TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Assessment of Treatment Plan Complexity using Neural Networks

ausgeführt am Atominstitut der Fakultät für Physik
der Technischen Universität Wien

Unter der Anleitung von Univ.Prof. Dr. DI Dietmar Georg
und DI Wolfgang Lechner, PhD

durch

Judith Schinerl, BSc
Hintere Fahrstraße 2
3500 Krems - Stein

Wien, 04.02.2019

Declaration

I, Judith Schinerl BSc, declare that this thesis submitted in partial fulfilment of the requirements for the conferral of the degree Master of Science, from the University of Technology, Vienna, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

_____

Judith Schinerl BSc

04.02.2019

Acknowledgements

I want to thank all the people that were involved in this work, especially Wolfgang Lechner PhD, for detailed guidance throughout the duration of my thesis and for the hours of support and interesting conversations. Many thanks go to Univ.-Prof. Dr. DI Dietmar Georg for supervising this thesis and giving me the opportunity to extend my knowledge in the field of radiation therapy. I would also like to mention Hugo Furtado PhD and Dipl. Ing Lukas Fetty, who supported me in understanding machine learning a bit better and provided helpful advice on how to overcome the problems I encountered.

Lastly I owe my family my deepest gratitude for making it possible for me to follow my education and for supporting me in every way they could.

Abstract

Machine learning and especially neural networks receive more and more attention in current scientific applications, as shown by the increasing number of publications. Especially in medicine and biomedical engineering, where human errors still prove to be a cause of failure, machine learning algorithms are used to overcome the limits of human decision-making. One field that may benefit from the recent developments is radiation therapy, as machine learning algorithms perform well on classifying image data. The standard clinical workflow at the Medical University of Vienna / General Hospital of Vienna (AKH) includes a quality assurance (QA) measurement in advance of each high precision patient treatment. Each treatment is planned beforehand using the software Monaco (Elekta AB, Stockholm, Sweden), which determines beam energies and the positions of collimators of the medical linear accelerator (linac) according to a desired dose distribution covering delineated regions of interest on computer tomography images. The QA measurement on the linac is then performed with a verification phantom, and finally planned and measured dose distributions are compared in order to ensure safe dose deposition in the patient. The gamma passing rate (GPR) serves as a measure of conformity of these two dose distributions. The GPR depends on the size, shape and location of the tumour in the patient and has to exceed a certain value in order for the plan to be regarded as safe to irradiate on a patient. In this thesis, the setup and training of a convolutional neural network (CNN) with the aim of classifying treatment plan data by estimating the GPR is described. Achieving this task with sufficient accuracy will enable a higher efficiency of the QA procedures and more efficient use of the medical linear accelerator.

A neural network is a deep learning concept mimicking the information processing in human neurons by its layered structure. Each layer is composed of a number of nodes, called neurons. They are connected to other neurons in the previous and/ or subsequent layers, each connection being weighted by a weighting function. The training process involves the passing of input data with known output to the network, i.e. data that has been labelled according to their corresponding class. To obtain input images, 600 volumetric modulated arc therapy (VMAT) treatment plans for

either prostate, gynaecological or head-and-neck (HN) cancer generated for either Versa HD or Synergy linacs (Elekta, Sweden) were extracted and assigned to one of three labels according to their GPR value. The planning data, i.e. the dose and positional information of all collimators contained in the planning data in Dicom (digital imaging and communication in medicine) format was then transformed into grayscale fluence maps, depicting the transmission of dose through the beam window. The complete dataset was separated into three smaller datasets designated for training, testing or evaluation purposes. During training, which was performed in Python using the framework TensorFlow, two datasets were used to set the weights in order to output the known label of each input accordingly following an optimisation operation. Several different models were trained, varying the learning rate, batch size and depth of network in order to improve achieved training accuracies and further testing the robustness of the resulting layer structure by switching and shuffling the datasets.

The achieved training accuracies range from ~57% to ~69%, showing a large variation upon changing the layer sequence and parameters. Furthermore, robustness testing revealed large variations of accuracy upon switching the used datasets, leading to accuracies between ~59% and ~69%. Evaluating the best performing convolutional neural network on unknown data, i.e. the third dataset not used during training, resulted in an evaluation accuracy of 59.5%, showing a reduction of 10% compared to the training accuracy of ~69%. Similar values can be found in recent literature evaluating fluence maps of radiotherapy treatments according to the associated GPR[1]. Since the obtained results only offer a first insight on the performance and behaviour of CNN, various approaches to increase the achieved accuracies and enhance network robustness have been identified. Improvements with respect to accuracy and robustness are necessary for utilizing these CNNs in a clinical workflow but go beyond scope of this work, as the objective was to identify general mechanisms and problems of neural networks in radiotherapy. The results obtained in this thesis show the potential of CNNs acting as a promising new approach for applications in quality assurance in radiation therapy.

[1] Interian et al., 2018 and Nyflot et al., 2018

List of Abbreviations

AKH	Allgemeines Krankenhaus
API	Application Programming Interface
3D-CRT	Three dimensional Radiotherapy
BEV	Beams Eye View
CNN	Convolutional Neural Network
CPU	Central Processing Unit
Dicom	Digital Imaging and Communication in Medicine
GPR	Gamma Passing Rate
GPU	Graphics Processing Unit
HN	Head and Neck
IDE	Integrated Development Environment
IMRT	Intensity Modulated Radiotherapy
Linac	Linear Accelerator
MAE	Mean Absolute Error
MLC	Multileaf Collimator
MU	Monitor Units
OAR	Organ at Risk
PNG	Portable Network Graphics
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SOBP	Spread out Bragg Peak
TPS	Treatment Planning System
VMAT	Volumetric Modulated Arc Therapy
XML	Extensible Markup Language

Contents

1 Introduction

In an aging society, illnesses that are more likely to develop with increasing age such as cancer are on the rise in Austria. However, cancer is not only an issue of the aging population. In 2015, there was a 28% chance of being diagnosed with cancer before turning 75 [1]. Malignancies are responsible for about a fourth of all deaths [2]. Therefore, radiotherapy is a growing and evolving field and has seen many improvements over the last years. This chapter introduces the main concepts of radiation therapy and its methods, the underlying fundamental physics, as well as treatment planning and delivery in section 1.1. The different particle types that can be used as well as the most common delivery modalities are discussed. Section 1.2 is dedicated to machine learning and the general approach and mechanisms present in today's research, with regard to the network type that is used in this thesis.

1.1 Radiotherapy

The main goal of radiotherapy is to deliver a maximum dose to the tumour while sparing as much of healthy tissue as possible. This is achieved by applying ionising radiation to the patient, which consists of charged or uncharged particles with an energy higher than the energy that is needed to ionise atoms. This radiation causes strand breaks in cell DNA, leading to cell death [3]. This happens either by directly ionising particles hitting the DNA double helix or by creating free radicals upon colliding with oxygen or hydrogen atoms, which can then react with the DNA strand itself. Charged particles ionise directly, while uncharged particles compose the indirectly ionising radiation and create secondary ionising particles and free radicals upon entering tissue.

