three- or four-dimensional data can be stored in the pixel data attribute but must be specified as such (Mustra, et al., 2008).

DICOM uses three different encoding schemes. Using explicit value representation (VR) data elements, and specified compression to ensure the image quality is not compromised. The specified formatting (for structure see Figure 1.10) for each data element is:

- GROUP (2 bytes)
- ELEMENT (2 bytes)
- VR (2 bytes)
- Length in Byte (2 bytes)
- Data (variable length).

Generally, the same basic format is used for all types of applications to ensure a universal system (There are of course exceptions however). When written to a file, usually a header (Which contains the key information, and the methods used) is added before the data set.

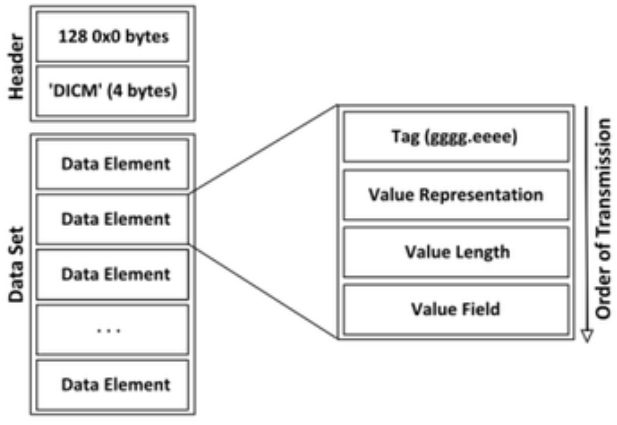

Figure 1.10 - DICOM File Data Structure (Mustra, et al., 2008)

1.4.3 DICOM File Types Used

Image Data

- CT
  Computer Tomography image data is used primarily for classifying tissue densities for the calculation of dose distributions, as the amount of dose absorbed is directly related to the materials density, and without proper density mapping it would be impossible to simulate dose deposition.
- PET
  Positron Emission Tomography image data is used primarily for identifying the functional location and extent of the tumour. This is done via injection of a decaying substance, which will release positrons which can be used to pinpoint the location of decay. If this substance is drawn to tumour cells, then the decay locations would indicate tumour locations.

Radiotherapy Data

- RT – DOSE
  Holds the dose information for each ROI (Region of Interest) either as a massive dose map (3D image of dose distribution) or already converted into a DVH (dose volume histogram).
- RT – STRUCT
  Holds the ROI definitions and various texture and biological scalar factors which are needed for planning.
- RT – Plan
  Defines the radiotherapy plan via control points, including beam angle, collimator settings, and of course how much fluence is required for each of these control points.

DICOM does have a few underlying issues, especially in our application. These issues revolve around human interpretation of different clinical protocols, defining the data entered and human errors, which often either results in data being missing, mislabelled or misplaced. These effects can seriously hamper our ability to develop an automized data extraction method and would result in many data pathing checks before any proper extraction could begin.

2 STATE OF THE ART

2.1 Existing Applications

2.1.1 Automated Treatment Planning

The primary issue with treatment planning is the consumption of large amounts of human effect, occasionally approaching days of continues work. This has a twofold detriment to the process, firstly the delay in treatment could result in complications or substandard treatment, secondly it means physicians need to spend more time on single patients, when it could be possible for them to treat more.

The removal of the human element in the process of radiotherapy treatment planning, as well as the speeding up of the process is not a new idea and has been attempted in various forms over the last decade. However, with the introduction of new technologies and novel techniques the research is approaching forms that are clinically viable. Some of which are described by Moore, 2019 in his review paper of automated treatment planning, as well as recommended methods to evaluate solutions and the potential drawbacks and implications of clinical implementation.

An interesting description is made between Library Based and Model (Knowledge) based dose estimations. The former attempts to match former patients to the new, thereby allowing it to reuse the underlying geometrical structures and speeding up dose estimation. The latter which is a model trained on a dataset of previous estimations, instead attempts to learn how dose interacts with underlying geometrical structures. Thereby allowing for much greater flexibility in handling patients, however it requires large amounts of datasets before it is able to be used (In the following section 2.1.2 we describe such a model)

Two approaches are suggested for developing automated solutions: Patient specific dose estimations or Dynamic Inferred parameters. Patient specific planning is closer to the classical approach, by creating an ideal DVH that would be just out of reach of the optimiser and have the optimiser attempt to approach it via DVH backpropagation with the underlying priorities and ideals would be handled by the automated system. The second dynamic option attempts to make the final set of objectives and weights are ultimately patient specific. Allowing the method to choose the priorities of the different ROI, as well as the ideal DVH curves. Moore warns that this would result in different outcomes after repeated evaluations, as the model would be partially RNG based.

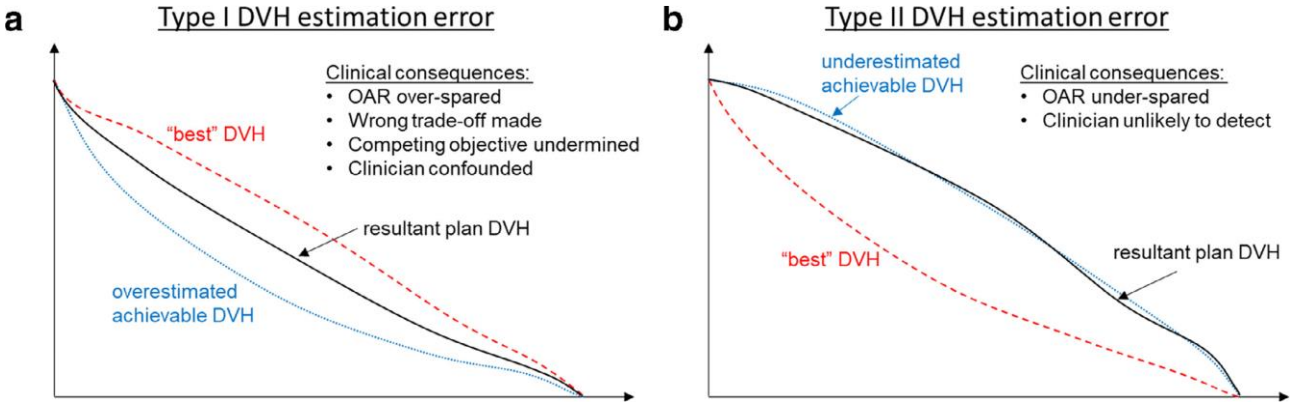

Figure 2.1 - DVH estimation errors and their clinical consequences.
Left: *Type 1 DVH Estimation Error,* ***Right:*** *Type 1 DVH Estimation Error (Moore, 2019)*

He discusses how new solutions are to be assessed and tested. He recommends directly comparing plans DVH for key ROI for all attempted patients. Either by direct plot comparison or using statically values such as V-60 or D-95 (Volume with 30 Gy, Dose covering 95% of volume), depending on the ROI type a different dose region would be more critical. Additionally, it is recommended, to apply checks for statistical significance, to ensure that significant changes are being made between the new solution and existing methods. Finally, when possible it is ideal to get a review of plans from a practicing professional thereby getting confirmation if the plan is considered clinically improved/viable (Depending on the goal of the tested solution)

Finally, he discusses potential pitfalls during and after implementation. The three primary problems identified, were the enshrined trade-off, estimation error and correction problem. The enshrined trade-off error is a lack of flexibility in a trained planning solution, resulting in an edge case prediction due to lack of training in a certain feature. The estimation error (Figure 2.1) occurs when the best DVH ideal is either unachievable, thereby causing the OAR to be over-spared and Target underdosed (a case of wrong trade-off) or the best DVH being undetected, therefore it being under spared. Of these the second is much harder to detect, as the target DVH would not be compromised, and the OAR would be sacrificed. Lastly the correction problem stipulates that if there is an error of the previous two, it would require for manual correction which means that the automatic solution has failed, or only an achieved partial planning solution.

2.1.2 Deep Learning in Treatment Planning

Deep learning has made significant inroads in treatment planning, seeing implementation in a wide variety of applications, varying both in scope and purpose. In this section we will discuss three applications, beginning with a general automatic planning solution, followed by a method used for accelerating the optimisation progress and finally investigating more in-depth methods for radiomic metric identification used in outcome predication and risk assessment purposes.

A knowledge-based planning (KBP) is a method used to augment and potentially replace manual treatment planning. Such a system was developed by (Ziemer, et al., 2017), with Dr Moore (from the previous section) being involved in its development. In the KBP study the researches demonstrate the quality of the KBP and investigate the factors that are important for physicians to better refine the model. This was done via blind plan assessment by two practicing physicians as well as using the quality assessment methods described in section 2.1.1.

The model itself is a modified 3d -CNN used to make dose predictions as derived from a 39-patient training set. This dose prediction is then used to create DVHs, which were checked for optimal performance before being used to generate patient-specific, inverse optimization objectives, for VMAT treatments. It considers different treatment planning solutions but has trained on comparatively few data set, when compared to modern machine learning projects.

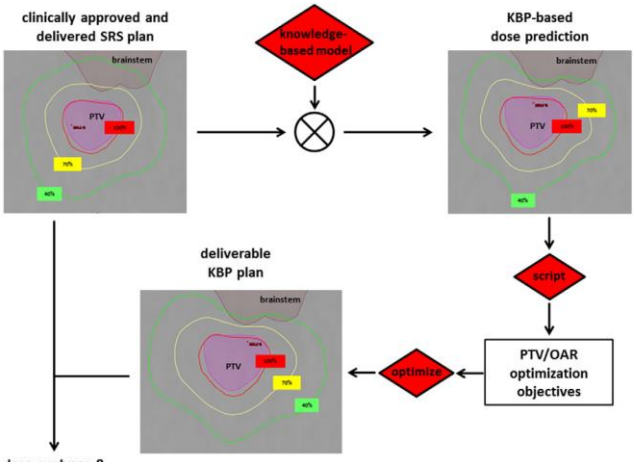

Figure 2.2 – KBP flowchart describing KBP plan generation and final comparison to the manually generated, clinically delivered plan. (Ziemer, et al., 2017)

Treatment planning automized has doubtlessly improved in quality and ease of generation, and while this improves the outcomes of radiotherapy treatments, it also means the increase cost both in time and increased complexity of generating the plans. There are multiple ways of alleviating this issue, one being an algorithm which can predict the dose distributions in a way that is still accurate but much faster and thereby cheaper for hospitals (As treatment planning requires the full attention of the physician, thereby inducing an opportunity for care cost).

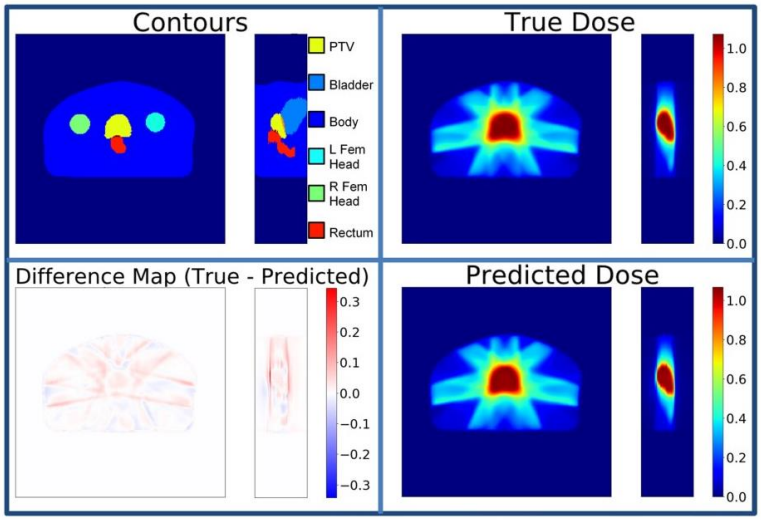

Figure 2.3 - Contours of the ROI, true dose, predicted dose, and difference map. (Nguyen, et al., 2019)

In Nguyen, et al., 2019 they have modified a convolutional deep network model, a U-net network that had been used for segmentation purposes (2.1.3)), into a network able to predict the dose distribution from images with pre-defined PTV and OAR. And were able to thereby predict the dose distribution of IMRT for prostate cancer patients. This application would act more as a preview of potential changes by the physician, allowing them to view the dose prediction immediately without waiting on a simulation to run, and then convey how they desire for the plan to changed or adjusted and cutting down on the total planning time.

Extraction of high dimensional data from radiotherapy imaging (Radiomics) has been a long-standing prognostic tool for cancer risk assessment. In Vallières, et al., 2017, one of our data sources 1615 radiomic features were extracted from the contained medical images, this data was then used to create prediction models via random forests and imbalance-adjustment strategies. The results from the study proved the potential impact radiomics has for risk assessment and tumour outcomes and has significant potential for allowing better personalisation of chemo and radiotherapy treatments. However, this approach only uses the tumour data before treatment, For example the CT and PET imaging. It does not take into account the radiotherapy treatment choices nor the underlying dose distributions. This is where we hope to make a difference with our approach, which will take these factors into consideration as well.

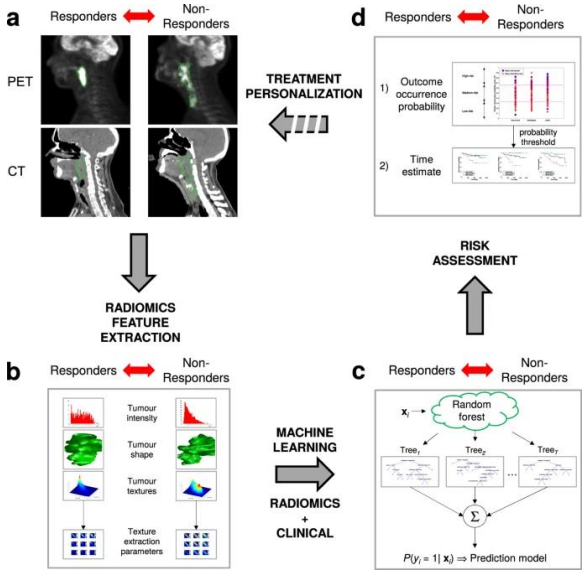

Figure 2.4 – Description of the workflow in Vallières, 2017.

2.1.3 ROI Segmentation & Contouring

Segmentation of the different ROI for treatment planning has been a time-consuming progress and often times subjective, even with the use of algorithms to automatically fill similar regions, the subtly of different tissue magnitudes has always presented a problem to existing techniques. Therefore, there have been multiple attempts to introduce ML into the segmentation and contouring process, specifically with the liver and other difficult areas.

ML lends itself well to this application, as the network can learn the more subtle differences both in pixel magnitude and spatial distribution, that physicians can identify while still being able to perform quickly and efficiently like an algorithm. The typical approach is a CNN classifier, usually being done through 2 layers, a region identifier followed by the fine segmentation network (Figure 2.5). This gives each pixel of interest a value between 0 and 1 in terms of the probability that it is part of the ROI that is targeted, then finally a probability cut-off can be specified and the mask can be extracted.

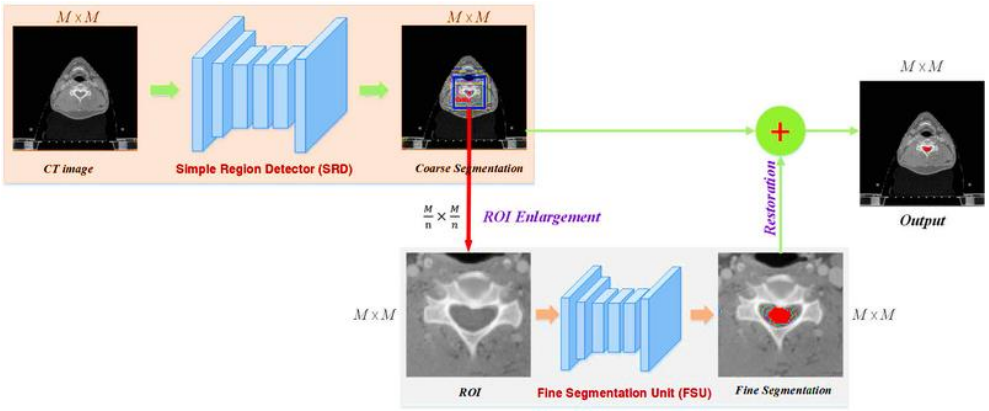

Figure 2.5 - The overall framework of the work by Sultana, 2020.

We have highlighted two recent applications of ML in segmentation, firstly an OAR segmentation network developed for the head and neck region, segmenting areas such as the parotid and mandibles (Men, et al., 2019). The second highlighted application (Sultana, et al., 2020) is the segmentation of the prostate, bladder and rectum (Figure 2.6), joining the segmentation of both the OAR and Target ROI, this time applying a U-net CNN method, and working down from a coarse to fine segmentation method. Both applications follow similar base methods but use slightly modified variations to achieve their goals.

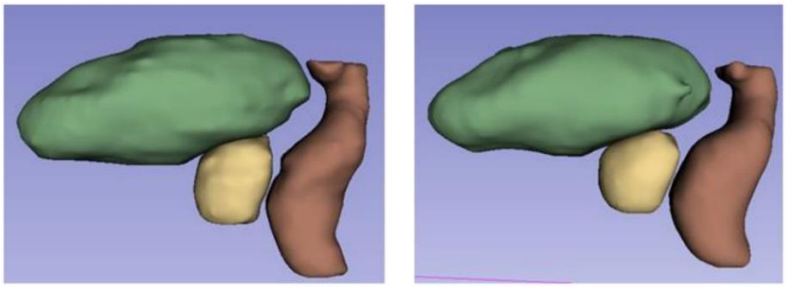

Figure 2.6 - 3D representation of the segmentations (yellow: prostate, green: bladder, brown: rectum).
***Left:** Ground truth. **Right:** Proposed automatic segmentation.*
(Men, et al., 2019)

2.1.4 Monte Carlo Dose Prediction

Monte Carlo (MC) is used for more accurate dose prediction in radiological simulations for research and design purposes. MC is a probability based iterative step simulator, providing the most complex and in accurate method currently available. However, MC is extremely resource intensive and often requires computation times that are unacceptable in a clinical setting.

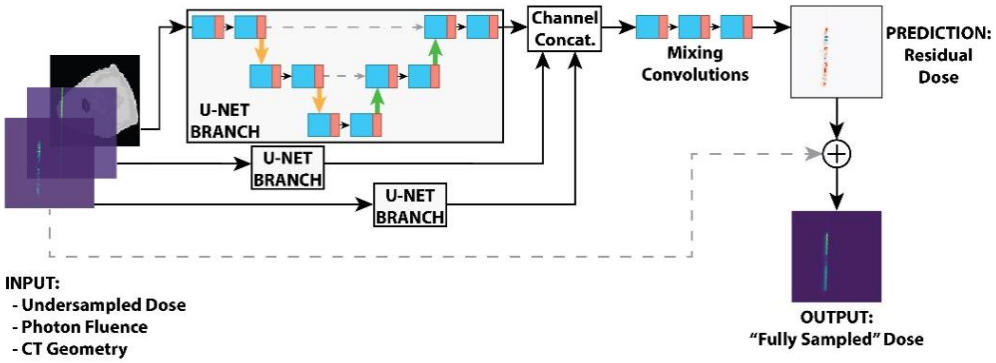

Figure 2.7 - Monte Carlo dose prediction network architecture (Neph, et al., 2019)

The next great leap toward improving radiotherapy dose prediction will be integrating MC into clinical RT solutions, Neph, et al., 2019 created a prototype deep convolutional neural network that allows significant acceleration and accuracy for MC dose results. It does this by relying on data-driven learned representations of low-level beamlet dose distributions. Using parallel UNET branches with 3 inputs (Under sampled dose, photon fluence and CT geometry), and later mixing latent understanding it can produce noise-free dose predictions in significantly shorter computation times.

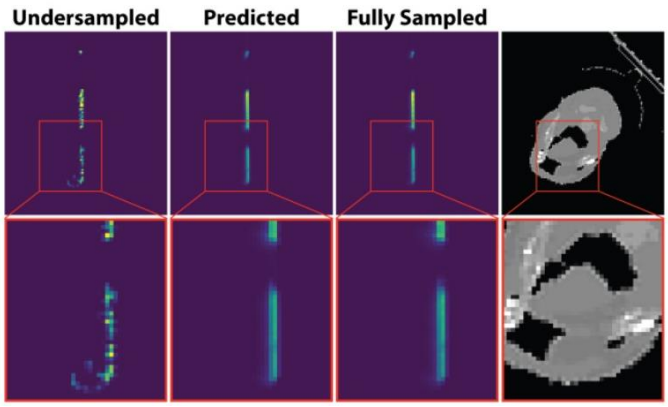

Figure 2.8 - Comparison of under sampled, predicted, and fully sampled for one beamlet (Azar Sadeghnejad-Barkousaraie, et al., 2020)

MC can also be integrated with ML for other radiotherapy issues. For example, (Azar Sadeghnejad-Barkousaraie, et al., 2020) created a beam orientation optimization algorithm via a deep learning reinforcement learning strategy using Monte Carlo Tree Search. The NN is used to guide the branches of the Monte Carlo decision tree and help the optimiser to decide if and how to add a new beam to the plan. Through testing it was found to perform much faster with similar accuracy to its contemporary solutions. Again, showing how machine learning can be merged with existing technologies to improve and innovate in the discipline of radiotherapy.