Due to the reduced ability to repair themselves, cancer cells are more affected by radiative damage compared to healthy tissue, which makes radiation therapy possible. Treatments are delivered in fractions, where the total dose a patient receives is split up in smaller doses and irradiated a number of times. This approach gives healthy cells time to repair until the next fraction is irradiated, therefore targeting mainly cancerous cells [4].

Radiotherapy is a rather old concept, the first treatment being delivered in 1896 by Austrian radiologist Leopold Freund only one year after the discovery of x-rays [5]. It since became a standard method of treating cancer patients, along with surgical removal of the tumour and chemotherapy. In certain cases, combinations of the aforementioned treatments are possible, for example administering radiation on residual tumour tissue if a complete surgical removal of the tumour is not possible.

Over the years, advanced methods have been developed that allow precise deposition of dose in the tumour and spare the healthy tissue around, both in delivery machines, treatment planning software and imaging, which allow a precise definition and irradiation of target volumes.

1.1.1 Delivery methods

Depending on the mode of delivery, external beam (teletherapy) and internal (brachytherapy) radiotherapy can be distinguished. Brachytherapy denotes the medical practice of inserting small sealed radioactive sources in cancerous tissue, placing probes in body cavities or placing sources externally on the skin. The deposition of dose is carried out locally and the time the sources remain in or on the patient is individually set, ranging from a few minutes to 24 hours per treatment fraction for temporary implants. Permanent implants stay in the patient's body but the radioactive material decays completely after several months. Brachytherapy is most commonly applied to the prostate, breast, uterus, vagina and cervix as well as around the mouth cavity, the eye and on the skin [6].

In teletherapy, the dose is delivered via an external radiation source, e.g. a linear accelerator (linac, see 1.1.4). The beam exits the linac from the gantry, which is able to rotate around the movable patient table and enables the irradiation of the tumour from different angles. For this treatment method, different ionising particles as well as X-ray photons can be used and are chosen according to their dose deposition properties.

1.1.2 Beam qualities

The central distinguishing feature of radiation types that determines the mode of treatment is the particle type. Charged particles, such as electrons, protons or heavier ions like carbon or boron are responsible for directly ionising radiation, while electromagnetic radiation, which is commonly described as a stream of photons, causes indirect ionisations in tissue. Photon and electron therapy devices have a similar but, compared to particle therapy, a comparatively easy setup in the form of a medical linac which accelerates electrons. Optionally, photons can be created by positioning a retractable transmission target in the beam path, which is hit by electrons (see 1.1.4). Therefore, radiation therapy using photons is one of the most common delivered treatments today. Subsequently, the topic of this work will be narrowed down to photon therapy, which is the most common treatment at the Medical University of Vienna / AKH Vienna and around the world.

The deposition of energy in a medium, in that case human tissue, is described by the linear stopping power [7]. This property differs for each particle type, therefore different particles serve different applications. As illustrated in Figure 1, the depth at which most of the dose is deposited depends on particle type and energy.

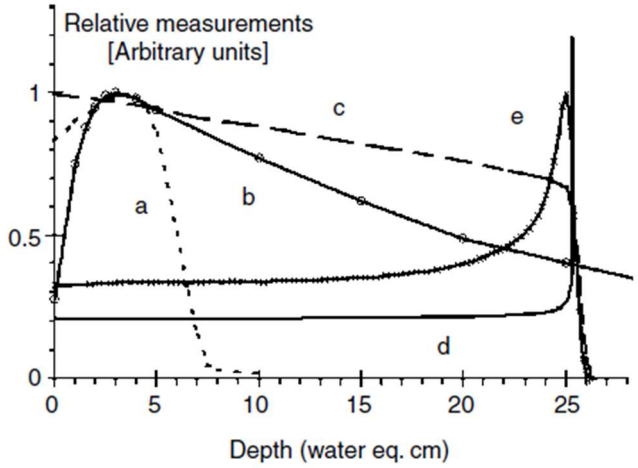


Figure 1: Depth-dose curves for a) 15 MeV electrons and b) 15 MV photons, c) fluence of 200 MeV protons, d) stopping power of 200 MeV protons and e) depth dose curve of 200 MeV protons [7]

For electron beams, curve a) shows the characteristic dose build-up-region after entry into the tissue, reaching a maximum of dose deposition shortly after. Since electrons are light charged particles, they undergo scattering reactions upon travelling through

the patient, which at first leads to an increase in the obliquity of electron paths since they are scattered from the original incident direction, increasing the electron fluence with depth until a maximum of dose deposition is reached. This is the case when the beam is completely diffuse and the mean scattering angle reaches a maximum. A steep dose fall-off is seen after the dose maximum due to further scattering of electrons, which enables sparing of the deeper situated tissues. Taking advantage of the shape of the depth-dose curve, electron beams are used to treat tumours at or close to the surface [7].

In comparison to electrons, photons of the same energy exhibit a significantly lower surface dose upon entry into tissue, while reaching roughly the same maximum dose after a dose build-up-region, as depicted by curve b). The fall-off, however, is less pronounced, the photons are able to penetrate deeper into tissue. The build-up is due to the generation of secondary electrons mainly by Compton scattering and the deposition of their energy along their path further away from the location of interaction. The dose reaches a maximum at a depth approximating the electron range, at which an electronic equilibrium is reached. Photon absorption and scattering leads to decreasing numbers of Compton electrons, which causes the dose to decrease after the dose maximum. Photon beams show a favourable surface dose in comparison to electrons and can therefore be used to treat tumours at a larger depth with better tissue sparing.

In recent years, radiation therapy with protons and heavy ions has gained more importance due to their favourable depth dose profile, which shows a Bragg-peak at a certain depth where most of the dose is delivered shortly before the particles are stopped by tissue (see curve e) in Figure 1). The depth protons can reach inside the patient's body is determined by the particle beam energy, therefore adjusting the energy to reach a specific depth in the patient's body is possible. Overlaying beams of different energies creates a favourable dose deposition in the patient, with proton beams creating a spread out Bragg peak (SOBP), covering a volume with a relatively homogeneous dose, sparing the tissue behind almost completely, as illustrated in Figure 2.

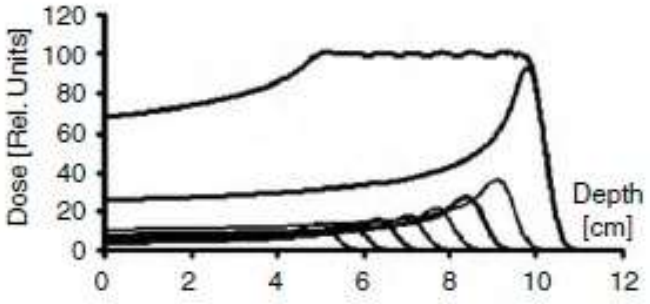


Figure 2: Spread out Bragg peak of a proton beam treatment, achieving a plateau of dose at the desired depth [7]

The downside of proton and heavy ion radiotherapy is the need for an accelerator in the form of a cyclotron or synchrotron, which – in contrast to linear accelerators used for electrons – has to be rather large and are therefore more expensive. Due to the higher weight of protons and ions compared to electrons, acceleration and steering the beam to the treatment rooms is more challenging. Nevertheless, more and more proton and heavy ion treatment centres are built aiming for a better patient treatment. One project especially important for Austria is the MedAustron, located in Wiener Neustadt.