2.2 Novel Approach

Our application takes a unique approach to optimising treatment planning through deep learning. Instead of working preapplication in terms of theoretical ideals, we aim to pull data from patient metadata, and importantly the corresponding treatment outcome information. This way we hope to bridge the gap that usually is only possible through many years of experience and detailed analysis of past treatment plans, For example the personalised treatment planning techniques and approaches of a physician.

Through the acquisition of a large set of radiotherapy plans, as well as the associated patient metadata, and treatment outcomes, we will extract the most critical patient information, and the key treatment choices that were made by the physicians while treating the patients. We then standardise the information into an accessible and robust form that we can use to train our fully connected neural network (FCNN), to identify these choices and associate them with the type of patient they were applied to. Importantly we will train the network which applications worked, and which did not via the post treatment outcome information, thereby allowing it to predict how well a specific treatment might turn out.

With a working prediction network, we hope to take the next step, and provide a way to improve the treatment plans. We hope to do this by integrating our machine learning network into a radiotherapy treatment planning software (for example MatRad) optimisation method, and use the network as a secondary regularisation function to adapt plans to better fit to the patient's metadata. Ideally the network integrates semi smoothly and acts as a proof of concept that deep learning can be used to personalise treatment plans.

Our metadata guided RT planning optimiser hopes to bridge the gap between planning and treatment outcome prediction, by allowing physicians to predict the outcome of the plans they prescribe and through the integration into treatment planning software, personalise the treatment plans to their patients and their needs. As well as reduce the intra planner differences in treatment plans, specifically between experienced and novice physicians, where decisions based on past experiences related to patient circumstances is most likely to occur. Ideally, allowing our proof of concept project expand influence of machine learning in radiotherapy.

3 MATERIALS AND METHODS

Note: This thesis was programmed using a combination of MATLAB v.2019a and Python v.3.7.1, making use of the following packages listed in no particular order: NumPy (Oliphant, 2006), matplotlib (Hunter, 2007), Pytorch (Paszke, et al., 2017), SciPy (Oliphant, et al., 2007), shapely (Gillies, et al., 2007), pandas (Mckinney, 2010), scikit-image (Van Der Walt, et al., 2014), PyDicom (Mason, et al., 2014), Colorama (Hartley, et al., 2017) & VTK (Martin, et al., 2008).

3.1 Data Acquisition

There were multiple requirements that had to be met for a dataset to be accepted for use. These were as following: Inclusion of Patient Metadata (Age, Sex, etc.), Treatment success metrics (If & when death occurred etc.), either CT or PET image data, and finally: RT- plan, -dose and -struct files. Only if all these were included would we accept and include it as part of the training data set.

This meant that we had to specialise on a particular type of cancer, finally deciding on head & neck as only in this case were we able to find enough patient data which fulfilled the previously mentioned requirements, and had large enough quantity of patient examples to allow for significant feature identification and weighting through training

3.1.1 Source 1

The first acceptable set is from Vallières, et al., 2017 This compilation of patients (initially used for risk assessment of recurrences, see 2.1.2) from various hospitals in Canada contains 300 patients, with an overall death rate of 19%, recurrence rate of 28%, and a median follow up time of 43 months. Thereby it works well as a good assessment of both good and bad treatment plans, allowing us to train our model in what it ideally should avoid (due to the high amount of un-ideal treatments).

There are however some issues with this set which complicate the pre-processing procedure. These are as following: Firstly, a large section of the patient data comes from the French province of Quebec and are as such primarily labelled in French, making ROI classification standardisation difficult and will require translated standards. Secondly the underlying data structures are not all consistent, as multiple different machines and procedures were used. Some involving Static (Volumetric modulated arc therapy) others Dynamic (Intensity-modulated radiation therapy) treatment plans.

3.1.2 Source 2

The second set is from Grossberg, et al., 2017, which was an investigation of BMI and other body metrics before, during and after treatment. And as such has much more in-depth metadata for us to analyse. It also boasts high consistency regarding treatment methods used, and thereby a more standard underlying data structure. However, the set only has 215 patients, and as such would be difficult to use by itself and will therefore be joined with the first data set for our final working dataset of 515 patients.

3.2 Pre-processing

Note: All pre-processing is elaborated in depth in the CD attached "Data Extraction and Processing" Report. To avoid repetition this report will only go into the critical details & changed procedures, or cases where specialised approaches were implemented.

If the data sets were used in their original states, the training data would contain almost 160GB. An amount that would be impossible to train effectively in a FCNN, without first extracting the relevant information, and filtering out only that information which in the end influences the success of a treatment plan. This was achieved through multiple steps, starting with the CSV metadata, then working through the various DICOM files patient by patient, extracting the required metrics, and synthesizing a scoring system to rate the plans based on outcome. Finally, we unify all the data into feature sets for each patient, that can be used for network training.

3.2.1 Metadata

Meta data was primarily processed through MATLAB, as the data is in table form and therefore more suited to its ecosystem. First step of extraction was to choose which information was deemed important to the project within our data sources, they were then combined, into a single 'MetaData.csv' file. The chosen information we chose to extract are the following:

- Patient ID
- Sex
- Age
- TNM Group Stage
- Last Follow Up Date
- If Locoregional Recurrence Occurred + When (Days)
- If Distant Recurrence Occurred + When (Days)
- If Death Recurrence Occurred + When (Days)

Classification and Standardization was completed through primarily string search algorithms looking for characteristic labels in the data sets, which are known to be used for various classifications. These then were converted to integer values, that can be later used within the network construction. (Since a neural network cannot interpret text as a direct input, without pre-processing)

Following this we investigate relationships between tumour site, stage, death rate, recurrence rates etc. giving us a better idea of our data sample distributions. We also worked with the conditional structures, to better understand how we were to later construct our quality score, and how the various outcomes should be weighted.

3.2.2 DICOM Extraction

DICOM extraction acts in 3 layers. Working outwards: firstly, the Data level, where we select a patient, pull the relevant data from the patient level and dump said data into a text file. The next layer is the patient layer, which handles the pulling of DICOM data, and feeds it into the relevant extraction and classification functions. Finally, the DICOM import level. This function cycles through each file/folder under the patient, looking for RT-Dose, RT-Plan and RT-Struct files, as well as PT or CT images which it can then feed into the 4 subfunctions.

DVH Data Extraction

The DVH extraction subfunctions purpose is to extract Dose Volume Histograms for each ROI, within the patient. If the DVH are stored withing the RT-DOSE file, the extraction is relatively simple, first we need to check how the data is stored. Second, we must check if the data is cumulatively or differentially stored, in the case of the latter we convert it to cumulative, and finally as a percent of volume.

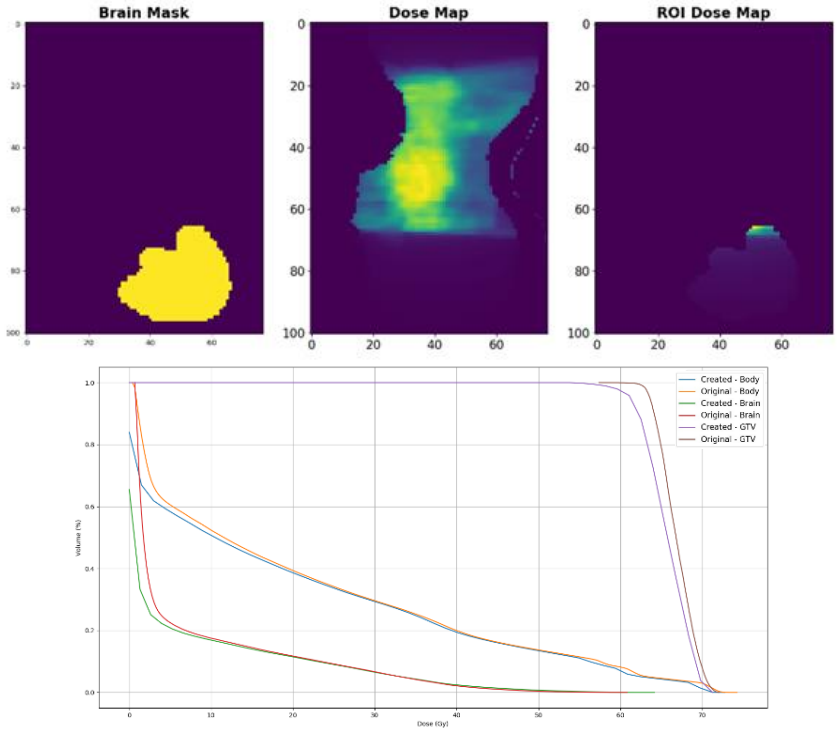

Figure 3.1 – ***Top:*** *Dose distribution, Mask Display and Overlay.* ***Bottom:*** *Resultant DVH estimation, compared to precalculated*

If DVH data is not stored within the RT-DOSE, the DVH-Creation function is activated. The first step in this case is to map the dose data to the struct ROI data. Once this is established, we can create a 3D masks of whatever ROI we wish to investigate. Then by multiplying the masks with the voxel dose data we can create custom dose bins and calculate the DVH (See

Figure 3.1 for an example). This is finally normalised into a percent cumulative and ideally the outputs would be indistinguishable from one another, if not for the terminal informing the user otherwise

BEAM data extraction

The BEAM data extraction subfunction purpose is to give us definitive information on what kind of beams and plans were applied to the patient. The first step of this module is to check the type of plan we are working with, in the case of Dynamic beam plans the steps are simple, however for a static beam plan we require additional steps. In the latter case, we often have multiple beam cycles, and it is important that we combine each individual and calculate their relative impact.

Once this option is accounted for, we can continue as normal, by working through the control points, extracting the angle, and the weight. Finally, we can normalise this angle and weight information against the total dose and convert everything into standardised values. Simultaneously for each control point we must calculate the area of exposure. The first step of which is to see if there are variable collimators in play. If so then we need to create the initial leaf settings, and area. After this we can update the collimator locations with the control points and calculate the exposed area from our previously acquired information (See Figure 3.2 for an example).

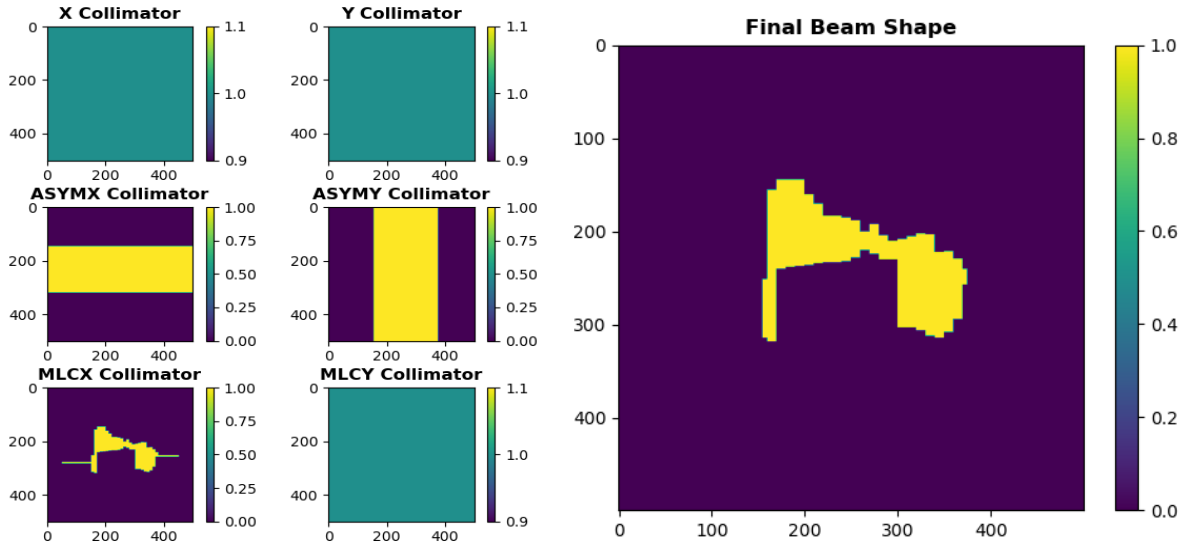

Figure 3.2 – Visualisation of beam shape calculation.
***Left:** Individual collimator settings, **Right:** Final beam shape*

`ROI Extraction and Classification

The ROI subfunctions purpose is to classify and identify what each ROI describes. By first creating a list of keys, for each ROI type we wish to consider, making sure to avoid overlapping and including French versions of each labelling metric. The method works by checking each ROI name against the key lists sequentially. Checking each key within the list against the name, and once a match is found skip ahead to the next ROI (to preserve efficiency)

OAR Distance Extraction

The OAR subfunction is to calculate the distance of each OAR to the GTV. The method works by creating a 3D mask of the ROI in, as well as any GTV's in the Struct File. We then create a 3D distance map and overlay the ROIs Mask, within which the minimum distance between it and the GTV is extracted. For efficiency if a Mask had already been created in the DVH subfunction it would be transferred forward and used here, to save on per patient processing time.

3.2.3 Score Synthesis

The scoring system was designed to collapse multiple quality factors down to a simple one integer value, that could be used to designate the overall quality of a respective treatment plan for that specific patients' traits and their initial state. The scoring system is a normalised score between 0 and 5, to avoid splitting the plans into to specific feature groups. It does not consider all data, instead focuses in on tumour stage, reoccurrences, and death, as well as modifications based on the time between them.

The final abstracted formula ends up as the following:

$$\text{Score} = 5 + \text{TumorStage} - \text{Death} * \text{Time}_{\text{D}} - \text{Reacurrance} * \text{Type} * \text{Time}_{\text{R}} - \text{Time}_{\text{FollowUp}} \qquad (3.1)$$

Final Score Synthesis equation. Values are integers, with time values acting as scalar.

The concept is simple, first we give each treatment the average value of 5. Then the higher the tumour stage (the more advanced the tumour), we assume it would be more difficult to provide a successful treatment to the patient; therefore, tumour stage adds to the overall score. We then subtract anything related to an unsuccessful treatment. This relates as to why we chose to place the score between 0 and 5, as there are 6 general states a patient could be in: Death, Recovery, Local, Distant, Both Recurrences and one of 6 different tumour stages. Thereby we made our bins range from 0-5 to allow for each case to be represented, but not so many bins that the prediction would become too complex and erratic.

Death is the harshest of penalties, followed by reoccurrences and lastly a short follow up time. The reoccurrence score is additionally modified according to the type of reoccurrence that has occurred, to account for the severity of each type. Each of these factors have weights that can be adjusted according to observed relationships in the data. The time factors are weighted depending on how long it took each of these negatives to occur and would be normalised against each other. In these cases, the shorter the time is, the harsher the penalty. (For example. an early death is penalised more than a death 2-3 years after treatment finishes). Finally, the score distribution is normalised and multiplied by five, to return a score distribution that is easily binned into 5 different score classes for our network training.

The underlying concept behind using a single score to describe a treatment, instead of the treatment outcome parameters (Death, last follow up, etc.) has 3 main reasons behind it. Firstly, as mentioned above, it gives us the ability to judge treatments with more variation, avoiding the assessment from being a binary choice, something the network might struggle with. A properly distributed score on the other hand would allow the network to understand the nuances of different plan/metadata combinations, and train on a wide range of cases. Secondly, as we intend on training a network for use in treatment optimisation, the output must be a single value easy to use in an objective function, thereby a score for the treatment (Planned/Predicted) would be ideal. Lastly, by using a score and binning methods it enables us to use a classification network rather than a multiple feature output. This avoids the binary choice issue mentioned earlier, (as from experimentation we know that the network would generally predict the mean value for all treatments to minimise the loss), as well as letting the network develop a clear idea of what classifies a good or bad treatment for each metadata combination.

3.2.4 Feature Conversion

The first step of converting our data sets to a workable feature set was to unify and standardise the various features. This came from a series of issues, firstly the metadata was still saved as a CSV file, and we had to link the DICOM data to it accurately. Secondly the beam and DVH data while complete needed to be further abstracted into forms that are dimensionally consistent.

To overcome the patient matching we used the patient IDs which were extracted for both the meta and DICOM data sets, and sorted each data set accordingly. The next step was to standardise the BEAM information, in its current state each patient would have varying lengths of data, that were not constant with each other, therefore we decided the best approach was to summate the beams that were coming from a particular direction. Deciding on 45-degree chunks as the pooling bins, into which we added Monitor units, Area of Exposure and Fluence (MU * AoE), then normalising across the bins, to find the relative weights. Keeping the total sum of each value, thereby giving us both scalar and distribution of all beam values in a constant dimensional form.

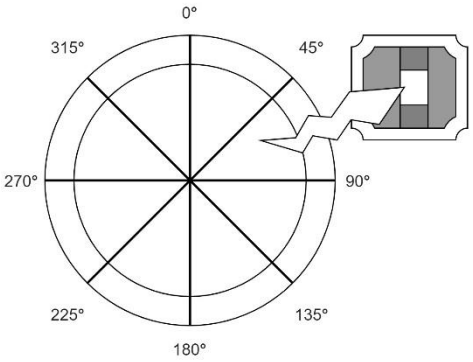

Figure 3.3 – Beam data pooling according to angle of exposure. For simplified and standardised feature dimensions.

The DVH had similar issues only that in this case it was not only different dimensions per patient but also per ROI. This is because the existing DVH's and our extraction method only saved what was necessary and did not involve standardising the dose bins. What we decided to do was to standardise the X axis (Bins) to be 5% steps of the total dose, this being the highest bin in the GTV or PTV, and to fit all other DVH according. This did involve the loss of some detail, specifically with more subtle changes in the OAR that had little exposure, however since we were much more interested in OAR that had high exposure (which could be optimised better) this ended up being less of an issue than first expected.

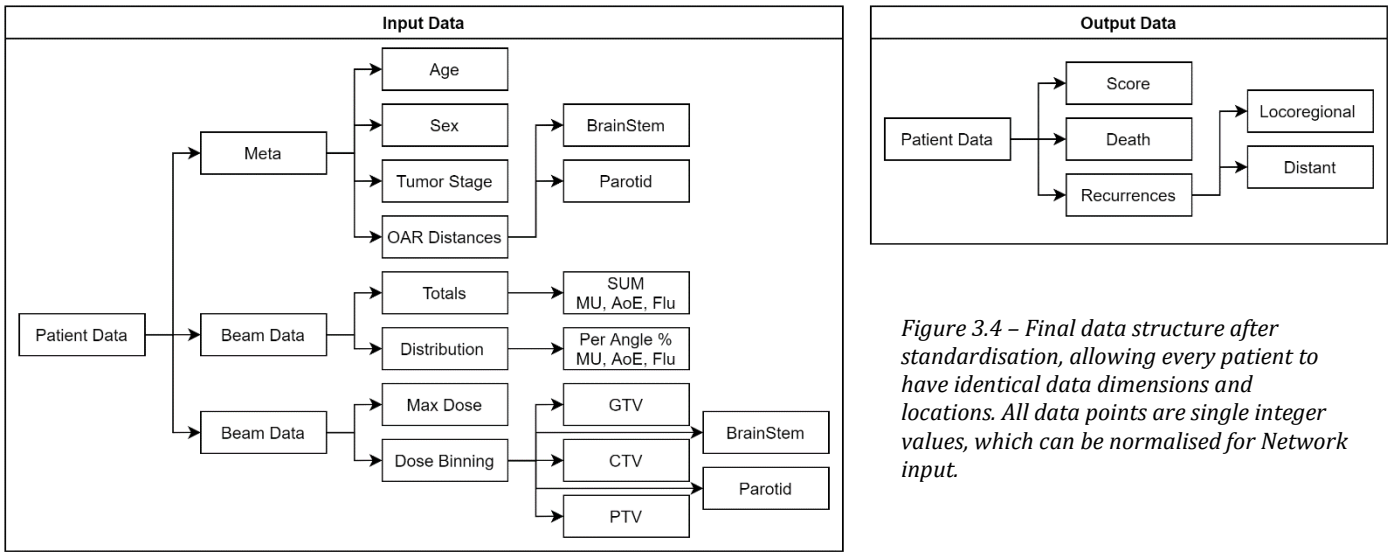

Figure 3.4 – Final data structure after standardisation, allowing every patient to have identical data dimensions and locations. All data points are single integer values, which can be normalised for Network input.

Finally, we unified and exported our patient's individual data, Metadata & Treatment Plan as the inputs and Treatment Outcome as the output. These were stored as layered NumPy arrays for easy processing/loading when it came time to train and evaluate the network. As there was a lot of inconsistency as to what data was available per patient when it came to the DVH, we saw fit to only take the most common or only the most critical/relevant to treatment OAR DVH's and fill in gaps with an overall mean DVH. To avoid having to train on incomplete data, but also avoid having to much artificial data and get a skewed model.

There is an argument for practicing data augmentation to fill out data gaps or even leaving the gaps in the dataset and let the network figure out how to approach the missing features. Leaving the features blank is not an option for us, as we have a severely limited data set, and need all the training data we can get. As to augmentation, there was an early attempt to integrate it for DVH information, however this proved difficult as the ROIs should be segmented by a medical expert, and therefore did not attempt to contour any ROI ourselves, light modification of existing DVH to replace the missing was additionally, however these could be wildly wrong and as such we decided to augment the data by using the average values/parameters, which ideally would cause the least amount of disruption to the training of the network.

3.3 Autoencoding Features

The decision to encode certain parts of the feature set came about when we realised that to increase efficiency of training and more importantly avoid inputs that did not have key impact. Autoencoding would be the easiest method to implement and would serve our needs well. Specifically, with the DVH features which with 10 features per ROI (50 features in total) and the beam information which at 27 would cause a significant unnecessary increase in the complexity of the training. When instead we could compress these inputs and thereby only input the key features, and still get a reliable model.

All inputted features had to be normalised between 0 and 1, as compared to the same features in all other patients. This way we speed up learning and leads to faster convergence, as well as help avoid infinite and null gradients. With us eventually being able to use arbitrary inputs that do not originate within our original data set, by normalising by the same statistical values (This is required for the final optimiser integration). Training was done via L1-Loss backpropagation; however, measurements were taken of both L1-Loss and MSE-Loss to better observe the training process.

3.3.1 Beam Features

Once the beam data tree is squeezed the resultant tensor is 27 values long, with 3 summation values, followed by 8 sets of 3 fractional values. The intent here was to achieve a 50-60% compression of the beam data (excluding the total MU/fluence/merged). We reasoned this would be feasible as each of the 8 sets would have to relate to one of another, as by definition need to sum up to 1, and the set of 3 summation values would influence each other.

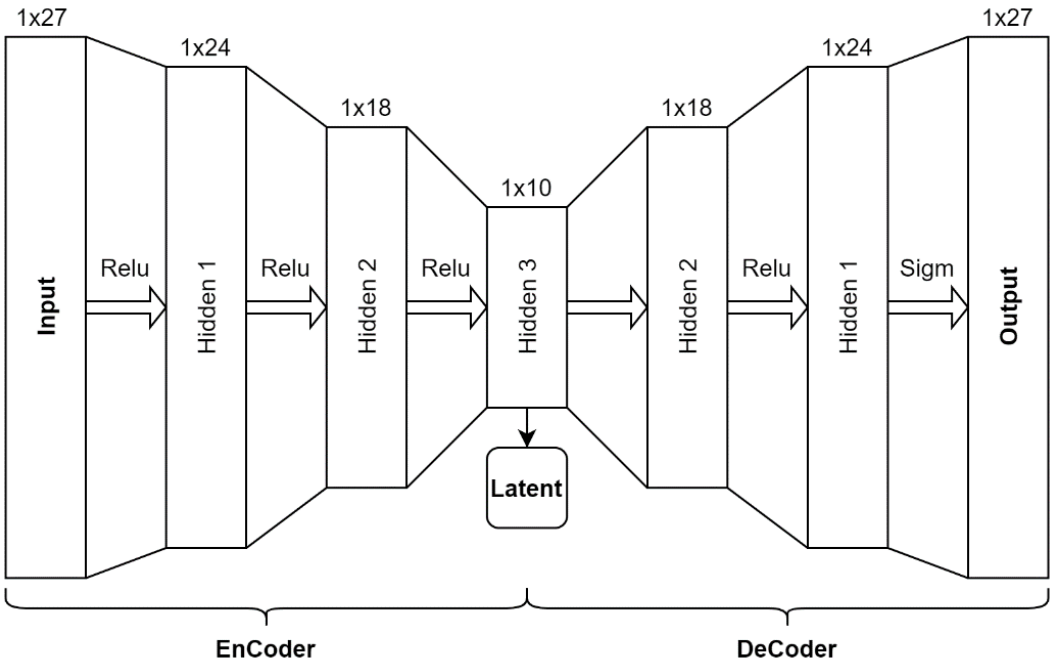

Figure 3.5 – Beam Autoencoder structure visualisation. With Activation function and pooling to scale.

The Final Network was constructed with 3 encoding and decoding layers, excluding the initial input layer. With the hidden layers being respectively 24,18 and 10 wides, the third being the encoded output. Each layer was activated using the ReLU function before pooling into the next layer, with the final output layer being smoothed using the Sigmoid function to ensure an output normalised between zero and one. The Adam optimiser was implemented in this case, with a learning and decay rate of: $1x10^{-5}$ and $1x10^{-8}$ respectively, with a total of 1000 training epochs. For data loading we decided on using batches of 5 for increased training speed per epoch but avoiding larger batches which caused unstable reconstructions.

To compensate for the lack of data it was decided to use k-fold cross validation, thereby simulating a larger data set by introducing variety. For this network we decided on a 4:1 split. With our initial data split being 20:80 Validation vs Training, this means that 20% of the training data would be kept out the epoch's backpropagation, with the remaining 80% being split again each epoch via random seed to ensure a fresh data mixture being trained with per epoch. This helped the network create a general case model and avoid overfitting to specific examples.

3.3.2 DVH Features

After squeezing the DVH data tree, the resultant tensor is a total of 100 values long, with 5 sets of 20 DVH values. This is without question to long for a network to realistically interpret and train on and would cause significant performance issues. Our for this encoder was to achieve a 70-90% compression. This is because each of the sets could realistically be compressed by 50% by manual methods, and it would only a matter of stress testing the autoencoding network, to see how much further we could compress our features before the loss would no longer converge acceptably.

The Final Network used individual sets of DVH values (1x20) as the input. Constructed with 3 encoding/decoding layers after the initial input layer. With the hidden layers being respectively 10,5 and 3 values wide, the encoded output being 3x1. Each layer was activated using the ReLU function before pooling into the next layer, with the final output layer being smoothed using the Sigmoid function. A simple Adam optimiser was implemented with a learning and decay rate of: $1x10^{-3}$ and $1x10^{-7}$ respectively, with a total of 400 training epochs.

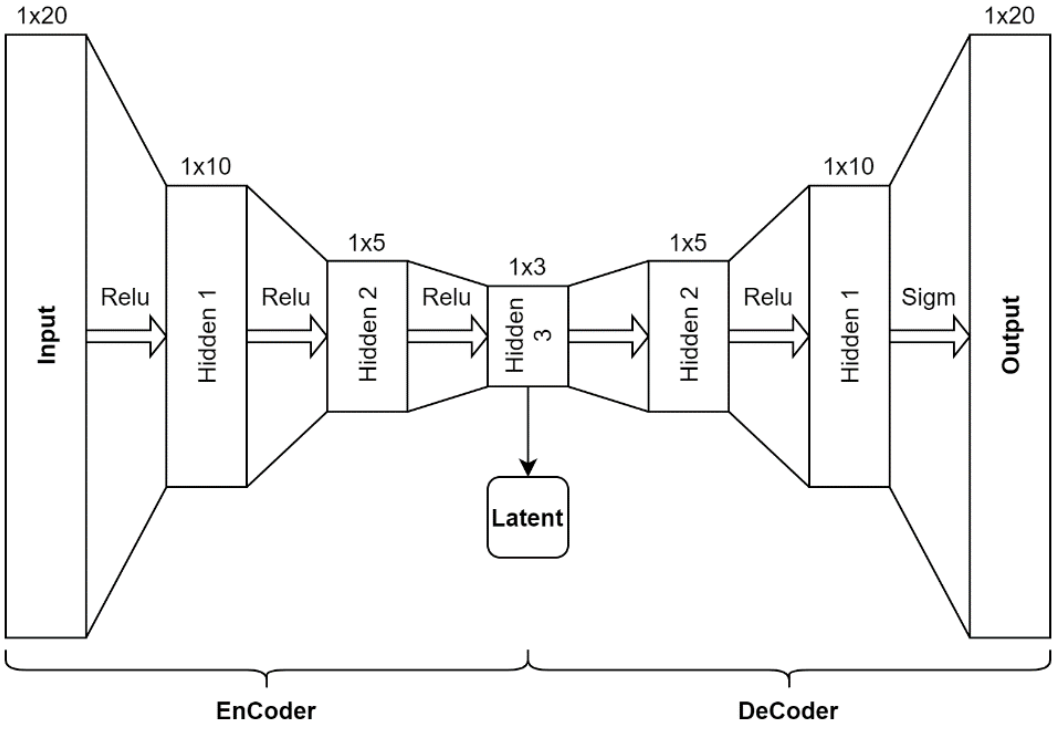

Figure 3.6 - DVH Autoencoder structure visualisation. With Activation function and pooling to scale.

As the network used individual DVH sets as the input, the available training data was 5 times more than what we had for the previous beam autoencoder. However, to maintain a general solution we kept the k-fold splitting method. For this network we again decided on a 4:1 split, after the initial 30:70 split for the validation data set. This means that 30% of the data would be kept out the training loop, with the remaining 70% being split again each epoch via random seed to ensure a fresh data mixture being trained with per epoch.

3.4 Predictor Network

The predictor network is responsible for predicting a treatments quality score using only the patient's metadata and the treatment data. The ideal case for this network would be perfect prediction of both the training and validation data. However due to lack of training data and the underlying nature of cancer treatment there is a likelihood of case memorisation and various edge cases in the validation set being impossible to classify. (As an example, a patient with a stage II cancer, who at the age of 22 died within 300 days of receiving treatment). An acceptable accuracy value in the validation set would be around 60-70% with near miss classification/soft accuracy (Off by less than 1 point/class) at 70-80%. As at these levels we could confidently apply the network on new cases without our prediction being equal to random guessing (which would result in a score accuracy of approximately 40% due to the normal distribution of our scores).

Our prediction network is based on the concept that each plan involved the physician making specific choices per patient, due to their experience in the field. Our score metric reflects how good those decisions were, and thereby how well personalised the plan is. Our network will be trained to identify the treatment choices in our training set, alongside the metadata information. Therefore, if the plan and patient are matched perfectly the plan should be classified as a 5, if the plan is detrimental to the patient it would be classified as a 0. Thereby in practice, the network will be steered by the fixed metadata values, with the plan feature values changing as different planning options are classified, allowing the network to return a reliable treatment outcome, score value.

3.4.1 Network Properties

The final input tensor for each patient was of length 31, after the encoding of the DVH data from 100 to 15, and BEAM data from 27 to 10. With the remaining 6 data points being filled with metadata and OAR distances. If the autoencoding step had not been added the predictor network input would be of length 133 which would have slowed down training significantly, and not necessarily resulted in a better model.

Age	Sex	Stage	BEAM										Bstem Dist.	Parotid Dist.	Max Dose	GTV DVH			CTV DVH			PTV DVH			BStem DVH			Parotid DVH		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table 3.1 – Final Feature set per patient, including data distribution per feature and the corresponding feature location in the tensor. Blue represents Metadata and Green represents Treatment Data. The meta data remains constant per patient, while the treatment data can be adjusted and optimised.

The final version the predictor network ended up having 5 hidden layers, which expanded in scope for more complex calculation. Until finally pooling back down into the 6 output classes [0 1 2 3 4 5] which represented the possible scores a treatment could be assigned. The choice to use a classifier network and not a single value predictor came from the thought that the treatment outcome would statistically end up being a probability problem, with various factors influencing what quality score it would skew towards. And it would be possible to calculate a final float score via using the various probability weights, either via network or a simple mathematical calculation. For this purpose, we rounded each individual score towards its nearest integer value, to accommodate the classification network output.

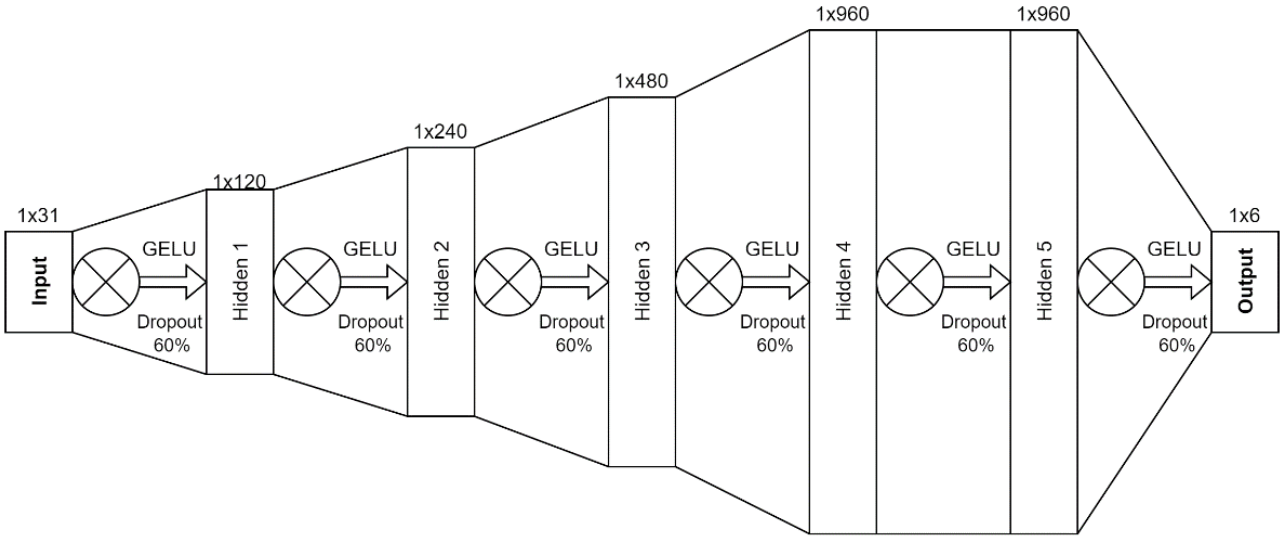

Figure 3.7 – Predictor Network Diagram (Control), FCNN with dropout per layer, and GELU activation.
Note: When running network outside of training, dropout will be disabled (no influence on treatment plan scoring)

We ran multiple different setups for training around the control network which was structured as such: An input layer of length 31, which was pooled into the 5 hidden layers of lengths, 120,240, 480, 960 and 960. Until finally being pooled down into the 6 output classes. Each hidden layer was activated using a GELU activation function, and we initiated a dropout sequence of 60% for each subsequent layer. This dropout proved critical to avoid the network from overfitting and memorising the training data. As without it we would not get any acceptable accuracy results within the validation set, as the network would overfit and memorise the training set too quickly, before a general solution was achieved. The dropout is shuffled with each activation, thereby ensuring all neurons will eventually be trained.

The Adam optimiser was used to train the network, with the default learning and decay rates of: $1x10^{-5}$ and $1x10^{-8}$, respectively. The loss was calculated via a Cross Entropy Loss function for classification network training, which allowed us to obtain a probability metric for each Class/Quality Score. The control training setup involved a 4:1 k-fold spit, with a batch size of 5, and 1000 training epochs, with a validation split of 15% stratified according to the scores of the data sets, to ensure even training and quality assessment of the training. The low validation split is due to the relatively small amount of patient data, and the network needing as large as possible training set, while keeping enough validation data sets for robust network quality assessment.

During model training we utilised CUDA integration, a technique to run the model training on the workstations GPU to allow for parallel processing and faster training times compared to normal CPU training. This is usually used for image-based training in convolutional neural networks, however, can also be applied for FCNN as in this case. With the use of CUDA we reduced training time by 50%, allowing us the ability to repeat our model training with different parameters without running into time constraints.

3.4.2 Training Method

Due to the uneven distribution of scores, it was decided to use a stratified splitting method when splitting the training and validation set, therefore ensuring that either set does not become easier or harder than the other. As well as allowing the training set to get a good range of edge cases, as they are already few in the total set, we decided not to make an official Test set, instead only splitting our data set once at start. This was followed by using the shuffled k-fold spitting of the training set per epoch, to avoid overtraining, as well as the dropout function, allowing the same training data to be trained on without encouraging as strong memorisation. Per epoch we send the networks results to a dynamically updating loss/accuracy plot as well as outputting the epochs stats to the command window, which returns a live progress of the training.

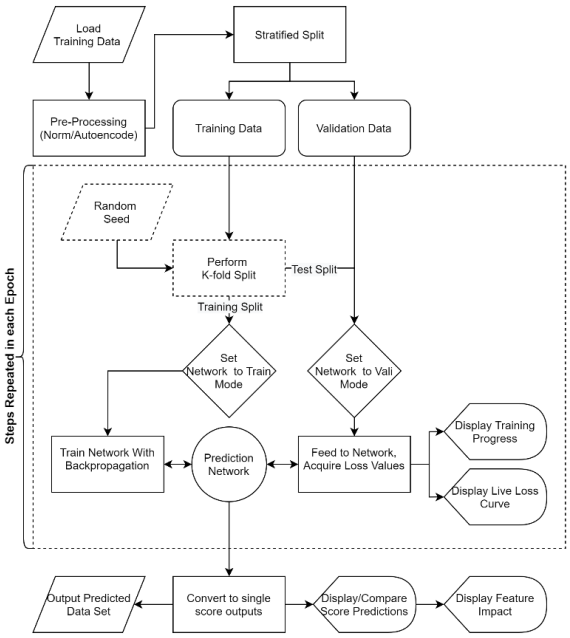

Figure 3.8 – Flowchart, describing training method for FCNN predictor model

To ensure that we can effectively diagnose issues and have the ability to work with any epoch of the training, we decided to implement a checkpoint system. This has two tasks, firstly it saves the models current weight at a set percentage interval of the total number of epochs. Secondly it saves the iteration with the 'best' model weights, this is determined by the highest training & validation prediction accuracy to that point, thereby allowing easy access to the final prediction model. Post training, we save the model structure, training history, data splits etc. All these are saved to a dynamically generated file at the start of training with a user defined name to enable easy identification. These folders can then be loaded into our network viewer function to extract statistics, and work with the selected network in a more stable non-training environment.

The step by step training method can be summarised via Figure 3.8. After the initial training of the control network, we adjusted the structure and training options until we get a reasonable outcome, this included network size, batch size, number of k folds, training step size etc. To better understand how we would be able to improve our network, both in prediction quality as well as training time and stability.

3.5 Optimiser Implementation

3.5.1 MatRad

MatRad is an open source software, for RT treatment planning of IMRT. It includes 3D dose calculation software for photon, proton, and carbon ion radiotherapy, as well as inverse planning software using MLC sequencing and biological weighting. It has a built in GUI for ease of use, and since it is entirely written in MATLAB, the code is easy to access and modify if required. The underlying optimiser is the IPOPT however this can be swapped out for the inbuilt MATLAB linear optimiser function if required. Additionally, since MATLAB can be easily interfaced with python, and as the optimiser's objective/ gradient functions are not compiled, it lends itself perfectly to integration with our planned machine learning network (Bangert, et al., 2015).

3.5.2 Network Mounting

The python model was mounted within MATLAB using the MATLAB Engine API for Python and Python to allow MATLAB to run python code and vice versa. This allowed for relatively seamless integration of the model into the MatRad optimiser. The model did however have to be rearranged and implemented as a single use return function, as well as giving feedback through repeated application of the model. As our testing is on existing patients within our data set the metadata input was kept within the python function and would draw from the data repositories. It can be modified to accept external input however, if the user wishes to experiment with new patients which are not included in the repository.

3.5.3 Objective Function

The Objective function involved transforming the internal IPOPT weights into the Dose and Beam values as required for the autoencoders (See section 1.2.4 for theory). By using the current iterations weights of the plan optimisation, we created a dose voxel grid/cube and acquired the updated MLC locations. This was then used to acquire DVH information for the relevant ROI, using the DVH with the highest total dose (To assume a worst-case scenario, as was done during the data extraction). The beam information was acquired using the shape vector beam, beam shapes were extracted from the MLC data acquired earlier, which were converted the input form for the autoencoder.