1.1.3 Photon interactions

Since this thesis deals with photon therapy, the three main interactions they can undergo upon colliding with matter will be described further. These effects include the photoelectric effect, the Compton effect as well as pair production. As demonstrated in Figure 3, these interactions show a different contribution to the total interaction probability depending on photon energy. The concept of cross sections is used to indicate the probability of an interaction by describing an area around the photon. Most relevant in radiotherapy, which is using photons in the energy range of 3 MeV up to 30 MeV, are the Compton effect and pair production.

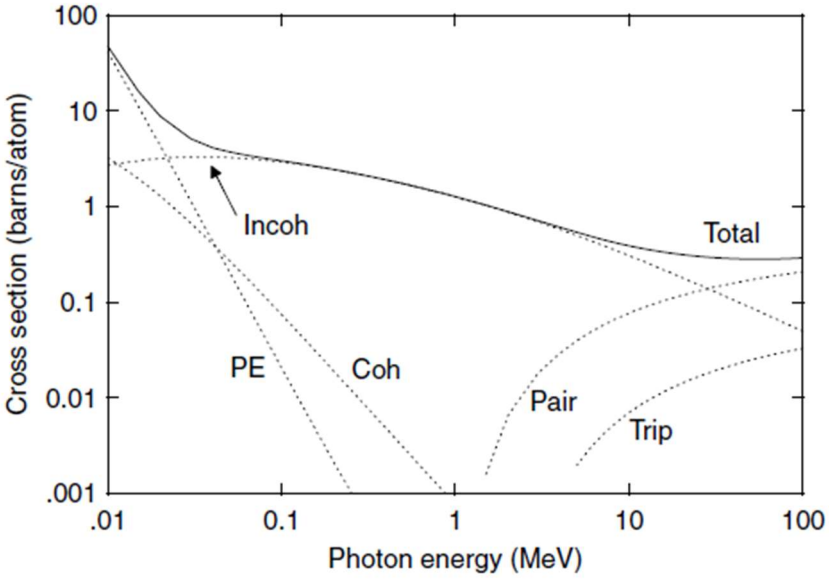


Figure 3: Contributions of different effects to total cross section of photon absorption in carbon [7]. Considered effects are the photoelectric effect (PE), Incoherent or Compton scattering (Incoh), Coherent or Rayleigh scattering (Coh), Pair- and Triple production (Trip).

The photoelectric effect describes the ionisation of an atom, i.e. the removal of a bound electron by the absorption of an incoming photon. This can only occur if the photon energy is larger than the binding energy of the electron and is the dominant effect in the lower energy range. The Compton effect, also termed Compton or incoherent scattering, describes a scattering process. The incoming photon interacts with an electron and transfers part of its energy to the electron, which is then ejected from the atom. The loss of energy causes the outgoing photon to travel at an angle compared to the incoming one. The third effect, pair production, becomes a contributing factor for higher energies. An incoming photon is absorbed in the electric field of a nucleus and produces an electron-positron pair. Both electron and positron have a rest energy of 511 keV, thus for this effect to happen, photon energies of at least 1.022 MeV are required [4].

1.1.4 Medical linac and Beam shaping

In radiation therapy using photons, linear accelerators are used to generate the photon beam. The electrons produced by an electron gun are accelerated by radiofrequency waveguides and focused onto a target, leading to production of X-rays. Before the beam enters the patient, a collimator, a flattening filter as well as a dual ion chamber are arranged in the beam path to achieve a uniform beam with homogeneous intensity distribution. The ion chambers measure ionisation of the contained gas and act as a safety measure to monitor the delivered dose.

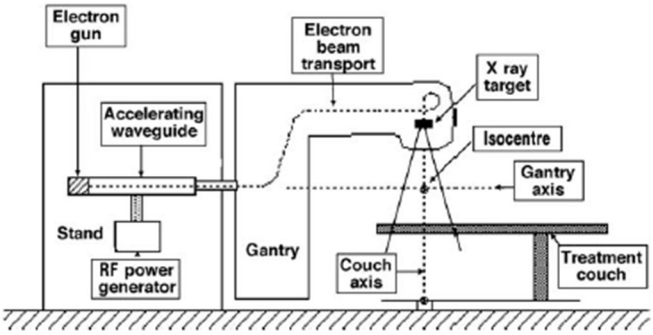


Figure 4: Schematic image of a linac setup . If the X-Ray target is removed, irradiation with electrons is possible [8].

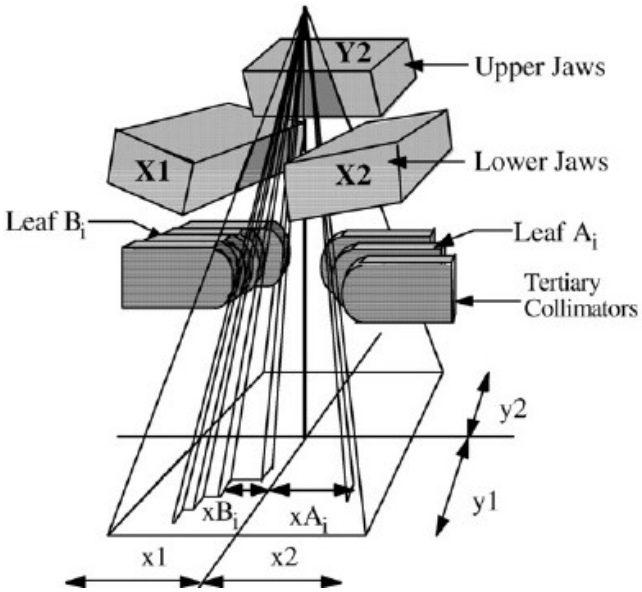


Figure 5: Collimators that shape the beam [9]

To further shape the beam and avoid irradiating a larger volume of tissue than necessary, usually two pairs of collimator jaws confine the beam in x- and y-direction (with z being the coordinate axis along which the beam travels, also called beams eye view, BEV), forming a rectangular field. The last and possibly most important step of beam shaping occurs after the jaws, where the multileaf collimator (MLC) is located. This collimator consists of a number of pairs of leaves mostly made out of tungsten; each one is moved independently to form the desired tumour shape, as shown in Figure 6. Depending on the treatment mode, this is done in a step-and-shoot manner, irradiating from fixed gantry positions around the patient only when leaf positions are fixed. The other possible mode involves the continuous movement of leaf pairs during the irradiation from several different beam angles. Regarding the observed plans for this work, Volumetric Modulated Arc Therapy was used, where additionally to MLC leaf movement also the gantry rotates around the patient during irradiation (see 1.1.5.3).

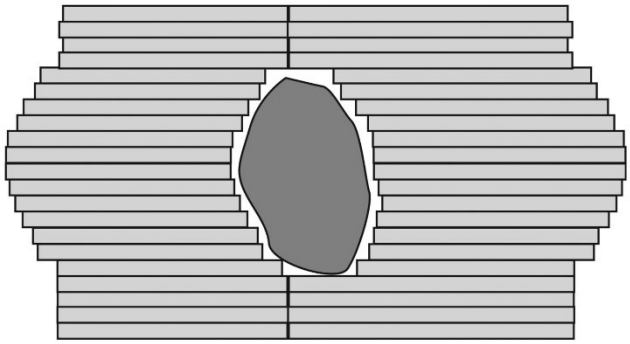


Figure 6: Multileaf collimator in BEV to adjust beam cross section to target shape in 3D-CRT [10]

1.1.5 Treatment Modalities

With the aim of improving dose deposition in the patient and therefore maximising healthy tissue sparing, multiple irradiation techniques have been developed. One improved modality is 3D conformal radiotherapy (3D-CRT), which achieves better dose distributions compared to former techniques only matching height and width of tumours by overlapping beams from different beam angles, aiming to conform to target volume shapes. Further improvements in aligning to tumour shape is given by

intensity modulated radiotherapy (IMRT), incorporating different beam intensities within a single beam. In Volumetric Modulated Arc Therapy (VMAT), improvements go another step further by rotating the gantry during the irradiation [11].

1.1.5.1 Three-dimensional Conformal Radiotherapy (3D-CRT)

Fulfilling the main requirement of radiotherapy of irradiating tumours of simple shape while sparing healthy tissue leads to one technique: 3D conformal radiotherapy. In a forward planning manner, beams are positioned around the patient, overlapping and therefore depositing the most dose over the target volume [11]. Most modes apply between two and ten beams, exposing more healthy tissue to a lower dose with increasing number of beams. Individual beam shaping and weighting is achieved by using multileaf collimator (MLC) to confine the beam to the projected target shape and by using wedges to account for different tissue thickness. The single beams only incorporate one energy, therefore adapting precisely to complex tumour shapes is rather difficult and shows the need for improved treatment techniques.

1.1.5.2 Intensity Modulated Radiation Therapy (IMRT)

A newer technique to improve dose distribution especially in regions surrounded by organs at risk or complex tumour shapes is IMRT – Intensity Modulated Radiotherapy. This method makes use of different intensities within a single beam, making a more precise and tissue sparing irradiation possible. In contrast, conventional radiotherapy methods use beams that only have one uniform beam profile and achieve the dose distribution in the body by overlaying of several beams and the use of collimators to align the beam dimensions to the tumour shape. For computation, IMRT techniques divide the beam into smaller sub-segments, termed as beamlets. These beamlets can be weighted differently, making it possible to generate non-uniform dose distributions. The delivery of these beamlets is achieved by using the (MLC), either in step-and-shoot or in dynamic mode.

In step-and-shoot IMRT, each beam orientation delivers several differently shaped segments depending on the tumour size and outline. The radiation is only turned on as soon as the MLC leaf pairs reach their fixed position and the shape of the opening corresponds to the segment, in this manner covering all the segments sequentially.

The dynamic mode requires the leaf pairs to move continuously during the irradiation, covering the whole tumour shape in beams eye view (BEV) for each gantry position [11].

1.1.5.3 Volumetric Modulated Arc Therapy (VMAT)

Volumetric Modulated Arc Therapy (VMAT) refers to a mode of IMRT that continuously irradiates the patient while the gantry performs one or more full rotations (arcs) [12]. This approach inevitably covers a large amount of healthy tissue but allows a lower dose that only adds up in the tumour. Furthermore, the rotation speed as well as the dose rate can be varied and the MLC leaf pairs move continuously throughout the irradiation. Additionally, the reduced irradiation time in comparison to static field treatments makes the method advantageous to conventional approaches in certain cases [13].

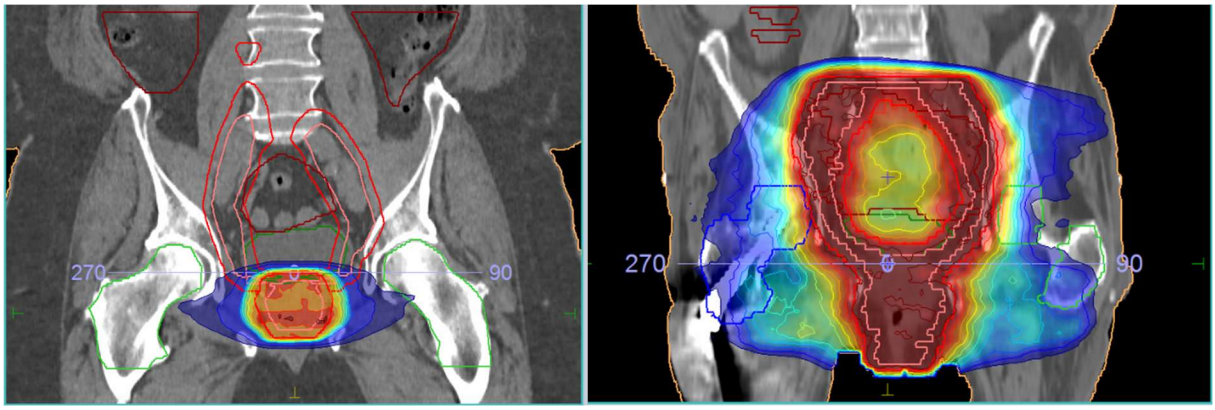
1.1.6 Treatment planning

Treatment plans are created for each patient before the actual treatment, where two approaches may be distinguished depending on the procedure used. In forward planning, the planner arranges beams in the treatment planning software, overlapping them at the tumour site based on 3D anatomy, e.g. the acquired CT images of the patient. This method can only be used for rather simple target geometries, irradiating with conventional radiotherapy methods and utilising from only 2 up to 10 beam entry angles.

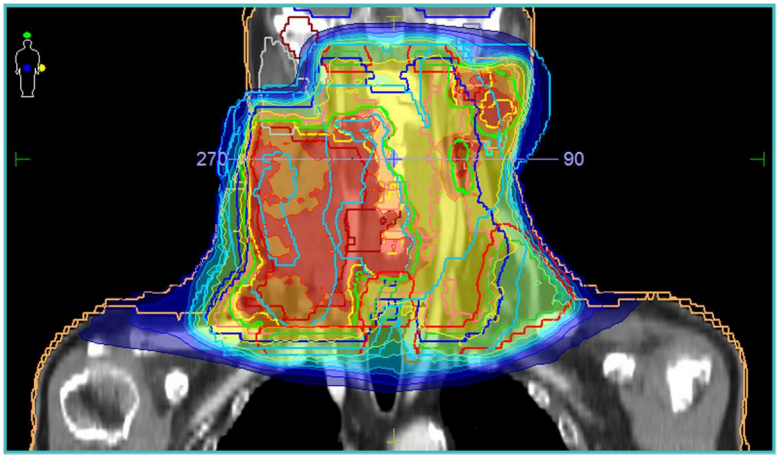
Today, IMRT is established as a standard and more frequently used, thus inverse planning is gaining more interest since it is necessary for this method. In inverse planning, which is also done for 3D-CRT treatments, CT images are used to delineate the volumes of interest. These include the tumour, a surrounding margin to counteract movement of the patient and machine inaccuracies, as well as the OAR (organ at risk) volumes, marking radiosensitive organs and tissues in the patient's body that should receive little to no dose.