As the optimiser tries to minimise the score, it was decided to invert the score output, as well as apply an exponential power the value, to better visualise the progression of the optimisation. Additionally, to avoid the optimisation failing, or going outside of biologically acceptable bounds it was decided to run the inbuilt optimisation in parallel and weight the final objective function value towards the model defined function. However, this weighting can be modified to better reflect the goals of optimisation if required. What is important to note is that the objective function output does not influence the optimisations progression as it only returns the current state, the direction of the optimisation is defined by the objective gradient.

3.5.4 Objective Gradient

The gradient function involved implementing an iterative decent method, by attempting different decent vectors every iteration (The number of vectors can be predefined) and by running the predicted output through the objective function the optimizer can access what path to take. The gradient vector not only can vary in its direction but also in the learning rate, thereby allowing it to overcome local minima when necessary. Additionally, a boundary check function was implemented to ensure that the generated vectors do not exceed the physical limits of the system, for example Leaf positions. Once the best gradient for score improvement has been identified, we merge the gradient from our score model with the existing MatRad gradient function. As mentioned in the previous section, this is done to avoid going outside biological bounds, since the gradient is already limited by physical bounds in the previous step.

3.5.5 Process of Optimisation

This section describes the process of optimisation with our integrated network and will explain how our changes fit into the normal workflow of MatRad.

Optimisation with MatRad begins with the reading in of patient data, specifically the CT and RT-STRUCT, as these contain the minimum information required. If the process also includes an existing plan, one can also load the RT-DOSE and RT-PLAN, however this can sometimes cause compatibility issues. Once loaded the user will specify their initial parameters, this includes the gantry angles, type of radiation, number of fractions etcetera.

The next step Is to specify the prescription dosages for the ROI detected in the RT-STRUCT, as well as the objective and constraint functions. This is where the majority of the decisions in terms of plan outcome will be made, and what will have the greatest impact. This is because even with our network being integrated it will only act as a final tweaking mechanism/regularisation function for the optimisation. After this the dose deposition calculations will run, and the optimisation will begin.

The first function to be called is the optimisation problem, this will work with the specified parameters in the previous steps and convert them into the required form for optimisation. Then the IPOPT setup function is called, here the optimiser base settings are defined and modified, including optimisation cut-off conditions. Importantly the pathing to the objective and gradient function are located here and are specified to our modified versions. Additionally, we ensure that our metadata information is fed forward as an input for both the objective and gradient function, as it needs to be called alongside the network.

From here the IPOPT optimiser works independently, calling the objective function and gradient function in turn, finding the optimum gradient to take to reduce the objective value. When the objective function is called, we allow it to calculate an objective value using the dose prescriptions as it normally would, however we also call our prediction model, and invert the score giving us a score loss value. Finally, we combine these scores with our score loss being exponentially increased to ensure it is considered during the IPOPT iteration step, however not to high that it ends up breaking the prescription-based optimisation.

Before the iteration update the IPOPT optimiser will call the gradient function, where again we allow the dose prescription-based gradient to be calculated for biological steering. Followed by the optimum score gradient, which involves obtaining a gradient return from the network, finally choosing the gradient which would result in the greatest score improvement. The score gradient needs to be checked for boundary violations, and will only be applied once either corrected for, and if said correction does not cause a reduction in score value. Only in this case is the score gradient combined with the prescription gradient in a simple 9:1 ratio and sent back to the IPOPT optimiser and used to calculate the net iteration step. (Note that the score gradient is considerably smaller than the prescription and so never overwhelms the prescription gradient even at a 9:1 ratio)

Once the optimiser reaches a point that is considered acceptable according to the parameters defined earlier it will stop and return the final weights. We then run these through a collimator calculation function, which will set our MLC locations, and run the DVH extraction function. Following which the UI will be updated to display the heat map of the metadata optimised radiotherapy treatment plan.

4 RESULTS & DISCUSSION

4.1 Training Data

4.1.1 Metadata

The ideal distribution of data would follow a flat or normal distribution for all analysed features, while also having a variety of combinations, and variances. (While avoiding as many exceptions as possible). This way our network will be able to consider the influence of each metric while not being over trained on any specific exception and thereby causing the model to be biased.

Visualizing our data we observe some expected trends (Figure 4.1), with many patients being around 60 years of age, and a normal distribution of age ranging from early 20s to late 90s,while this is not ideal and a flat distribution would have been preferred, cancer tends to arise in older patients which makes young patients unlikely. Males outnumber females 4:1, this unbalance can be attributed to the cancer type we are working with, as head/neck tumours are often associated with smoking which has higher prevalence in men (Kreuzer, et al., 2000), however as the inbalance is still within reasonable tollarances it shoud not cause a bais in the calculations, as there are sufficent females to train an exception.

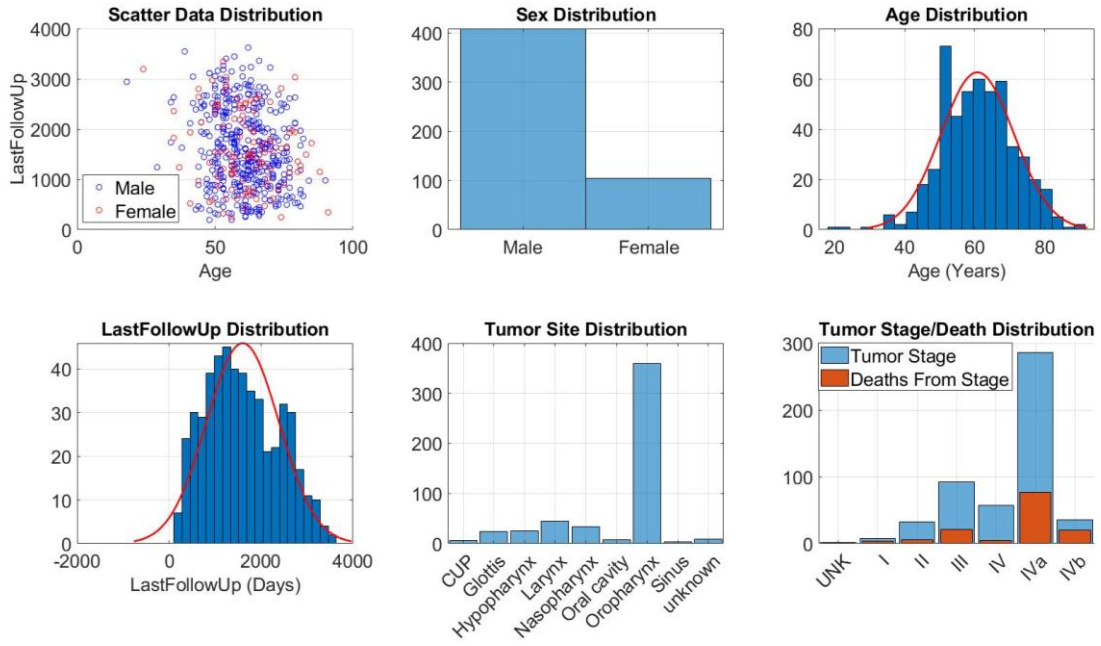

Figure 4.1 - Comparison Compilation of different Metadata Parameters
Top Left: *Age/Sex/Follow Up Scatter Plot,* ***Top Middle:*** *Sex Distribution,* ***Top Right:*** *Age Distribution*
Bottom Left: *Last Follow Up Distribution,* ***Bottom Middle:*** *Tumour Site Distribution,* ***Bottom Right****: Stage & Death Distribution*

Last follow up, is a significant indicator as to how reliable the data is (longer follow up means better assessment of treatment quality as there is more contact with physicians). It tends to drop off after the 1600-day mark which is often enough time to assume the patient cancer free (Werthmann, et al., 2018). Most cases where the follow up time is significantly shorter are usually unsuccessful treatments or low stage recovery, where recurrences/death or loss of patient contact can occur.

Tumour site is the most unbalanced of our metrics, however it is important to note that not all patients had a specific site given (many sites were simply listed as 'Throat') and were grouped with the oropharynx, thereby inflating that metric. Nonetheless it was decided to remove tumour location from our training metrics due to the imbalance. Similarly, the average tumour stage was rather advanced with the majority being type IVa or III however, with the percentage of deaths increasing alongside the cancers initial stage. The tumour stage was too critical for training purposes and therefore was maintained as a training metric even with the distribution imbalance we observed.

4.1.2 DICOM Features

After successful extraction of the core plan metrics it was important to investigate the underlying distributions and understand how different radiotherapy plans were applied. This was done after the intial data standardisation, both as a visualistaion of trends and as a way to ensure the standardisation didn't compromise the integrity of our extracted data.

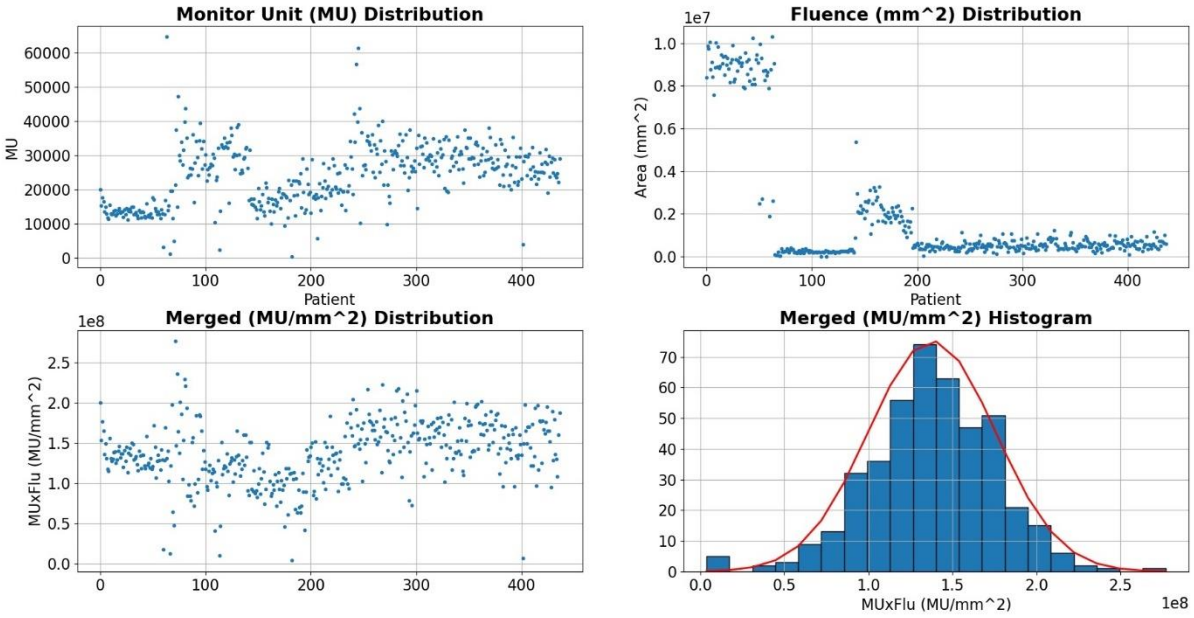

Figure 4.2 - Beam metric distributions per patient
Top Left: *Monitor Unit Values,* ***Top Right:*** *Fluence Sum Values*
Bottom Left: *Merged MU-Fluence Values,* ***Bottom Right:*** *Merged MU-Fluence Distribution*

Figure 4.2 describes the distribution of our extracted Beam information. With the Monitor Units, fluence and merged values being displayed per patient, showing us the different approches taken, for example high MU but low fluence or vice versa. Additionaly we notice that once multiplied/merged the values seem to converge to a simmilar range. The last plot is a histogram distribution of the merged metrics giving more wieght to the convergent nature of these beam plans as well as providing us a clean normal distribution, which will assist in the training of our model.

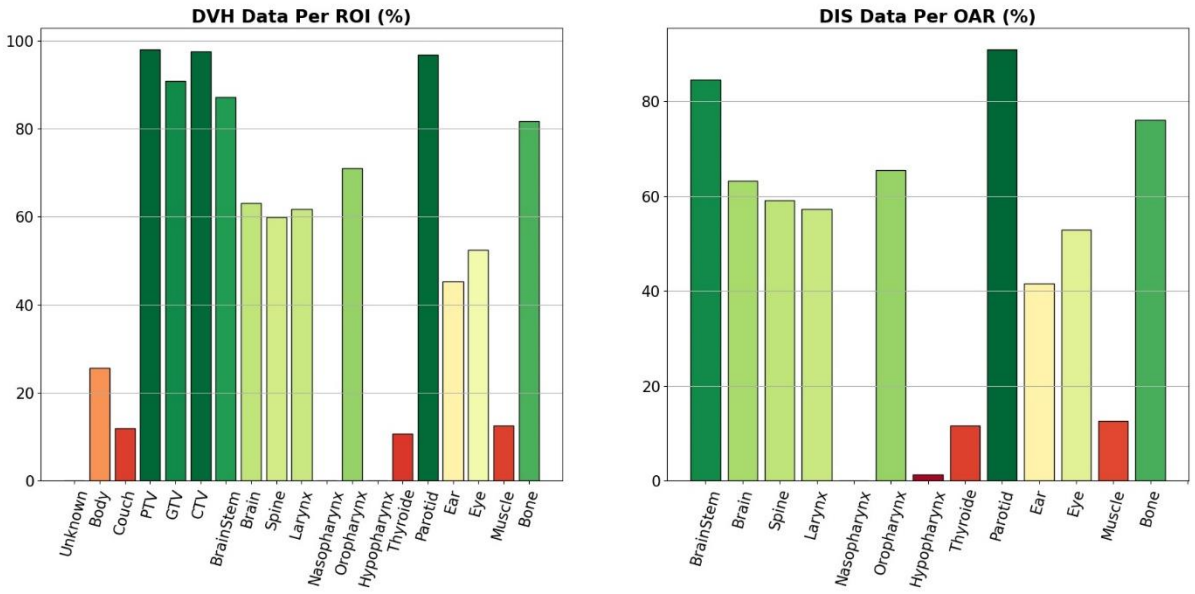

Figure 4.3 – Data Integrity Percentages
Left: *DVH Existence per ROI,* ***Right:*** *Distance Value Existence per OAR*

The Dose metrics could not be as easily compared one to one due to the underlying compexity of a DVH, additionaly the primary concern with our dose metrics is to maintain a data set that was mostly intact. Therefore in Figure 4.3 we displayed the percentige of patient data sets that had data for each ROI, for both DVH and OAR distance information. From this it was decided to only use data for the target ROI's and the parotid and brain stem, since these were the ROI best covered. The remaining data holes could then be patched using either a mean value or a slightly variented DVH as is required. Had we done the same for ROI that had larger holes we would have risked introducing bias into the network. Another option would have been to allow the gaps to remain (set the values at -1) however as we intend to autoencode our metrics this could cause issues in processing and was therefore avoided.

4.1.3 Survivability Score

After extraction and standardisation of the various post therapy parameters, a preliminary investigation of the relationships present was undertaken. This was summarised in Table 4.1, which was used to determine the scoring system that would be used to rank the various treatment plans and how well they worked for a patient.

Status	Had Recurrences	Local Recurrences	Distant Recurrences	Both Recurrences	Total Patients
Surviving Patients	49 (34%)	34 (43%)	13 (26%)	2 (11%)	379 (74%)
Deceased Patients	95 (66%)	44 (57%)	36 (74%)	15 (89%)	133 (26%)
Total	144 (28%)	78 (15%)	49 (10%)	17 (3%)	**512**

Table 4.1 – Table describing relationship of recurrence to patient survival

The first relationship we determined is that death is strongly related to reoccurrences, with 71% of deaths involving recurrences. This means that in the majority of treatments the primary tumour is controlled, and only post treatment does long term complications result in the death of the patient. Furthermore, the type of recurrence is important, only 57% of local recurrence resulted in death, while almost 74% of distant recurrences were fatal. This has to do with the tumours metastasising and spreading throughout the body, usually indicating a bad treatment plan as it did not destroy all the primary tumour or the radiation caused tumour within the irradiated tissue (Sandru, et al., 2014). Finally, although rare, in the case that local and distant recurrences occurred, the fatality percent was almost 90% implying the worst-case scenario, second to death by primary tumour.

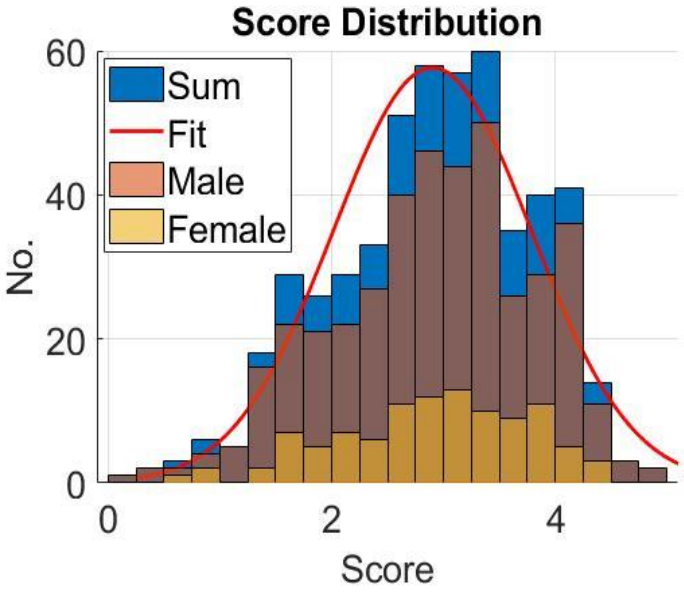

Figure 4.4 – Score Distribution Histogram

These reoccurrence statistics are an ideal indicator of treatment quality, instead of purely looking at patient death, or when the last follow up occurred. Neither of which would not allow for semi successful treatments to be scored adequately. Thereby we chose our weights to reflect the fatality chances giving each result a proportional score penalty. The last follow-up time (LFUT) was then used to ensure the best data was prioritised as well as applying a smoothing effect to the score distribution. The final weights for Equation (3.1 are described in Table 4.2, and the final score distribution can be seen in Figure 4.4.

Death Weight	Death Time	Recur Weight	Local Weight	Distant Weight	Both Weight	Recur Time	LFUT Weight
5	0.5	3	0.3	0.6	0.9	0.5	3

Table 4.2 – Table describing final score weights

4.2 Autoencoders

4.2.1 BEAM Encoding

As seen in Figure 4.5 the final training loss of our beam autoencoding after 1000 epochs was 3.15 (L1Loss) and 0.2 (MSE-Loss), with the validation loss being 10-20% higher at 3.99 (L1Loss) and 0.35 (MSE-Loss). We tracked both loss types simultaneously while training on L1Loss, this way we were able to better observe the training behaviour of the network as well as have a better idea as to when the network is acceptable. Training speed was acceptable with an epoch taking on average 0.3 seconds, and with live feedback in the form of the animated loss tracker, and console prints.

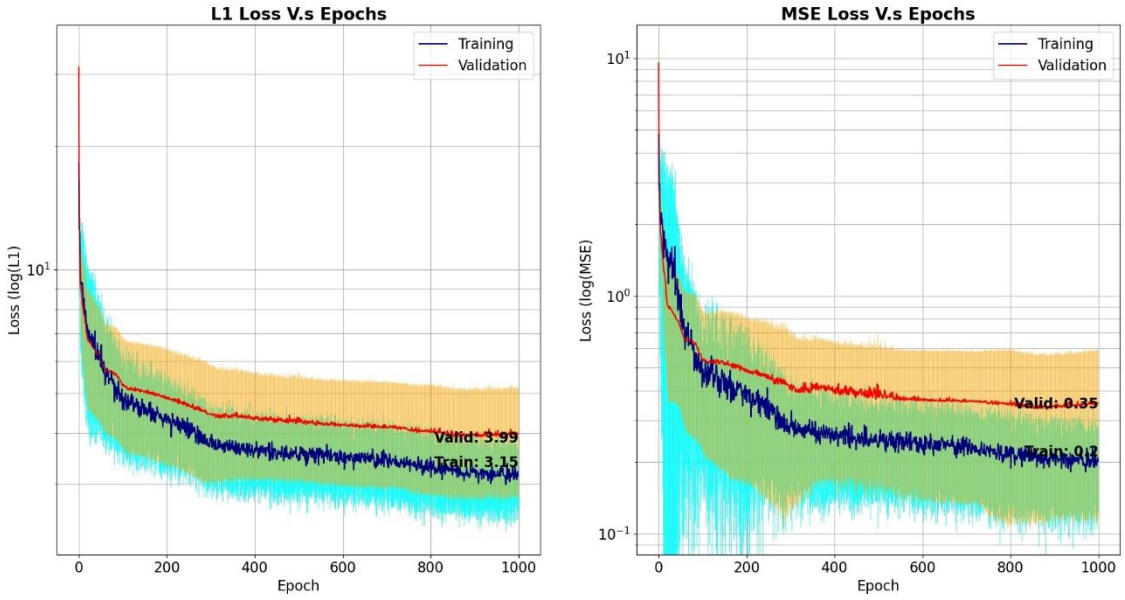

Figure 4.5 – Beam Training History
***Left:** L1 Loss Vs Epoch, **Right:** MSE Loss Vs Epoch*
With standard deviation shown for training in blue and validation in yellow.