According to the delineation conducted by a medical doctor, optimisation programs utilizing dose calculation algorithms (e.g. Monte Carlo or eCC) for dose calculation

then find the optimal way to apply beams to the volumes from different angles. The software further creates the complete treatment plan including the information of all machine parameters to achieve the simulated dose distribution. The complete treatment plans then provide information about the three-dimensional dose distribution in the patient.



a) b)



c)

Figure 7: Dose distributions for a): localised prostate cancer, b): cervix cancer, c) head and neck cancer patient

In Figure 7 the planned dose distributions are shown, with red being the area receiving the highest and blue marking the areas receiving the smallest amount of dose. To verify IMRT and VMAT plans, measurements on a phantom have to be performed to verify the machine is able to fulfil the requirements according to the plan. If the plan is approved, the patient receives the treatment.

1.1.7 Patient Specific Quality Assurance

According to current QA protocols, every treatment plan is measured on a phantom before applying the plan to the patient to ensure the precise delivery of dose is possible with the used medical linac by measuring the 3D dose distribution. Using a respective software, a direct comparison between measured and computed dose is made and quantitatively described using the gamma passing rate (GPR). This value evaluates the conformity of the measured dose distribution to the planned one and therefore serves as a criteria of quality for the tested plan, determining if irradiation on the patient is safe and executable [14]. It is given as a percentage of points that fulfil two criteria: the dose deviance (DD), setting a maximum divergence of measured dose to a certain fraction of the planned dose, as well as the distance-to-agreement (DTA), which describes the maximum spatial distance of the measured in comparison to the planned dose distribution.

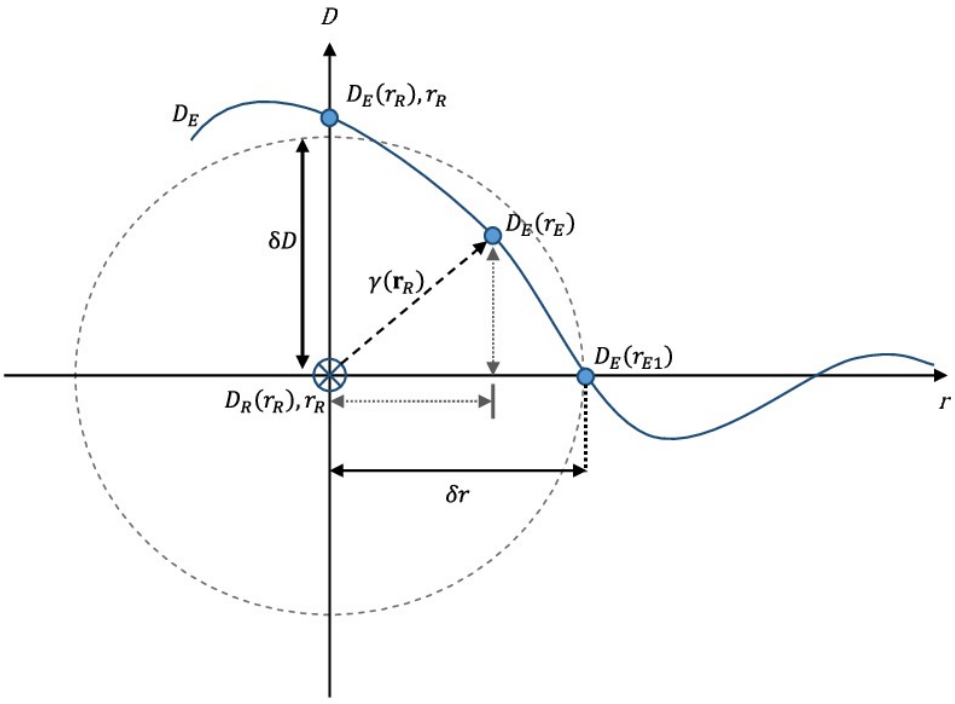


Figure 8: Schematic depiction of the calculation of the gamma index with the distance r plotted on the x-axis and the dose D on the y-axis. The blue curve shows the measured dose values D_E at different spatial points r_i relative to the planned value of the dose at the origin $D_R(r_R)$, r_R. The dotted circle represents the DTA and DD criteria, with δD being the margin for dose deviance and δr the margin for the distance to agreement. If a point along the blue line that lies within the dotted circle can be found, the gamma index of this point is below or equal to 1 [15].

One frequently used DTA/DD criteria margin is 3mm/3% but is somewhat arbitrary. Decreasing these numbers to 2mm/2% and therefore increasing the required conformity would result in a high number of treatment plans not passing the gamma criteria. However, with constantly improving delivery and planning techniques, dose may be deposited more precisely, leading to the possibility of adjusting the passing criteria in the future, consequently improving the delivered treatments.

1.2 Machine learning

Artificial intelligence (AI), a branch of computer science that is used in numerous different industries and research topics today, is the general term for automating human learning processes in machines and therefore mimicking human intelligence [16]. A multitude of different techniques exist to serve its wide-ranging application areas, including natural language processing, speech, vision, robotics and machine learning. Machine learning specifically focuses on training and teaching machines to recognize connections and patterns by using a large number of examples as a database. During the training process, machine learning algorithms identify parameters and patterns in order to characterise and classify the input data, so that upon being presented with new data, those learned features are used to classify unknown data [17]. Deep learning is a subgroup of machine learning, describing the layered structure of learning and decision-making models, one of them being neural networks, which are the main focus of this work. Prominent examples of the use of machine learning today are determining handwriting, face recognition or distinguishing and classifying objects in images [18], [19].

Especially in biomedical research, machine learning gathers more and more attention: numerous applications involving machine learning components are tested and researched on in an attempt to avoid human mistakes when it comes to medicine and treating patients [20] or in order to make qualified predictions [21], [22]. Specific examples include neural networks and machine learning algorithms in general to be used for diagnosis of heart conditions by analysing the heart sounds [23] or identifying skin lesions by using images [24]. In light of recent advances in software as well as in

hardware and imaging techniques, radiotherapy is a field that can highly benefit from implementing machine learning routines into the standard workflow. One step of treatment planning in radiation therapy requires a medical doctor to delineate the target volume which is defined as the tumour volume including a safety margin, as well as and organs at risk, the healthy tissue that must not receive any dose during treatment (see 1.1.6). This delineation is performed by hand through a medical doctor on CT or MRI images and builds the basis for the treatment planning software, which arranges the beam energy and entry angles according to the required dose distribution. Since the accuracy of the delineation strongly depends on the executing medical doctor and the highly individualised problem, methods to improve the correctness of these delineations are developed. Naturally, as in this case, edge detection and image recognition is present, machine learning has been used to delineate volumes [25]–[27].

Machine learning can be roughly divided into supervised, semi-supervised, unsupervised and reinforcement learning. Supervised learning describes learning processes with already labelled data, therefore the model is built using the knowledge of the outcome. Unsupervised learning tries to identify new data solely by finding and memorizing commonalities in the unlabelled input data, while semi-supervised learning combines both above approaches and uses labelled data to optimize predictions made with unlabelled data. Reinforcement learning describes methods that are trained by rewarding certain actions that are taken. The so-called software agents hereby learn by trying to maximize the rewards that are given after certain goals are reached [20]. Reinforcement learning can be implemented with different methods, among those are Monte-Carlo-Algorithms and neural networks.

1.2.1 Neural Networks

As a subgroup of machine learning methods, neural networks will be described further. The general concepts of neural networks and machine learning were developed in the 1940s, but it is only now that computational power is strong enough to build complex machine learning networks that can be used for current

research [28]. One early example and the predecessor and ground concept of neural networks is the perceptron, as described by Rosenblatt in 1958 [29].

The perceptron is the simplest decision making model and consists of a single logical element, which receives a number of inputs which are then translated into an output. Mimicking the process of an action potential travelling through neurons in the human brain, a perceptron receives weighted information and outputs information if the sum of all the inputs exceeds a certain threshold, described mathematically with a step-function, as depicted in Figure 9. Adding more layers with more perceptrons results in a multilayer structure which can be used to predict simple models.

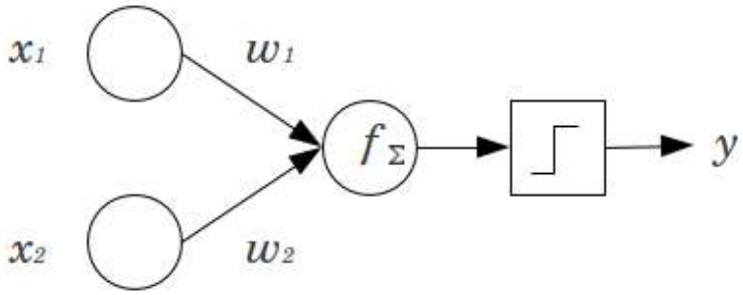


Figure 9: Structure of a single-layer perceptron , receiving two inputs, x_i, with their respective weights ω_i, which are summed up and passed to a step-function to determine the output y, which, in this simple case, results in either 0 or 1 [30].

Deep neural networks used in research today are based on the perceptron model and expand it to build large layered structures. Neural networks consist of a number of layers, each containing up to thousands of neurons, named after their biological example [31]. The main difference in comparison with the multilayer perceptron is the two- or three-dimensional arrangement of neurons, the third dimension accounting for different input image colours on the RGB scheme. The very first layer consists of the input, one neuron representing one parameter of the input data. In case of using image data as input, each pixel of the image forms one neuron in the first layer. The final layer, also called the output layer, either estimates a value for the given problem when a linear regression model is used, or classifies the input data into different groups, forming a classification model. In the second case, the output

layer consists of the same number of neurons as there are labels defined for the current problem. All layers in-between are termed “hidden layers”, serving several different purposes. The number of hidden layers defines the “depth” of a network, if it consists of a large number of hidden layers it acquires the attribute deep [32].

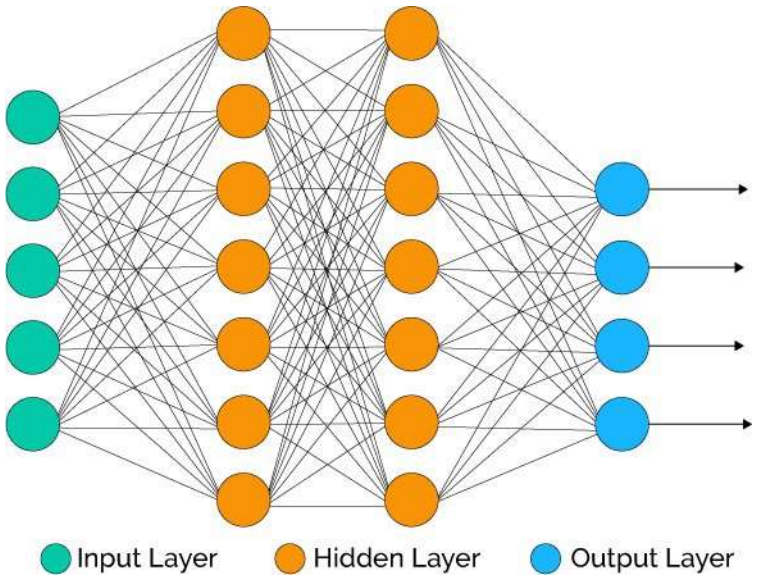


Figure 10: Schematic structure of neural networks [33].

Neural networks are built as depicted in Figure 10, consisting of a number of neurons per layer. In analogy to the human brain, the neurons in different layers are connected, either feeding information forward to the subsequent layer (feed-forward network) or transferring information to the same or back to the previous layer (recurrent neural network). All connections are weighted, either resembling a human excitatory neuron if the weight is positive or an inhibiting neuron in case of a negative weight. The weights determine the influence of a neuron on the connected neurons in the subsequent layer. Similar to the perceptron, the activation of a neuron is dependent on the sum of input it receives. The output is described using activation functions, which may have different shapes. Most used are the step-function, the sigmoid function as well as the ReLU, the rectified linear unit [34].

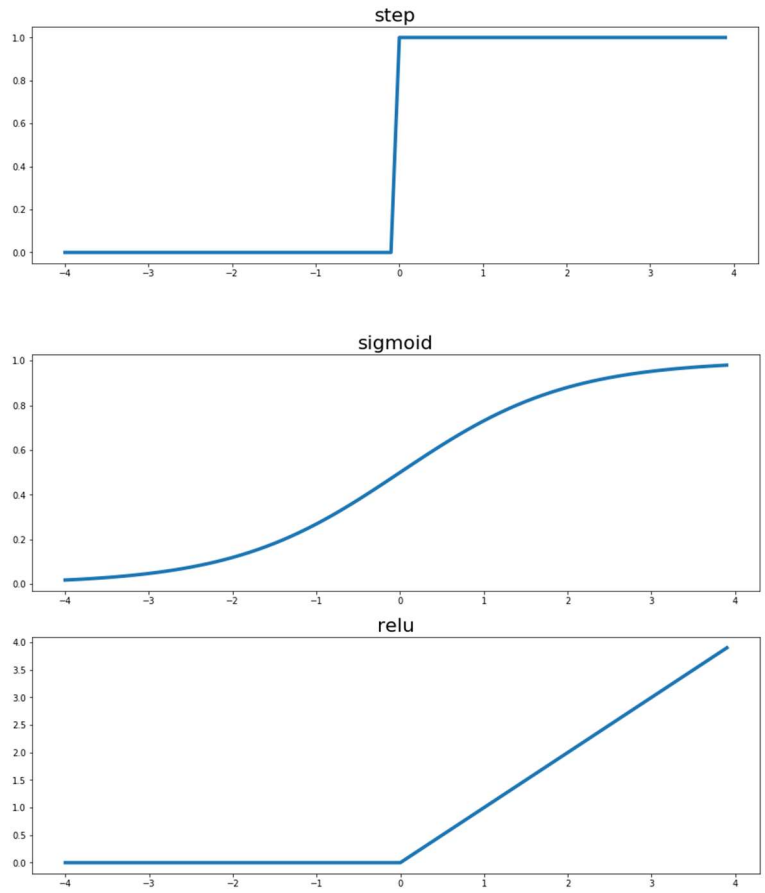


Figure 11: Three most popular activation functions: Heaviside step-function, a sigmoid function and the ReLU, rectified linear unit [35].