The encoding quality of the beam autoencoder was not perfect, however it did not have to be as we would never intend to decode it in our functional use simply to use it as a compression model to feed into the prediction model. However, it did perform quite well, with Figure 4.6 showing a direct comparison of the decoding of some validation beam sets. We notice that in some cases, for example Plot 1, the peaks are smoothed away by the encoder, but still maintains the general shape, comparing that to plot 2 the data is almost perfectly decoded.

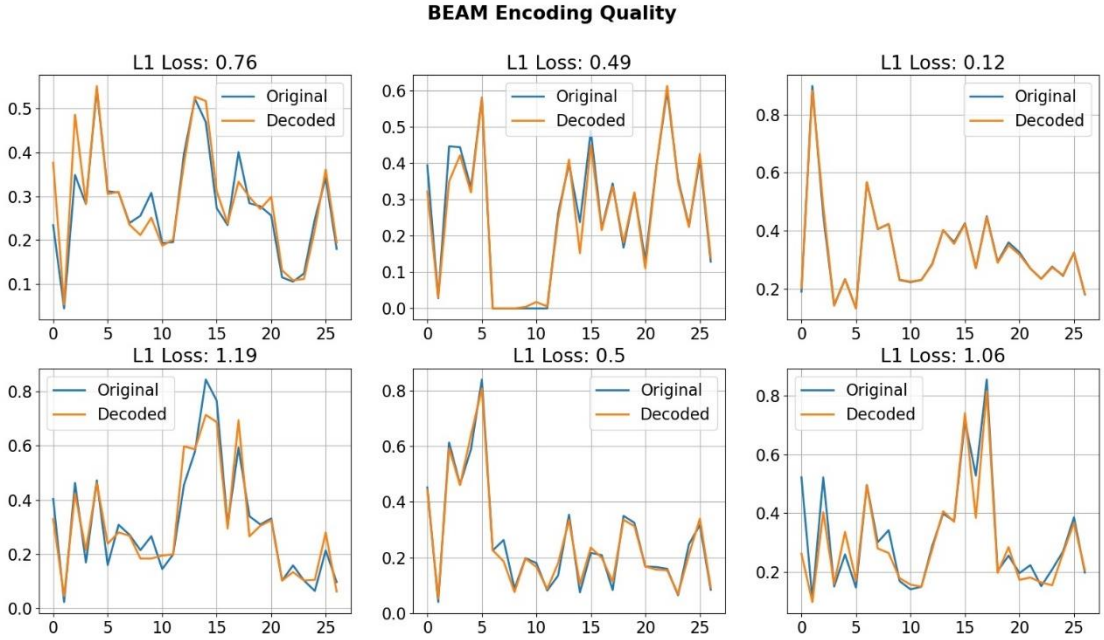

Figure 4.6 – Comparison of Original & Decoded Values for Beam Autoencoder

4.2.2 DVH Encoding

As seen in Figure 4.7 the final training loss of our DVH autoencoding after 400 epochs was 2.21 (L1Loss) and 0.37 (MSE-Loss), with the validation loss being approximately 5-10% lower at 2.08 (L1Loss) and 0.45 (MSE-Loss) which is unusual but can be explained. The reason why the validation loss ended up being lower than the training can likely be attributed to the fact that we were not able to stratify the DVH curves to evenly distribute the harder/easier cases, therefore the network was able to better predict the validation set, likely due to an easier distribution of data.

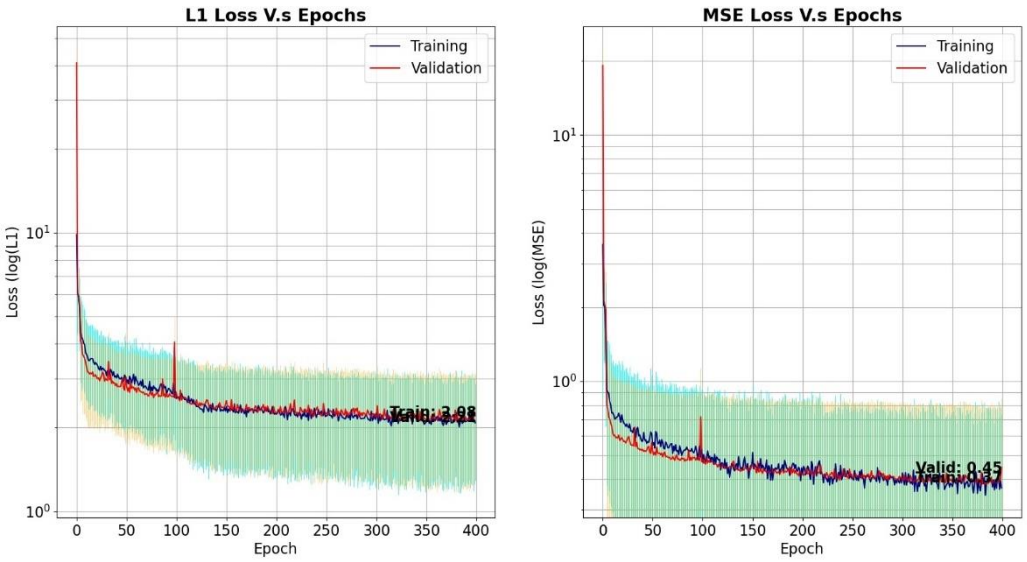

Figure 4.7- DVH model training history.
***Left:** L1 Loss Vs Epoch. R**ight:** MSE Loss Vs Epoch*
With standard deviation shown for training in blue and validation in yellow.

Training speed was slower than the Beam Autoencoder with an epoch taking on average 1 second, this is likely due to the dataset being 5x larger, since for each patient we had 5 DVH curves, (3 Target and 2 OAR). We also observed that the network converged to a general solution much faster with the standard loss deviations remaining similar throughout training. However, loss would still improve slowly over time, but was decided however not to overdo the training, to maintain a general solution.

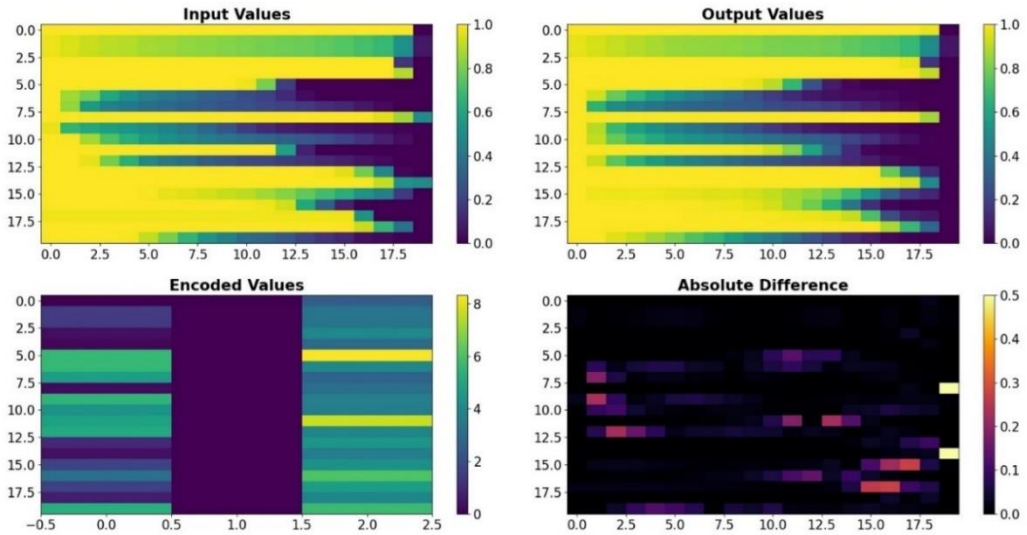

Figure 4.8 – Visual representation of DVH autoencoder model quality
DVH values are displayed horizontally, indicating individual feature magnitudes

The quality of decoding was visually impressive with only some specific edge cases giving the network trouble. As seen in Figure 4.8 the decoding had only minimal variations, and visually there is no perceivable difference between the encoded and decoded plot. The exception that the autoencoder is unable to reconstruct is an error that arose during DVH extraction, which is a small uptick in the last dose bin (an example can be seen in the vertical line 8), which the autoencoder removes upon reconstruction, fixing the extraction error, thereby we would claim that there are no true variations above ±0.2. Additionally, when we look at the encoded values, we can see hints to the underlying logic, how each variable influences the decoded version.

4.3 Network

4.3.1 Feature Statistics

Post autoencoding we investigated the statisical distribution of input features (Figure 4.9), via a boxplot anaysis of each individual feature. This would allow us to visualy identify bias in our training data, and possible cases where the network might run into trouble differentiationg different features or where exeptions could end up causing problems with a general solution.

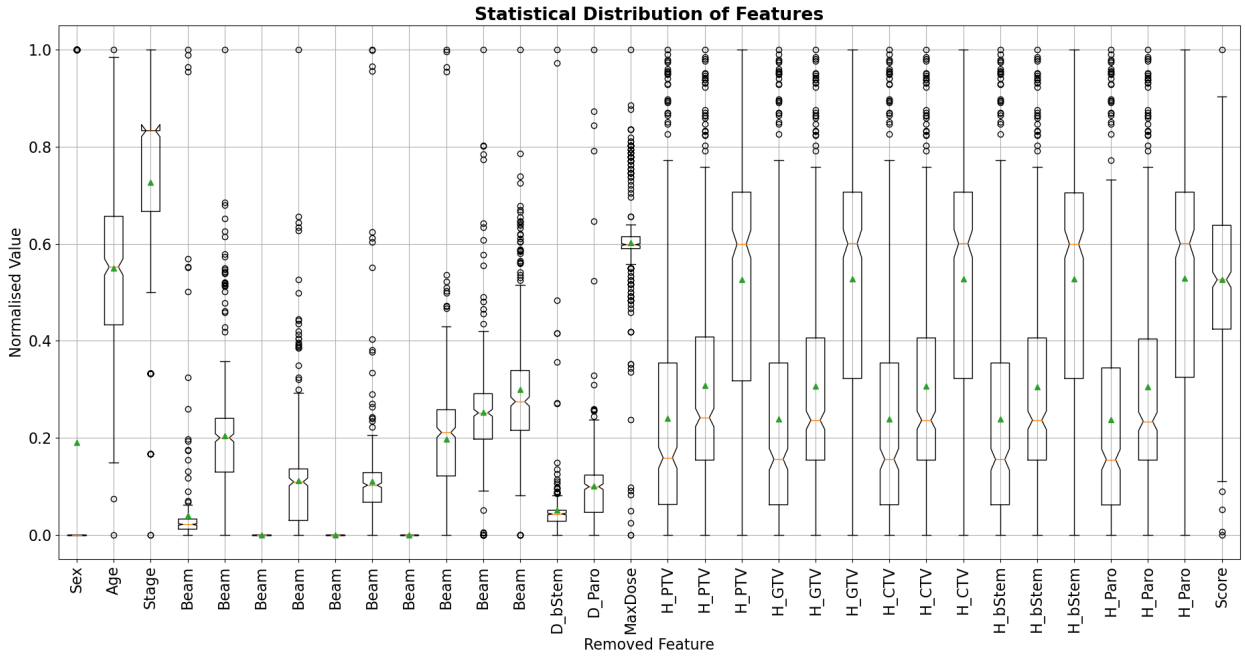

Figure 4.9 – Statistical distribution of network features, via boxplots. (Larger version available in appendix Figure 6.5) Green Triangles indicate mean values, red line/notch indicate median.

We quickly realised that our DVH encoding was reasonably successful, with few outliers, and with those that existed still being within a reasonable scope, implying that the DVH were similar but with significant differences. Our metadata had 2 issues, the first being sex, which as mentioned in 4.1.1 is biased, however as it is a binary choice it should be easy for the network to identify. The second issue is within the Distance to OAR values, which likely due to tumour placement had some values that were considerably outlying of the standard range. As a result, we took extra care to investigate the effect of each feature on the prediction after training, to make sure the network was not relying on a single feature too much, as well as no feature being irrelevant.

What is also obvious is the 0 clustering in 3 of our beam features, this is due to the beam autoencoder, while it is unknown why it chose to place all encoded values to 0 in these features, we made sure there was no underlying issue, and the model still decodes the information fine. (It might mean that we could have compressed the beam information even further, but this was not attempted. As it would require re-training/calibrating of the predictor network, and might not have resulted in as effective of an autoencoder)

Otherwise our plan data is varied enough to hint towards different approaches in the treatment planning, which is what we wanted to see and investigate. The variations should help our predictor network both predict the outcome of a treatment, as well as help push our later optimiser in the right direction when personalising future radiotherapy plans., by observing what choices worked well for specific metadata combinations.

4.3.2 Training Result

Figure 4.10 shows the training history of our predicitor network. The left plot displays the cross entropy loss over time, as well as the standard deviation, and the right dispays the accuracy of the score prediction. Right away we notice a significant problem, the valididation loss starts to increase very quickly around the 200th epoch. Initialy this may seem to indicate that the model is being overfit, however when we look at the predicition accuracy, the vaidation accuracy is increasing proportional to the training accuracy. This is because crossentropy loss does not nessarly say how accurate a model is at classifying, instead it acts as a metric to say how confident the networks prediction is. For example , the network may be unsure in its predictions, but still prediciting correctly. (There is an argument that a differnet loss function would have been better, however we were not able to find one that gave a comparable prediction accuracy). Additionaly we observe the networks predicton accuracy beginning at 45% which is the random guess probability mentioned in 3.4, showing how it begins in a random guessing state.

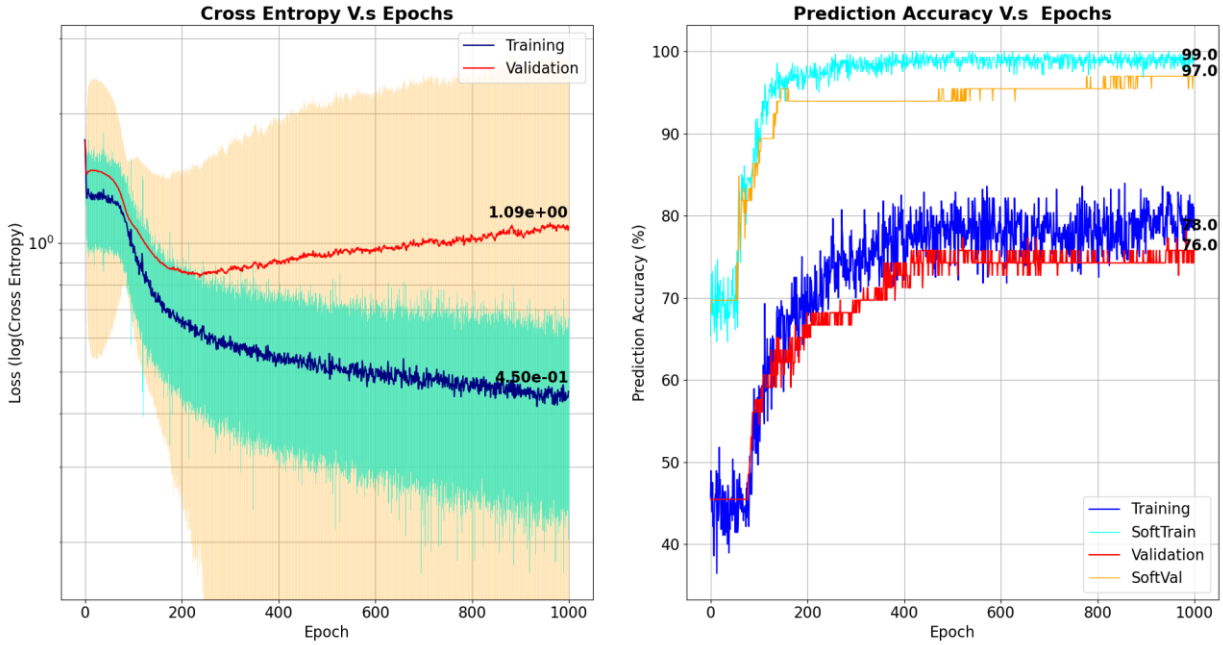

Figure 4.10 – Predictor Network Training History
***Left:** Cross Entropy Loss Vs Epoch, **Right:** Prediction Accuracy Vs Epoch*
With standard deviation shown for training set in blue and validation set in yellow.

In addition to the usual accuracy metric, we also decided to add a secondary test, the soft accuracy. As our score predictor is linear, and therefore the score bin classes are neighbouring, we decided to investigate the accuracy of the network, if the predicted score was within 1 class. (For example, the network predicted a score of 3, but the actual score was 4. This would count as an correct soft prediction) The reason behind this can be seen in Figure 4.10, we can watch as the network starts to understand the data, around epochs 80-120 with an increase in the soft accuracy, and then become more accurate in its predictions around epochs 150-200 as it learns the underlying nuances. After this point is where the validation loss starts to increase as the network becomes more and more sure of the training data (the reason why the training loss continues to steadily decrease), and less sure of the validation data (since it is not being reinforced).

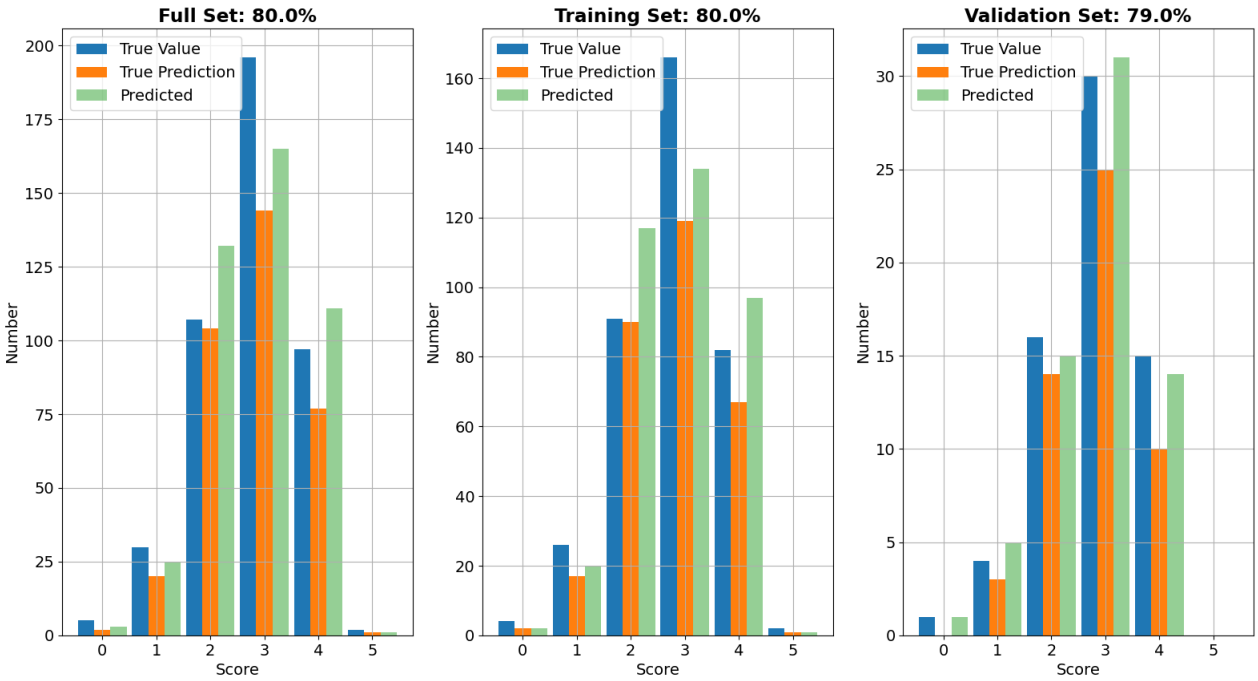

Figure 4.11 – Predictor model, per Class Accuracy
***Left:** Full Data Set, **Middle:** Training Data Set, **Right:** Validation Data Set*
True Value refers to the true class, True Prediction is a correct class prediction, Prediction is both true and false predictions.

Figure 4.11 shows the result of our network predictor at the point of highest accuracy. With a total accuracy of 80% in the full set and training set, and 79% in the validation set, the network does not seem over trained nor having memorised the training set. From the figure we can see the correct predictions (orange bar) showing that in the validation set the extreme cases are not as well identified as in the training. However, the distribution of predictions in general (green bar), including false predications is distributed normally, with the central scores being overpredicted (indicating cases where the network was not able to identify what impact the treatment had on the outcome). Overall, however the prediction seems stable and reasonable, with the identical score distribution indicating the stratification split having been successful.