Serving as the easiest activation function and most obvious representation of the threshold model, the Heaviside step-function describes the activation of a neuron if a certain input value is exceeded, therefore having only two possible outputs, 0 or 1. Logical XOR-problems can easily be described using this function. However, since for further computation and deep networks the derivative is needed, this function is not used in recent models. For more complicated models that require a non-binary output, the sigmoid function may be used as an alternative. This function represents a smoothed step-function, only converging to 0 for negative and 1 for positive real input values, mathematically describing the increased firing rate found in human neurons if the input is increased. Non-trivial models may be predicted using this activation function and it is able to describe probability due to its values between 0 and 1. The third and probably most used activation function, the rectified linear unit (ReLU), returns 0 for all negative values and increases linearly with values larger

than 0. The use of ReLUs as activation functions improves computation time, shows advantageous performance in comparison with other activation functions during the training process and is a standard in convolutional neural networks. Additionally, the ReLU prevents the neural network from the vanishing gradients problem, since – in comparison to the sigmoid function – it does not converge to a value and therefore a gradient can be obtained also for large input values [36].

1.2.1.1 Layers of a Convolutional Neural Network (CNN)

One class of supervised, deep, feed-forward networks that is frequently used and will be described further are convolutional neural networks. These networks are formed by a number of layers of different shape, which can generally be divided into two groups concerning their function. Convolutional and pooling layers serve the purpose of identifying features and finding patterns, whereas flattening and especially fully connected layers are responsible for the actual classification [37].

The convolutional layers perform a convolution on the input data. So-called kernels or filters of a chosen size scan the input matrix in a stepwise manner, calculating the inner product as depicted in Figure 12. The resulting, significantly smaller matrix is the so-called feature map, containing only certain features of the input information. The enhanced features depend on the values of the kernel, which are set during the training process [37]. Following the concept of shared weights, the kernel values do not change within the same layer, thus reducing the number of total parameters that have to be set during training and also improving computation time which makes CNNs preferable to other models. The convolutional layer represents the receptive field present in the brain, reducing the number of neurons that are used as input for the next layer. The “step size”, the number of neurons the kernel moves for generating each convolved feature is called stride and is usually set to 1, so adjoining features have overlapping receptive fields. The process of surrounding the input matrix with zeros is called padding, and can be chosen to use in order to keep the number of neurons constant in both layers.

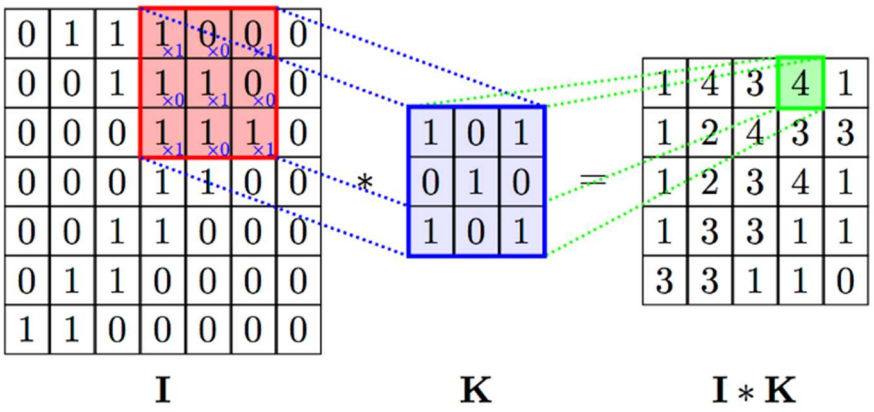


Figure 12: 2D representation of convolution operation with a kernel K on input data I, resulting in a feature map [38].

Convolutional layers are followed by pooling layers. These offer a method of further reducing the neuron number by either choosing the most active neuron within a certain cluster of neurons (max pooling), calculating the mean value (average pooling) or summing up all the neuron values within the cluster (sum pooling). These layers are used to remove redundant information in order to improve computation time and allow for deeper networks.

The layers that compute the classification start with a flattening layer. This layer arranges all the neurons into a one-dimensional vector, which is needed for the succeeding layers, the fully connected layers. These are also called dense, describing the fact that all the neurons of these layers are connected to all the neurons of the previous layer. The final layer, the output layer, is in fact also a fully connected layer, containing the number of classes set for the model. Linear regression models, which calculate a value, have only one neuron in the output layer. One possible complete layer sequence of a CNN is depicted in Figure 13.

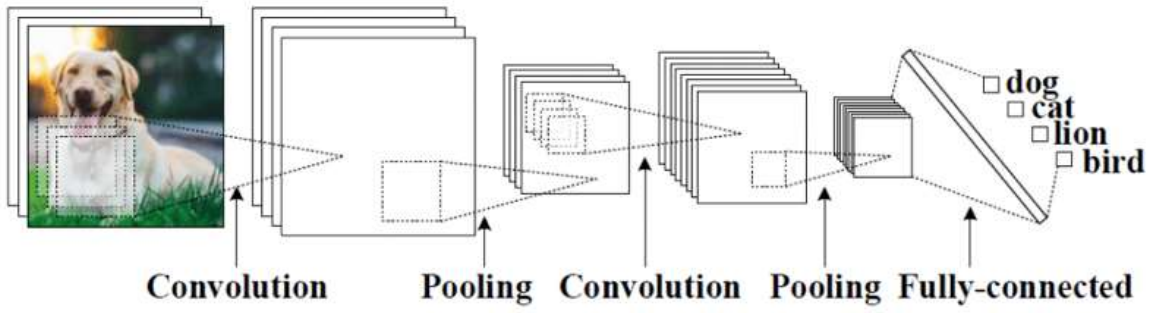


Figure 13: Schematic depiction of layer sequence in a CNN used for a classification problem [39].

1.2.1.2 CNN Training process

For the training process, two different datasets are needed: the training and testing dataset. These include the input information including their corresponding label, i.e. the class the input belongs to. The main procedure can be described by using the training data to set the weighted connections between neurons in a manner so that the network produces a correct output [31]. Subsequently, using the testing dataset on the training weights, errors between the network prediction and actual label are minimized to improve the network output. One process of feeding data in and then correcting the network to adjust to the found error is called epoch. The number of epochs performed during training can be set.

As a first step, the training dataset and the corresponding labels are fed into the neural network. The number of input data that are presented to the network at the same time is defined by the batch size. This parameter can be chosen to be equal or smaller than the total number of data in the dataset and is termed mini-batch in the latter case. All the mini-batches are passed through the network, therefore all training images are used per epoch, i.e. the number of times the whole dataset has been passed to the network and an adjustment had been performed. The weights of all connections are set in a manner so that the network outputs the correct label for each training input. After this is done, the testing dataset is passed through the network in the same manner regarding the batch size. The network is now configured to work well on the training dataset and predicts output labels for the testing dataset. The prediction is compared to the correct output label and an error is determined

and mathematically depicted as the loss function. This function is now optimized using one of many available optimisation algorithms. To adjust and correct the network, the weights of each connection are changed following the found gradient when applying the optimizer. Optimisation happens by adjusting the weights following the descending gradient of the error [40]. Error correction is done in a back propagating manner, first adjusting the weights leading to the output layer and then subsequently going back layer per layer, correcting all the weighted connections of the model.

Two parameters that can be monitored during the training are accuracy and loss. The accuracy describes the fraction of correct predictions made on testing data by the network, ideally converging to 1 with increasing number of epochs. The loss is the sum of all errors of the made predictions and should converge to 0.