There is an argument to be had as to where the model should be stopped/what epoch should be used as the final prediction model. As the accuracy of predictions never dropped it would be logical to take the most accurate epoch (For both Validation & Training). On the other hand, there is an argument that as soon as the validation loss starts to increase as at that point the model is starting to rely too heavily on the training set, which is typically considered bad practice, however it does mean that increased training accuracy is lost. To properly decide this further investigation was necessary, as well as an un-binning of the scores back into float values from their integer bins.

4.3.3 Structure Experimenting

For our network structure experiments, we disabled the random seed generation for the intial stratified splitting, therefore we are able to directly compare the results. We did three main experiments, firstly changing the size of the hidden layers, changing the number of k folds and changing the batch size.

Hidden Layer Size	Loss Training	Loss Validation	Total Accuracy	Training Accuracy	Validation Accuracy
Control/2	0.40 (88%)	0.95 (85%)	81 (101%)	82 (102%)	79 (100%)
Control Size	0.45 (100%)	1.12 (100%)	80 (100%)	80 (100%)	79 (100%)
Control*2	0.32 (71%)	2.16 (192%)	85 (106%)	86 (108%)	82 (104%)
Control*4	0.23 (51%)	3.23 (288%)	89 (111%)	90 (112%)	80 (101%)

Table 4.3 – Hidden layer shape experiments results

Looking at Table 4.3 we notice a clear trend, as we increase the size of the hidden layers, the training loss decreases, while the validation loss increases. Simultaniously the training accuracy improves while the validation accuracy stays constant. What is important to note, is that the values are taken from the epoch where the training and validation accuracy were both highest the network ever reached (the best checkpoint). We can infer therefore, what we are seeing is that with more layers the validation accuracy can maintain itself longer, and allow the network to continue training. However it does cause the network to loose confidence in its predictions as seen by the validation loss.

The raising of the number of k folds caused an increase in the time required for each epoch, but also caused the network to converge faster to a solution, likely as there was more data used to train per epoch. Thereby it also acted more stable in its loss values with less fluctuations per epoch. However there was no significant change in the accuracy or performace of the network. Simmilarly we can see the effect of changing the size of the training batches in Table 4.4, were again no true impact was made on the quanifiable model results, but it sped up the training by reducing the time needed for each epoch. It did tend to cause some instability in the higher number though).

Size of Batches	Batch Size	Epoch Time
Control/2	3	4.8 (160%)
Control Size	5	3.0 (100%)
Control*2	10	1.9 (63%)
Control*3	15	1.4 (47%)

Table 4.4 - Batch size experiments results

4.3.4 Integer Conversion

The logic behind the class to integer conversion was based on the underlying theory of crossentropy loss. Crossentropy loss was intialy used to avoid the network simpily returning the mean value for each prediction (As MSE-Loss does not work well with only a single output), however it needs to be converted to a single integer value if it is to be used in treatment planning optimization. Therefore we developed a statistical method to convert a set of probabilities into a single intiger. The solution was to give each class a weight, the remove any class that had a negative probability and use a weighted sum average to extract the joined probaility prediction.

$$\text{Integer Score} = \frac{1}{\mathrm{P_n}\ (\mathrm{P_n} > 0)} \sum \mathrm{P_n S_n}\ (\mathrm{P_n} > 0) \qquad (4.1)$$

Integer Conversion Function
P_n: Probability of Score Class, S_n: Value of Score Class

In Figure 4.12 we can see a direct compariosn of raw and predictied scores. As we already saw in Figure 4.11 the extreeme scores are hard for the network to predict (and there is still not perfect prediction of the validation set), however other than a few highlights one only sees minimal differences between the raw and predicited scores. Using the absolute score difference we can see where the issues occur, and extreemly few deviate more than 1 (Which can point to cases where the treatment although personalised was not enough to save the patient, something that would be better understood if we had a larger data set). Overall however the conversion method is functional and will be used for the later optimiser integration.

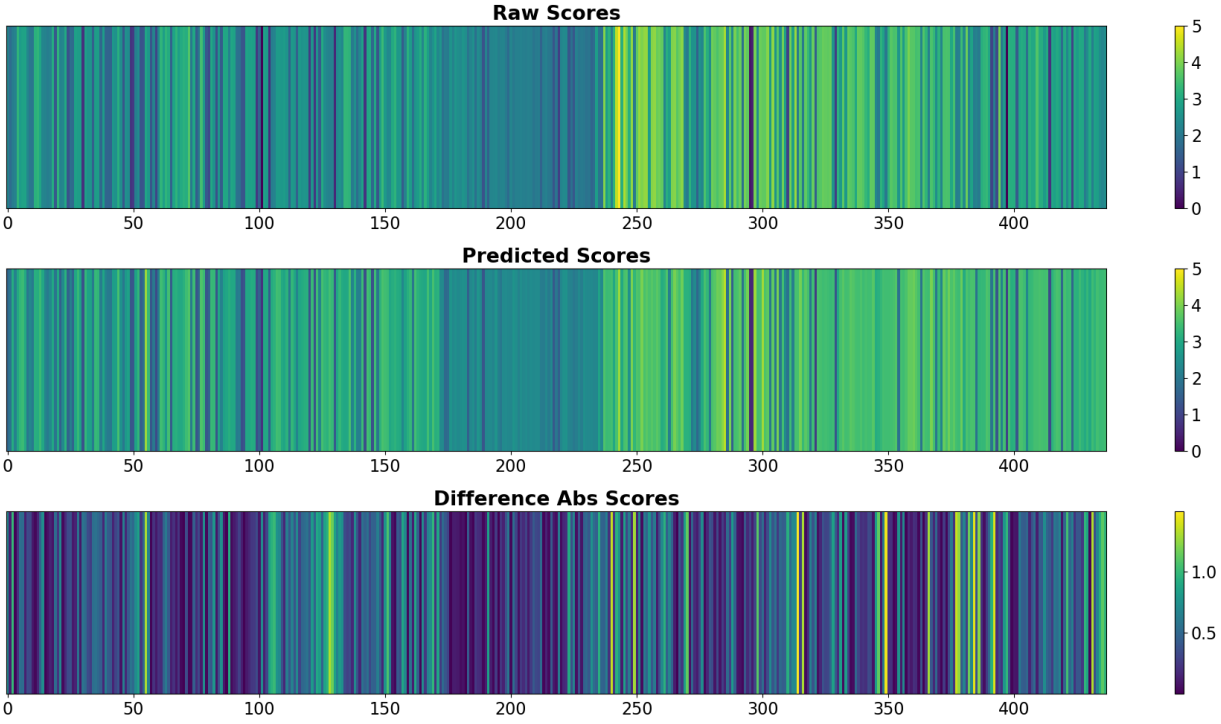

Figure 4.12 – Visual Indicator of score prediction, after integer conversion

4.3.5 Feature Impact

To investigate the impact of individual features on the score prediction we ran an experiement, sequentialy removing each feature by setting its value to -1. We then observed what kind of an impact it would have on score accuracy and on the mean score value. This would enable us to both investigate which metadata information the network utalises in its prediction, as well as what information is most critical for prediction/optimisation.

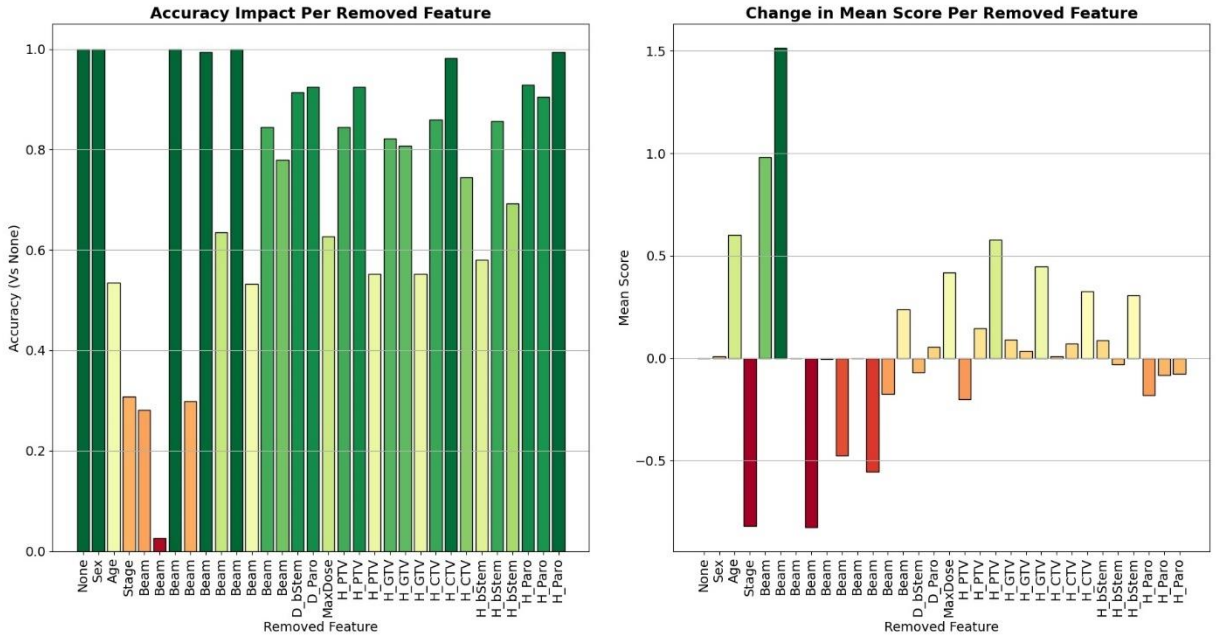

Figure 4.13 – Feature Impact comparison
***Left**: Accuracy Percent, **Right**: Score change*

As we can see in Figure 4.13 the metadata most influential in the score prediction is the distance to brainstem, tumour stage and the age of the patient. The least influential feature was the sex of the patient; however, this is expected as treatment plans are generally not personalised to that feature. Additionally, the distance to parotid had relatively low impact compared to brainstem, around the same as the age of patient.

As to planning parameters, most are around the same range in terms of influence, only having an impact between 20-40%. The beam information was most influential, (not including the 0 values) one feature even having an impact of almost 65%, and others around 50%. DVH information seemed to have various levels of impact, likely depending on the encoding which section of the DVH was more important for each ROI. Especially the brainstem got special attention with an impact of almost 70%, which supports the high influence of the brainstem distance metadata metric.

4.4 Plan Optimiser

4.4.1 Performance Analysis

Running the optimiser with score assistance has significant performance impacts, the iteration times increase on average 100x, to about 10 seconds each. This is due to the inefficient nature of our gradient function, as well as the need to convert our beam information and generate DVH for the dose cubes each time the optimisation function. Additionally, as our gradient function can only sometimes obtain an optimal score gradient the IPOPT optimiser sometimes begins to find the next step using finite differences, this can cause the gradient function to be called 5-10 times before the next step is taken. However as this is a proof of concept integration and not intended for clinical or consumer use, the performance hindrances are acceptable.

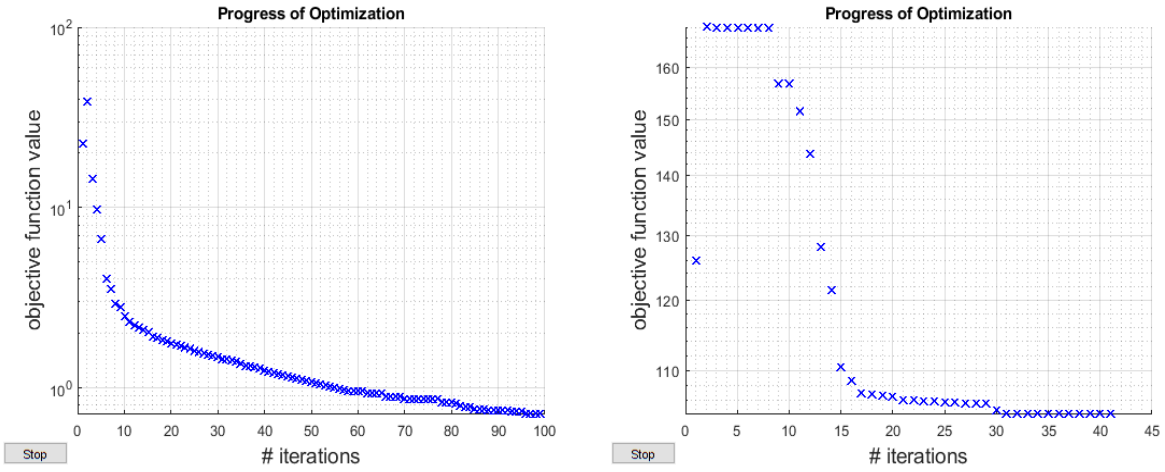

*Figure 4.14 – **Left:** Normal Optimization Progress & **Right:** Score Assisted Optimization Progress*

For the purposes of testing, we have limited our number of iterations to a maximum of 50, which results in a total run time of 8-10min, or if the optimiser is no longer able to reduce the loss further the manual stop function is triggered and the result is returned. To ensure that the score is adequately considered in the optimisation we implemented an exponential weight to the score loss, thereby supressing gradients that would improve the inbuilt loss but cause the score loss to increase. This causes some issues with the gradient function, and the optimiser repeats gradient calculations until it finds a solution which reduces the overall loss, which is then displayed on the optimization progress plot.

4.4.2 Case Studies

The case study will be examining two different optimisations for individual patients. One will have only the basic target prescriptions, the other will include the OAR prescriptions each time comparing the inbuilt fluence optimiser to a score assisted one. We do this to see the score optimiser in action where its acting as the only cost function compared to an assistance function to the existing cost functions. Our prescriptions will be following standard head and neck procedure as described in (Deasy, et al., 2010) & (London Cancer, 2014) , however we will not be creating new ROI and only using patients that exist within our data set. This avoids us making assumptions, as well as avoiding the need for further data acquisition. Note: We did multiple case studies for 5-6 patients, however for the report we decided to highlight two cases where the effect was easy to identify.

Patient 1: HN-CHUS-044

The first patient we investigated was HN-CHUS-044, a 70-year-old male with a stage III tumour in the Larynx, who historically ended up having a local recurrence after 1053 days, but did not result in the death of the patient with the last follow up occurring on day 1760. This treatment ended up with a final score of 2.55, reflecting the recurrence as well as the median last follow up time. This is an average patient, with a middle stage tumour and a high age near the average of 65.

The first comparison involved running the IPOPT optimisation having only specified dose prescriptions for the Target ROI, defining them as: squared deviation with a dose Reference of 70, 65 and 60Gy for the GTV, CTV and PTV respectively. This means that the optimiser will not account for the OAR and will only focus on achieving the target doses. The result of running the optimiser with only target prescriptions (Figure 6.1), the model seems to encourage maximising the dose on the targets, even going so far as to allow more exposure on the parotids. However, it does not go out of bounds and still returns a reasonable output. There is also the argument that this is a Type II estimation error (As described in Moore, 2019 and Section 2.1.1) as there likely exists a better Parotid OAR sparing DVH, however it might also be based on the internal priorities of the model regarding ROI/OAR. Therefore, there is still significant room for improvement in the network-based optimisation, and with more training data as well as an improved optimiser integration we could see the network becoming more applicable on difficult cases.

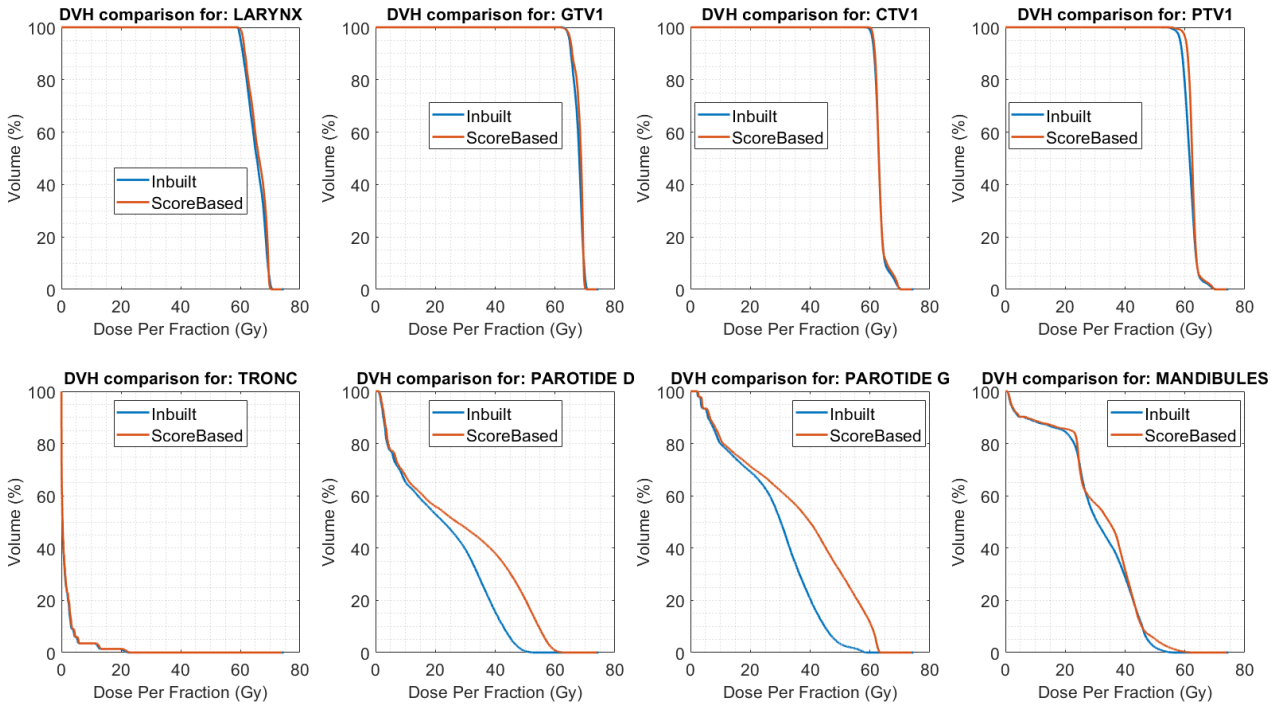

Figure 4.15 - HN-CHUS-044, DVH histograms comparing OAR Prescription (With/without score assistance)

When we apply the OAR prescriptions: squared overdosing with a dose reference at 40 and 25Gy for the brainstem and parotid, respectively. We see the score assisted optimiser (Figure 4.15) still insisting on the maximum dose for targets, fighting the inbuilt optimiser in its attempt to lower both the target and OAR dose. It is important to note here that the score model is not designed to return the best theoretical plan, but to adjust it according to the patient needs. Giving the final plan after optimisation a score of 2.925, which is a significant improvement from the historical score of 2.55.

Looking into the radiotherapy metrics for quantifying a good treatment plan we have extracted some values, highlighting the V-69 values (Table 4.5) and the D-95 values (Table 4.6). There are additional metrics tabulated in the appendix under Table 6.1 & Table 6.2, however the former best illustrate the effect of the score assistance, for this patient's treatment planning solution.

ROI	GTV	CTV	PTV	OAR
Target Only	28.97	2.14	0.91	0
Target Only + Score Assisted	44.11	3.22	1.37	0
OAR Prescriptions	23.68	1.74	0.74	0
OAR Prescriptions + Score Assisted	37.53	2.74	1.17	0

Table 4.5 – V-69 Percent (%), Value comparison for HN-CHUS-044

In the V-69 values we can observe how the score assisted plans prioritise a high target dose, and how the model attempts to work within the limits applied by the OAR prescriptions, lowering the total target dose to spare the OAR. This is obvious when we look at the D95 values, the heat map giving us a good visual indication of how the OAR prescription with score assistance is the best plan for target dose coverage while also significantly sparing the OAR compared to the Target only prescription models. Additionally, as the historical data shows, the patient had a local recurrence which is often the result of an underdosed target area, and thereby the model attempts to keep dose as high as possible.

ROI	GTV	CTV	PTV	Brain Stem	Parotid Right	Parotid Left	Mandible
Target Only	64.842	61.002	58.74	0	2.184	3.963	1.95
Target Only + Score Assisted	65.079	61.386	60.48	0	2.22	3.933	1.992
OAR Prescriptions	64.842	60.891	58.389	0	2.004	3.546	1.839
OAR Prescriptions + Score Assisted	65.16	61.215	60.318	0	2.202	3.918	1.98

Table 4.6 – D-95 Dose (Gy), Value comparison for HN-CHUS-044

Patient 2: HNSCC-01-0083

The second patient we investigated was HNSCC-01-0083, a 34-year-old male with a stage IVb tumour in the Oropharynx, which was one of the best treatments in the metadata set, having a full recovery and no recurrences, with a final score of 4.3. The models score being pulled down by the comparatively short follow up time of 2500 days. The difference to our first patient is the younger age and higher tumour stage, in a way being the opposite patient metadata state.

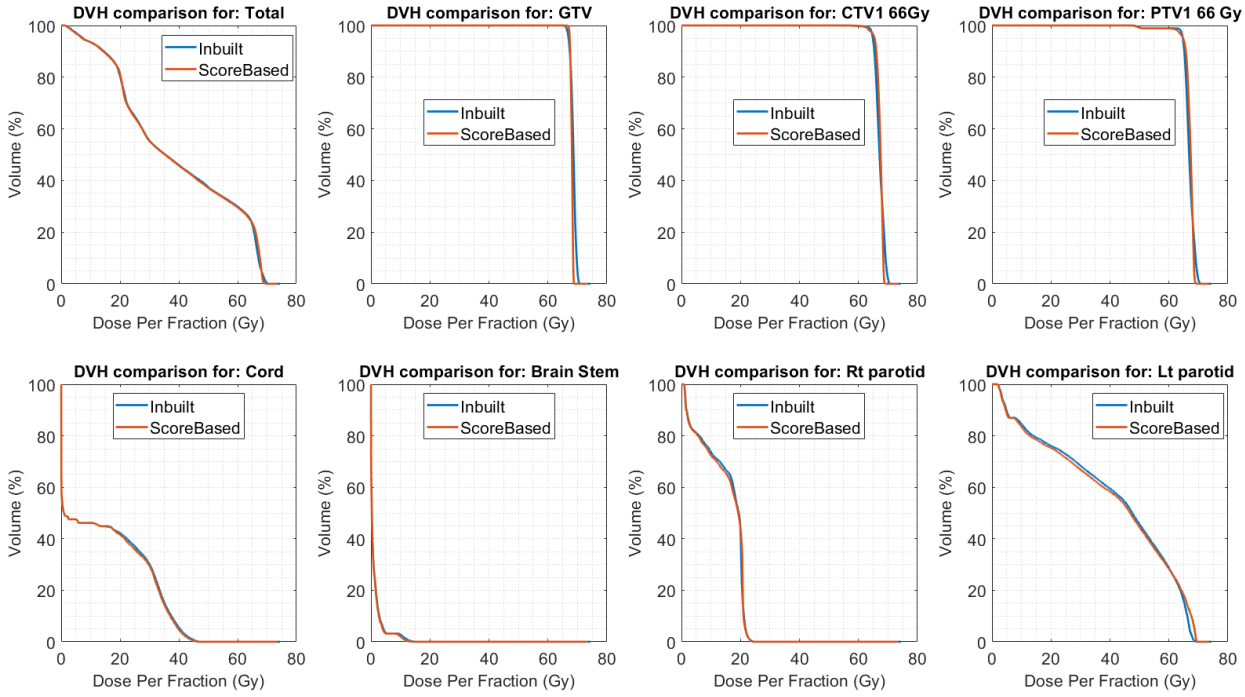

Figure 4.17 – HNSCC-01-0083, DVH histograms comparing Target Directed (With/without score assistance)

In the DVH figures we can see the impact of that metadata steering, firstly we notice that there is very little impact by the optimiser to the standard plan, as expected of a standard plan that had a high score rating. It did however make some light adjustments, specifically sparing the OAR where possible, while trying to maintain the target doses. This is more obvious in Figure 4.17 than Figure 6.2, we observe the parotid being significantly spared, it is of course important to remember that the model is not supposed to make sweeping changes to the treatment plans, and simply adjust/regularise them a little to better fit the patient's needs. Therefore, even a light change can be considered important. (Note: we see no effect on the spinal cord dose, since the network is not designed to optimise for that ROI).

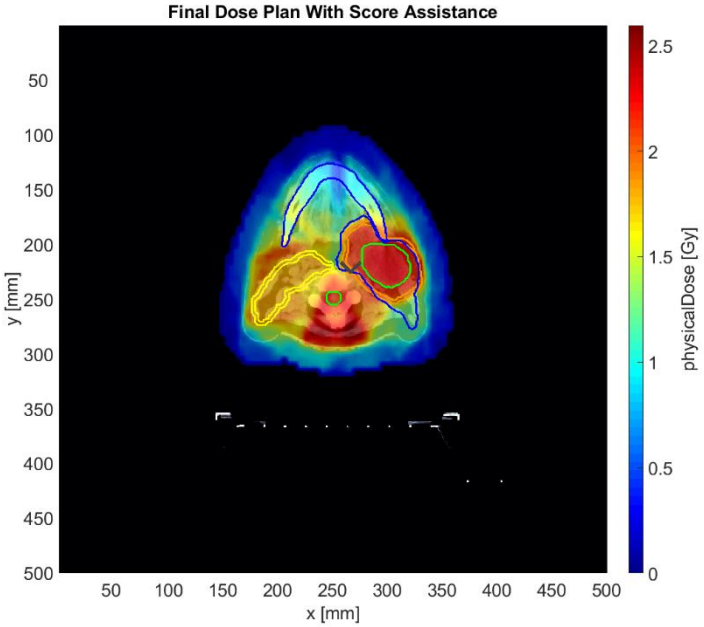

Figure 4.16 – Dose map for, HNSCC-01-0083 treatment plan with score assisted optimisation.

ROI	Targets	Spinal Cord	Brain Stem	Parotid Right	Parotid Left
Target Only	100	44.87	0.0375	66.83	79.3
Target Only + Score Assisted	100	44.87	0	65.62	78.48
OAR Prescriptions	100	44.87	0.0188	66.68	78.31
OAR Prescriptions + Score Assisted	100	44.87	0	65.92	78.66

Table 4.7– V-30 Percent (%), Value comparison for HNSCC-01-0083

When we look into the RT metrics, we see a similar pattern, in the V-0.5 values Table 4.7, the score assisted plans are always the ones minimising the dose to OAR, even the OAR prescription did not reduce it as strongly as the assisted version. The D-50 metric comparison Table 4.8 shows a similar pattern, although here we see that the plan with only target prescriptions + score assistance is theoretically the best option, with the highest target dose, as well as the lowest OAR doses (With the exception of the GTV dose, which is the lowest of all options, thereby making it perhaps no the most ideal option to go for). However, it solidifies the fact that the score assistance is prioritising sparing the OAR tissue, while attempting to maintain target dose, but at a lower priority. (There are additional metric comparisons in the appendix under: Table 6.3 & Table 6.4)

ROI	GTV	CTV	PTV	Spinal Cord	Brain Stem	Parotid Right	Parotid Left
Target Only	68.676	67.134	66.825	0.996	0.306	19.425	47.562
Target Only + Score Assisted	68.124	67.512	67.368	0.987	0.288	19.248	46.827
OAR Prescriptions	69.015	67.128	66.792	1.002	0.3	19.506	38.121
OAR Prescriptions + Score Assisted	68.244	66.957	66.684	0.981	0.294	19.413	46.623

Table 4.8 – D-50 Dose (Gy), Value comparison for HNSCC-01-0083

4.4.3 Score Assistance Summary

The network integration into the optimiser as a regularisation function has shown great promise. From our highlighted case studies, we have shown that it makes an impact on the treatment plans depending on the patient's metadata and making treatment choices depending on the outcomes of the trained treatment plans.

On average we saw a score improvement of 10-20% for each treatment plan we optimised for, some having more subtle and others more obvious changes from the original optimiser solution. In some cases, we even had significant improvements vs the historical score values, implying that the generated score assisted plan would have resulted in a significantly improved treatment outcome.

For example, a patient that historically had a score of 1.3 due to an early distant recurrence. After optimisation, the score assisted treatment plan had a final predicted score of 2.9, which would imply a short last follow up time or a low likely hood of a local recurrence. It is of course important to note that this model is predicting according to data sets, and there is a biological randomness to consider in the treatment outcome. This is supported by the fact that no matter how long we let the training run we can never get 100% accuracy in validation set, score prediction.

The way the regularisation behaves according to metadata information reflects what we found during our investigation in section 4.3.5, where we observed how different features affect the score value, and in turn what would be optimised for. This was confirmed in the example case studies, where we see that a lower age seems to correspond to more care being taken to spare the OAR, or how target ROI are prioritised when recurrence is a risk. Thereby subtly acting as a regularisation parameter and personalising the treatment plans according to the patient's metadata information.

4.4.4 Future Statistical Analysis

Due to time constraints on this thesis project we were unable to run all the statistical tests we intended. This is partially due to each optimisation using score assistance can take a significant amount of time, as well as the need to manually specify the dose prescriptions for patients target ROI and OAR (Something that should be ideally done by an expert in the field). However, the underlying methodology has been decided on and will be described here.

We would intend to take a selected set of 40-50 patients whose CT and ROI information is known to be stable and does not contain underlying issues. These would then be split into groups depending on the major influencing metadata features (Age, stage, and brain stem distance) as well as according to their historical predicted score. This split will later be used to categorise the generated plans and compare how different metadata information influences the model's decisions. (As while individual case studies can be illuminating, they do not hold statistical weight).

We would then run our plan optimiser, for each of our selected patients. With and without OAR prescriptions and with/without the score assistance model. Automatically collecting the different RT metric data (D-95, D-50, V-60, V-30) and generating mean and standard deviation values for each patient group as well as value lists. This would allow us to directly analyse how the model decided to handle each patient group, both in terms of single feature metrics as well as combinations. (for example, young to old. But also, young + high stage vs. old + low stage). We would also be able to run t-test statistical analysis to ensure there is significant difference (null hypothesis) between the different planning decisions, or if the changes are statistically insignificant.

The implementation and execution of this statically analysis method is very much possible for the project in its current state, and would be a priority in the case of project continuation, as it would be the ideal way to confirm whether or not the model and the MatRad implementation is successful.

5 CONCLUSION

The personalisation of treatments has become a primary aim of medicine in the last decade, with more and more methods being developed to help design treatments in a way that it targets the individual patient needs and situation. This trend has also been seen in radiotherapy, were lack of personalisation can be deadly, or cause severe long-term ailments. Additionally, the increase in the accessibility of deep learning technologies and its ability to identify subtle patterns in data structures, and make decisions based on what it learned, provides an excellent opportunity to create solutions that assist in personalisation, based on proven radiotherapy treatment methods.

In our thesis our primary aim was to develop an innovative method involving the use of deep learning as a tool to primarily predict the outcome of RT plans, by using publicly accessible metadata and DICOM information. This involved the processing of our acquired data, via classification and standardisation algorithms, into a form that is readable for a ML network. We then auto-encode the larger data sets, the beam plan, and the dose volume histogram information to avoid overwhelming the network, finally feeding our encoded plan data and extracted metadata into our predictor network and experimenting with different training methods and network structures.

Our extraction and classification methods were expanded and improved upon from the preceding lab report and resulted in working data sets for 85% of acquired patients, with an acceptable distribution of the key features. Via standardisation we were able to fill in remaining gaps where possible and ensure data stability when later feeding into the network. Additionally, using the extracted treatment outcome data, we were able to develop a plan scoring method, using the underlying data relationships. We also contacted practising physicians to get their professional input and feedback. Thereby allowing us to train our final model not only on the underlying patient metadata but also to consider the quality of the treatments, with the intention of finding relationships between successful approaches and patient characteristics.

At this point we had obtained a large set of trainable features for each patient, it was decided however to compress certain feature sets to avoid training on non-critical information, as well as improve training performance by reducing redundant complexity. Therefore, we auto-encoded our beam plan information by 63% and our DVH information by 85%. Overall allowing us to reduce our predictor network inputs by 77% (as described in Table 3.1) and in turn allowing us to train our network with much higher efficiency and knowing that all the key information of our beam and DVH data is retained and still influences the prediction.

The final predictor network was a classification cross entropy loss model, that would place the treatments into a score bin according to the quality of the treatment. Training involved splitting the data using a stratification method to ensure our training and validation set were equal in difficulty and distribution and thereby avoiding bias. With the final network shape (Figure 3.7) being 5 hidden layers deep, along with a dropout rate of 60% between each layer to avoid overfitting.

During training we implemented a k-fold training method, thereby avoiding the network from memorising the data as well as simulating a larger dataset. It also enabled us to measure the training loss more robustly, with the final training parameters being: 1×10^{-5} and 1×10^{-8} for the learning and decay rates. After successive training of the predictor network using various settings, we acquired a final trained version with a training/validation accuracy of 80% and 79%, and a loss of 0.45 and 1.12, respectively.

There were versions of the model that had a higher accuracy however the underlying logic was less stable, for example if the size of each hidden layer was multiplied by 4, we would have a training/validation accuracy of 90% and 80%, and a loss of 0.23 and 3.23 respectively, which implies a very unsure prediction of the validation scores. This was followed by an in-depth analysis of how the various features impacted the model's outcome, and the creation of a method to convert our classification output back into a single integer for true comparison.

At this point the primary goal of our thesis had been completed, we had created a neural network, able to predict the outcome of radiotherapy treatment plans by using the patient metadata in tandem with the beam planning information. The accuracy of the model was high enough to impart significant confidence as a quality assessment metric, as well as providing feedback to physicians and patients regarding the probabilities of treatment outcome. Some improvement could be made by introducing different metadata information or including other DVH ROI information, however to do so the data must first be acquired from completed treatments, and must be present in an overwhelming majority of patient data sets or will require significant data augmentation, to ensure training stability.

Our secondary aim for our thesis was to integrate our trained predictor network into a RT plan optimisation software. Thereby allowing us to generate treatment plans for patients, while using the predictor network as a regularisation function. Slightly altering the resulting plans to better fit the patient's needs, while maintaining the biological prescriptions as set by the physician. In the ideal case personalising the treatment plan in a way that the dose is clinically acceptable and the beam physically applicable.

By modifying an existing RT plan optimisation function, we were able to incorporate the prediction network as a feedback method, and thereby automatically optimise radiotherapy plans for patient metadata. This was achieved via the integration of our Pytorch, python model into the MATLAB based radiotherapy treatment planning software MatRad. This involved the modification of our network to accept raw treatment information and integrating the autoencoders into the score call-back function. The MatRad optimisation files were equally modified, as metadata would now need to be fed into the optimiser and the optimiser function would need to pre-process the weights into radiotherapy data. Additionally, we integrated a gradient acquisition function to ensure that where possible the optimiser would be able to move towards reducing the score loss.

We decided to go with a hybrid optimisation approach, maintaining the inbuilt methods and making the physicians dose prescriptions as the core parameters. Then using the prediction model as a regularisation function, which would slightly alter the optimisation (by minimising for score loss) and thereby personalise the plan according to what is the predicted optimum. This way the resulting treatment plans were biologically acceptable and would not overstep any dose bounds or explode planning values. However the underlying method of integration was rough and inefficient causing the optimisation to stagger and get stuck occasionally, as well as take an unreasonable amount of time, if we wished to run larger statistical analysis of the network assisted optimisers performance.

As seen in our case studies the network acts logically on the treatment plans, and changes its approach depending on the patient's metadata information. Thereby personalising the treatment planning approach with the help of our prediction network, acting as a final regularisation function within MatRad. Potential improvements include allowing for easier UI control of the score regularisation as it is currently inaccessible to the user, and general performance improvements (The majority of performance issues are a result of weight conversions and needing to call the python function multiple times per iteration).

6 APPENDIX

A. References

1. Shun'ichi A., 1993. Backpropagation And Stochastic Gradient Descent Method. *Neurocomputing*., Pp.185 – 196.
2. Adam, Kingma D. And Ba J. 2014. A Method For Stochastic Optimization. *International Conference On Learning Representations*.
3. B., Giuseppe. 2017. Machine Learning Algorithms: A Reference Guide To Popular Algorithms For Data Science And Machine Learning. *Packt Publishing*.
4. Bangert M., Wieser Hp., Cisternas E., Klinge T., Jäkel O., Mairani A., Gabrys H., Ziegenhein P. 2015. *Matrad An Open Source Multi-Modality Radiation Treatment Planning System*. Heidelberg: German Cancer Research Centre.
5. Barrett S, Hanna Gg, Marignol L. 2018. An Overview On Personalisation Of Radiotherapy Prescriptions In Locally Advanced Non-Small Cell Lung Cancer: Are We There Yet? *Radiother Oncol.* 128(3), Pp.520-533.
6. Benjamin P. Ziemer, Satomi Shiraishi, Jona A. Hattangadi-Gluth, Parag Sanghvi. 2017. Fully Automated, Comprehensive Knowledge-Based Planning For Stereotactic Radiosurgery: Preclinical Validation Through Blinded Physician Review. *Practical Radiation Oncology*. 7, Pp.269-278.
7. Bidgood, W. D., Jr, Horii, S. C., Prior, F. W., & Van Syckle, D. E. 1997. Understanding And Using Dicom, The Data Interchange Standard For Biomedical Imaging. *Journal Of The American Medical Informatics Association : Jamia*. 4, Pp.199–212.
8. Francois C., 2016. *Building Autoencoders In Keras*.. Available From: https://blog.keras.io/building-autoencoders-in-keras.html [Accessed 10 Sep 2020]
9. Cilla, S., Ianiro, A., Romano, C. Et Al. 2020. Template-Based Automation Of Treatment Planning In Advanced Radiotherapy: A Comprehensive Dosimetric And Clinical Evaluation.Sci Rep. *Sci Rep*., P.423.
10. Deasy, J. O., Moiseenko, V., Marks, L., Chao, K. S., Nam, J., & Eisbruch, A. 2010. Radiotherapy Dose-Volume Effects On Salivary Gland Function.. *Radiotherapy Dose-Volume Effects On Salivary Gland Function. International Journal Of Radiation Oncology, Biology, Physics,*. 76, Pp. S58–S63.
11. Gillies S., Et Al. 2007. *Shapely: Manipulation And Analysis Of Geometric Objects*. [Online].
12. Grossberg A, Mohamed A, Elhalawani H, Et Al. 2017. Data From Head And Neck Cancer Ct Atlas. *The Cancer Imaging Archive*.
13. Grossberg A, Mohamed A, Elhalawani H, Et Al. 2018. Imaging And Clinical Data Archive For Head And Neck Squamous Cell Carcinoma Patients Treated With Radiotherapy. *Scientific Data 5*., P.180173.
14. Hartley J., Et. Al. 2017. *Colorama: Simple Cross-Platform Colored Terminal Text In Python*. [Online]. Available From: https://github.com/tartley/colorama, [Accessed 04 Apr 2020].
15. Hendrycks D., Gimpel K. 2016. Gaussian Error Linear Units (Gelus).
16. Hunter, J.D. 2007. Matplotlib: A 2d Graphics Environment. *Computing In Science & Engineering*. 9, Pp.90-95.

17. Kreuzer, M., Boffetta, P., Whitley, E. Et Al. 2000. Gender Differences In Lung Cancer Risk By Smoking: A Multicentre Case–Control Study In Germany And Italy. *British Journal Of Cancer 82*., Pp.227-233.
18. Forker L.J., Choudhury A., Kiltie A.E.. 2015. Biomarkers Of Tumour Radiosensitivity And Predicting Benefit From Radiotherapy. *Clinical Oncology*. 27(10), Pp.561-569.
19. Liu, H. H. Et Al. 2004. Feasibility Of Sparing Lung And Other Thoracic Structures With Intensity-Modulated Radiotherapy For Non-Small-Cell Lung Cancer. *Int J Radiat Oncol Biol Phys 58*., Pp.1268–1279.
20. London Cancer. 2014. *Head And Neck*. London.
21. Ma, J., Setton, J., Morris, L., Albornoz, P. B., Barker, C., Lok, B. H., Sherman, E., Katabi, N., Beal, K., Ganly, I., Powell, S. N., Lee, N., Chan, T. A., & Riaz, N. 2017. Genomic Analysis Of Exceptional Responders To Radiotherapy Reveals Somatic Mutations In Atm. *Oncotarget*. 8(6), Pp.10312–10323.
22. Martin K., Schroeder W., Lorensen B. 2008. *Vtk*. [Online]. [Accessed 04 Apr 2020]. Available From World Wide Web: <Https://Vtk.Org/>
23. Mason, D. L., Et Al. 2014. *Pydicom: An Open Source Dicom Library*. [Online]. Available From: https://github.com/pydicom/pydicom, [Accessed 04 Apr 2020].
24. Mckinney, W. 2010. Data Structures For Statistical Computing In Python. *Proceedings Of The 9th Python In Science Conference*. 1, Pp.51-56.
25. Men, K., Geng, H., Cheng, C., Zhong, H., Huang, M., Fan, Y., Plastaras, J. P., Lin, A., & Xiao, Y. 2019. More Accurate And Efficient Segmentation Of Organs-At-Risk In Radiotherapy With Convolutional Neural Networks Cascades. *Medical Physics*. 46, Pp.286-292.
26. Monro, Herbert And Sutton. 1951. A Stochastic Approximation Method. *Annual Math Statist*., Pp.400-407.
27. Moore, Kevin L. 2019. Automated Radiotherapy Treatment Planning. *Seminars In Radiation Oncolcology*. 29, Pp.209-218.
28. Mustra, Mario, Kresimir Delac, And Mislav Grgic. 2008. Overview Of The Dicom Standard. *Elmar*. 50th , Pp.39–44.
29. Burnet N.G., Thomas S.J.,Burton K.E. And Jefferies S.J. 2004. Defining The Tumour And Target Volumes For Radiotherapy. *Cancer Imaging*., Pp.153–161.
30. Nguyen, Dan & Long, Troy & Jia, Xun & Lu, Weiguo & Gu, Xuejun & Iqbal, Zohaib & Jiang, Steve. 2019. A Feasibility Study For Predicting Optimal Radiation Therapy Dose Distributions Of Prostate Cancer Patients From Patient Anatomy Using Deep Learning. *Scientific Reports*. 9, P.1076.
31. Nguyen B.T., Hornby C.J., Et Al. 2012. Optimising The Dosimetric Quality And Efficiency Of Post-Prostatectomy Radiotherapy: A Planning Study Comparing The Performance Of Volumetric-Modulated Arc Therapy (Vmat) With An Optimised Seven-Field Intensity-Modulated Radiotherapy (Imrt) Technique. *Journal Of Medical Imaging And Radiation Oncology 56*., Pp.211-219.
32. Oliphant, Travis E. 2006. *A Guide To Numpy*. New York: Usa: Trelgol Publishing.
33. Oliphant., Travis E. 2007. Python For Scientific Computing. *Computing In Science & Engineering*. 9, Pp.10-20.

34. P Mayles, A Nahum, J.C Rosenwald. 2007. *Handbook Of Radiotherapy Physics - Theory And Practice*. Oxford: Taylor & Francis.

35. Paszke, Adam And Gross, Sam And Chintala, Soumith And Chanan, Gregory And Yang, Edward And Devito, Zachary And Lin, Zeming And Desmaison, Alban And Antiga, Luca And Lerer, Adam. 2017. Automatic Differentiation In Pytorch. Nips Workshop.

36. Knoops P., Papaioannou A., Borghi A., Breakey R., Wilson A.T., Owase Jeelani, Stefanos Zafeiriou, Derek Steinbacher, Bonnie L. Padwa, David J. Dunaway & Silvia Schievano. 2019. A Machine Learning Framework For Automated Diagnosis And Computer-Assisted Planning In Plastic And Reconstructive Surgery. *Scientific Reports*. 9, P.13597.

37. Personalized Medicine Coalition. 2020. *Personalized Medicine Coalition*. [Online]. Available From: http://www.personalizedmedicinecoalition.org, [Accessed 09 Sep 2020].

38. Neph R., Huang, Y. Yang Y., Sheng K.. 2019. *Deepmcdose: A Deep Learning Method For Efficient Monte Carlo Beamlet Dose Calculation By Predictive Denoising In Mr-Guided Radiotherapy*. Cornell.

39. Rajalingappaa S.. 2017. Deep Learning For Computer Vision: Expert Techniques To Train Advanced Neural Networks Using Tensorflow And Keras. *Packt Publishing*.

40. Sadeghnejad-Barkousaraie A., Bohara G., Jiang S., Nguyen D. 2020. *A Reinforcement Learning Application Of Guided Monte Carlo Tree Search Algorithm For Beam Orientation Selection In Radiation Therapy*. Cornell.

41. Sandru A, Voinea S, Panaitescu E, Blidaru A. 2014. Survival Rates Of Patients With Metastatic Malignant Melanoma. *Journal Of Medicine And Life*. 7(4), Pp.572–576.

42. Scikit-Learn. 2020. *Cross-Validation: Evaluating Estimator Performance*. [Online]. Available From: https://scikit-learn.org/stable/modules/cross_validation.html

43. Srivastava N., Hinton G., Krizhevsky A., Et Al. 2014. Dropout: A Simple Way To Prevent Neural Networks From Overfitting. *Journal Of Machine Learning Research*., Pp.1929–1958.

44. Sultana, S., Robinson, A., Song, D. Y., & Lee, J. 2020. Cnn-Based Hierarchical Coarse-To-Fine Segmentation Of Pelvic Ct Images For Prostate Cancer Radiotherapy. *Proceedings Of Spie--The International Society For Optical Engineering*. 113151i, P.11315.

45. Tanikić D., Despotovic V. 2012. Artificial Intelligence Techniques For Modelling Of Temperature In The Metal Cutting Process. *Metallurgy - Advances In Materials And Processes*.

46. Vallières, M. Et Al. 2017. Radiomics Strategies For Risk Assessment Of Tumour Failure In Head-And-Neck Cancer. *Sci Rep 7*., P.10117.

47. Vallières M., Kay-Rivest E., Perrin L.J., Et Al. 2017. Data From Head-Neck-Pet-Ct. *The Cancer Imaging Archive*.

48. Van Der Walt S., Schönberger J.L., Nunez-Iglesias J., Boulogne F., Warner J.D. , Yager N., Gouillart E., Yu T. 2014. *Scikit-Image: Image Processing In Python*. Peerj.

49. Werthmann, P. G., Kempenich, R., & Kienle, G. S. 2018. Long-Term Tumor-Free Survival In A Patient With Stage Iv Epithelial Ovarian Cancer Undergoing High-Dose Chemotherapy And Viscum Album Extract Treatment: A Case Report. *The Permanente Journal*. 23, Pp.18-25.

B. List of Tables

C. List of Figures

D. Additional Images

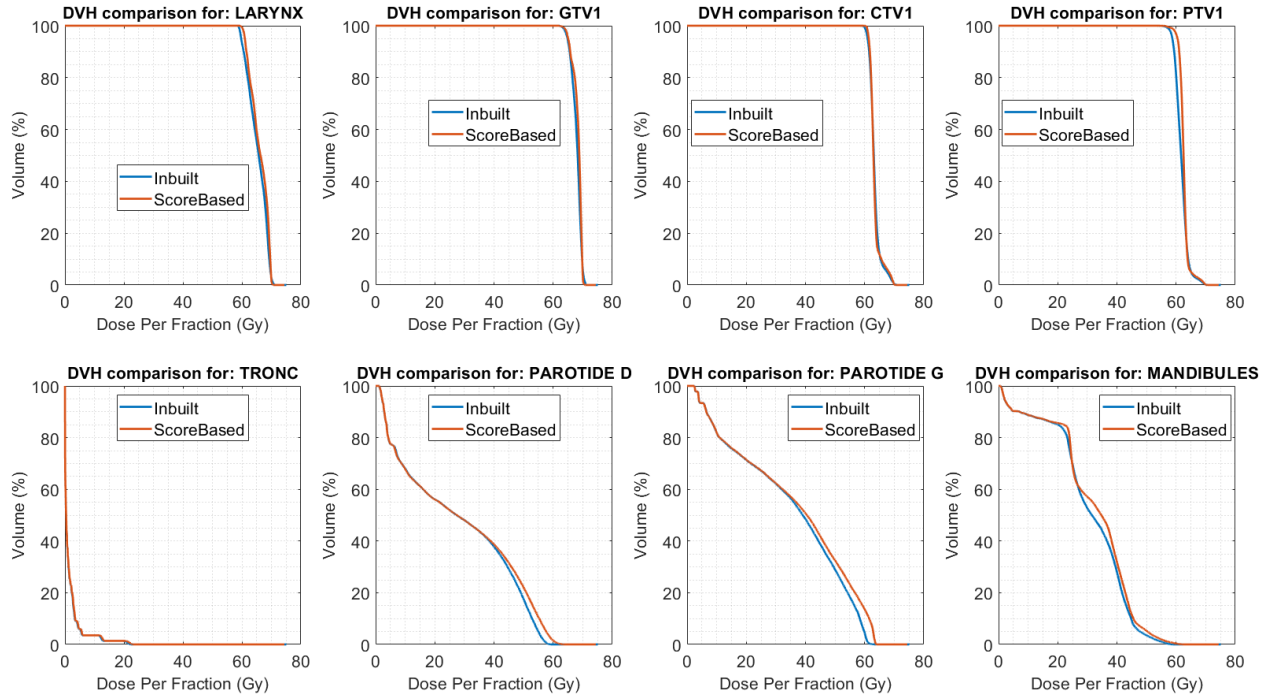

Figure 6.1 – HN-CHUS-044, DVH histograms comparing Target Directed (With/without score assistance)

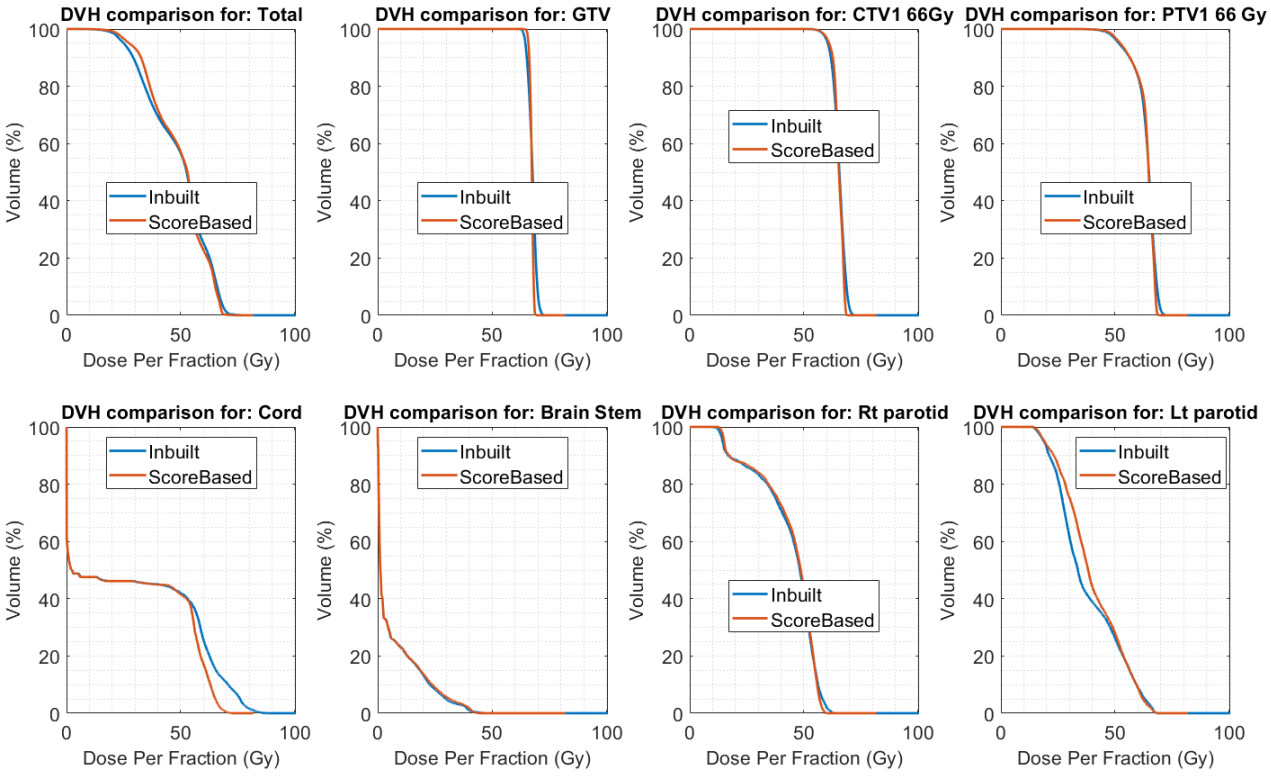

Figure 6.2 - HNSCC-01-0083, DVH histograms comparing OAR Prescription (With/without score assistance)

ROI	GTV	CTV	PTV	B Stem	Paro R	Paro L	Mandible
Target Only	68.262	63.096	61.683	0.423	27.33	39.084	31.551
Target Only + Score Assisted	68.844	62.964	62.514	0.435	27.492	40.506	34.731
OAR Prescriptions	68.076	62.925	61.545	0.408	22.515	30.183	30.654
OAR Prescriptions + Score Assisted	68.691	62.967	62.433	0.432	27.153	39.909	34.68

Table 6.1 – D-50 Dose (Gy), Value comparison for HN-CHUS-044

ROI	Targets	B Stem	Paro R	Paro L	Mandible
Target Only	100	3.07	64.67	78.93	87.97
Target Only + Score Assisted	100	3.41	64.45	78.96	88.07
OAR Prescriptions	100	2.83	62.86	77.85	87.65
OAR Prescriptions + Score Assisted	100	3.4	64.37	78.91	88.03

Table 6.2 – V-12 Percent (%), Value comparison for HN-CHUS-044

ROI	GTV	CTV	PTV	Other OAR	Parotid Right	Parotid Left
Target Only	66.996	64.869	64.677	0	1.317	3.381
Target Only + Score Assisted	67.389	65.472	64.812	0	1.269	3.258
OAR Prescriptions	67.467	65.154	64.677	0	1.29	3.141
OAR Prescriptions + Score Assisted	67.323	65.571	64.983	0	1.278	3.273

Table 6.3 – D-95 Dose (Gy), Value comparison for HNSCC-01-0083

ROI	GTV	CTV	PTV	Other OAR	Parotid Left
Target Only	100	99.88	98.92	0	28.57
Target Only + Score Assisted	100	99.82	98.87	0	28.23
OAR Prescriptions	100	99.91	98.83	0	4.27
OAR Prescriptions + Score Assisted	100	99.77	98.9	0	26.87

Table 6.4 – V-60 Percent (%), Value comparison for HNSCC-01-0083

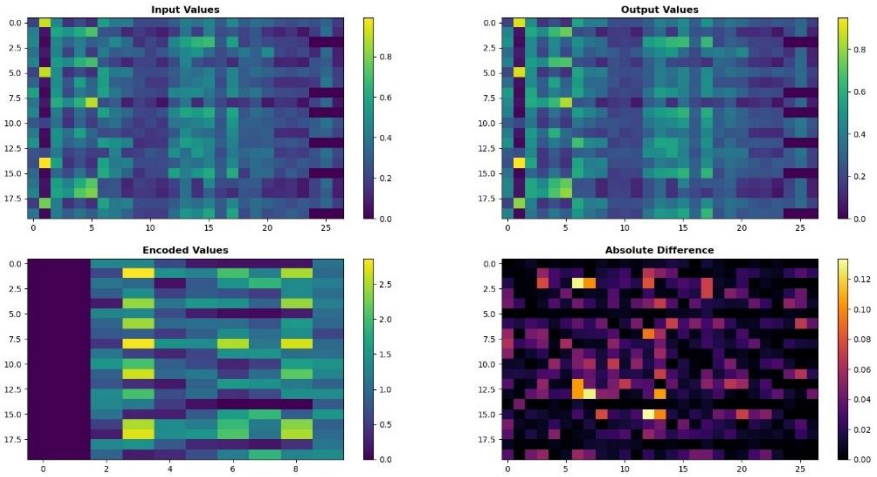

Figure 6.3 – Visual Indicator Of Beam Autoencoding

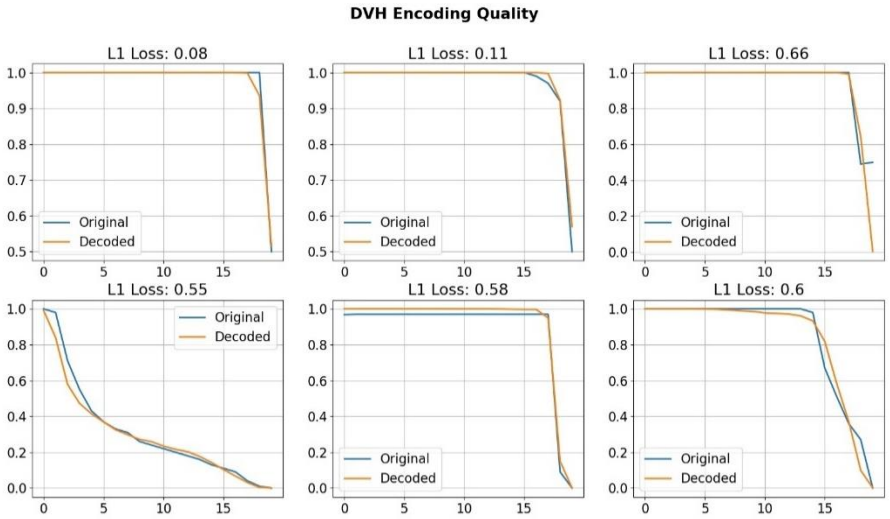

Figure 6.4 – Plot Of DVH Autoencoder Quality

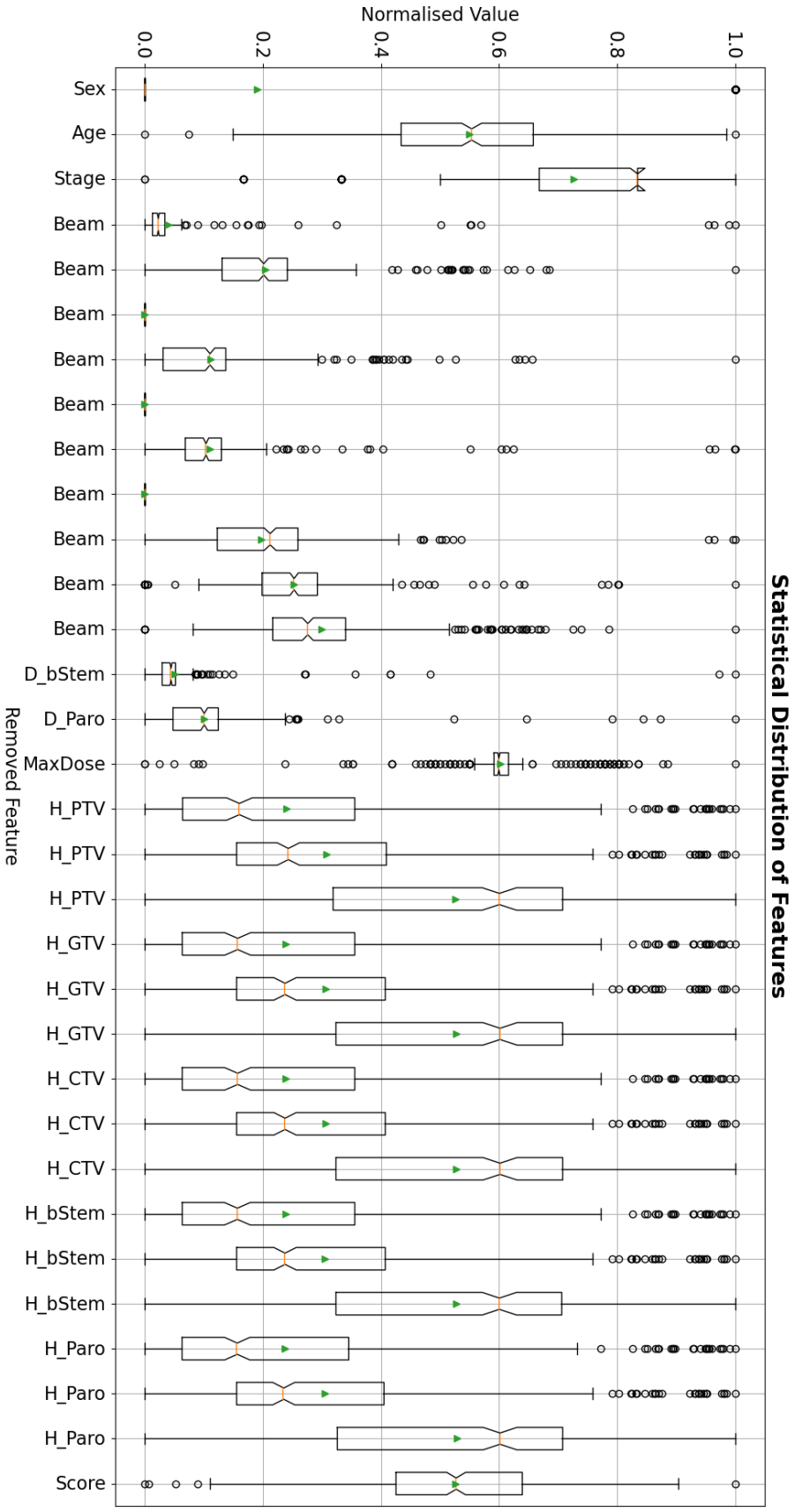

Figure 6.5 – Large Version of Figure 4.9