After a model is trained, it may be evaluated using another unlabelled dataset. Upon passing this evaluation dataset to the network, it outputs the probabilities the input belongs to each of the labels. This shows the practical use and ability of the model to sort unknown images into the labelled groups.

1.2.1.3 Overfitting

One problem encountered with neural networks is the probability of overfitting. This describes the process of the network adapting too well to training data and achieving high accuracy during training. Since the network focuses too much on details it is not able to generalize parameters and patterns very well and therefore shows significantly reduced performance when unknown data is presented. There are two possible ways to counteract such a behaviour. Firstly, by adding a so-called dropout layer, a set percentage of training data is dropped randomly during training and is therefore not used for calculations [41]. Secondly, a regularizer may be implemented. Regularisation adds an additional term to the loss function, therefore keeping the weights from being adjusted too well to the details of the training data and achieving a better generalisation. Additionally, also pooling layers help to control overfitting by reducing the information and focusing only on main features.

1.3 Purpose of thesis

Current research in machine learning in the medical field is also pushing forward in the area of treatment plan evaluation [42]. For example, Interian et al. used a convolutional neural network to determine the gamma passing rate of treatment plans. This study served as motivation for this thesis, Interian et al used a dataset of 498 images, artificially increasing the data by using images several times, either by utilizing rotated or mirrored images additionally to the original. This is done with the assumption that the input images are rotationally invariant in regard to the GPR value. Different network types are tested on the dataset and then evaluated according to the percentage of correct predictions that were made, and then compared with a previously designed Poisson regression model, achieving similar results.

In short, the aim of this master thesis is to design a neural network that is capable of categorising radiotherapy treatment plans into groups according to their predicted gamma passing rate with sufficient accuracy. The GPR is obtained during the quality assurance measurement, which precedes each patient irradiation in order to ensure the safe execution of the treatment plan. The GPR acts as a criteria that has to be fulfilled in order to apply the plan to the patient. If it can be predicted with a sufficient accuracy using treatment plan data only, there is a possibility of omitting the QA measurement in future cases where neural networks provide reliable predictions of the GPR.

This thesis intends to use a similar approach as presented by Interian et al. and aims to achieve comparable or improved results [42]. The built neural networks should also be tested for their robustness and performance with varying parameters and datasets, respectively.

2 Materials and Methods

In this chapter, the hardware resources used for generating the information needed as well as the selection of patient data are described. Furthermore, the overall approach of generating the input data and the coding effort and workload used for this work are elaborated. The setup of the neural network is discussed, along with the complete layer structure and the used activation functions, optimisation processes and metrics. Finally, the different approaches of optimising the network are discussed.

2.1 Hardware

2.1.1 Linac

The investigated treatment plans were designed to be applied by two different types of linacs produced by Elekta, Sweden. The linac types include Elekta Versa HD and Elekta Synergy, differing in MLC types, i.e. in their number of MLC leaf pairs and the existence of jaw collimators in the x-direction. There are 80 leaf pairs in the Agility treatment head of the Versa linac and 40 in the MLCi2 of the Synergy linac, while the x-jaws are only present in the Synergy. For plans generated for Versa linacs, the values for the non-existing x-jaws necessary for computation were assumed as the maximum open position. The beam opening covers an area of 40 cm x 40 cm in both linac types, therefore the width of a single leaf amounts to 5 mm for the Versa and 10 mm for the Synergy linac.

2.1.2 Delta^4 Phantom

At the Department of Radiation Oncology of the Medical University of Vienna / AKH Vienna, a Delta^4 Phantom (ScandiDos, Uppsala, Sweden) is used to perform the quality assurance measurement. The phantom is equipped with a cylindrical shaped container which consists of PMMA containing two detector plates placed in orthogonal orientation relative to each other, forming a cross shape. The 1069 detectors distributed on those detector plates measure the deposited dose, which is

compared to the planned dose distribution in order to determine treatment plan accuracy.

2.2 Gamma Passing Rate

Upon quality assurance measurements of treatment plans, which are performed on a phantom in advance of each patient treatment, the parameter which is used to categorise input data is obtained: the Gamma Passing Rate (GPR) [14], [43]. This rate is a measure of agreement between the dose distribution that was calculated and planned before and the actual delivered dose distribution onto the phantom. For the data used in this work, these underlying criteria were set to 3% and 3mm, respectively [12]. Fulfilling these criteria leads to a gamma index of ≤ 1; the percentage of all the points exhibiting a gamma index of ≤ 1 then forms the GPR value. Therefore, a high accordance of the planned and measured dose distribution results in a GPR value close to 100%. This is usually achieved by relatively simple target geometries like prostate boost treatments. The GPR is actually a measure of dosimetric accuracy but can also be interpreted as a measure of complexity, since intricate target volume shapes usually exhibit a lower GPR.

A plan with a measured GPR lower than 90% must not be applied to the patient and requires the setup to be checked. In case no irregularities are found there, e.g. the detector and linac configurations are correct, the plan has to be redone in order to ensure safe treatment of the patient.

2.3 Patient Data

In order to train a neural network and to achieve good performance, a sufficiently large sample dataset is necessary. Therefore, VMAT radiation treatment plans created using the software Monaco (Elekta AB, Stockholm, Sweden) were exported and by using the parameters and values that are set in the planning data, images of the transmitted dose were created and used as an input for the neural network. Each patient is given a unique patient ID, which was used to identify plans throughout the process and is also encoded in the filenames of the images.

2.3.1 Selection

For this investigation a total of 600 IMRT/VMAT radiation therapy plans were exported, which were taken from three tumour groups: gynaecological tumours, head and neck (HN) as well as prostate cancer patients. An even distribution between the groups was desired, leading to total numbers of plans for each group presented in Table 1. The respective dose distributions are shown in Figure 7 in section 1.1.6.

Table 1: Numbers of plans per tumour group

Group	Number
Gynaecological	151
HN	227
Prostate	222
Total	600

While selecting prostate patients, boost-plans were used exclusively for better comparability, referring to the administration of a dose of typically 13-28 Gy to the prostate gland only, omitting the pelvis region and surrounding lymph nodes. This leads to a sharply delimited volume of dose deposition and therefore to a favourable target volume regarding accuracy and complexity.

The plans were designed for two different linacs, Elekta Versa HD and Elekta Synergy. The selection of plans did not depend on the linac type, since it is assumed that the different linacs do not produce different results, as for the Versa HD linac plans the missing x-jaws are replaced by the values for completely opened ones and can therefore be evaluated using the same tool as for the Synergy linac plans.

Further selection criteria include the completed QA measurement, i.e. the requirement that a GPR value has been obtained for the plan beforehand. Only the plans that had an associated GPR were used, since this value is needed for labelling the plans. To avoid differences in patient Dicom standards as well as updates or changes in data format or data storage modalities, it was aimed to utilize fairly recent plans. The acquired plans according to the presented criteria were generated in a timespan between 2015 and 2018.

2.3.2 Labelling

For the sake of describing the problem, a multi-class classification model was chosen and all plans were labelled according to their GPR value. As a majority of the used treatment plans showed a very high GPR value, grouping them into equally large groups to ensure validity of the results of the neural network proved to be a challenge. The prostate plans exhibit a GPR of 1, since the prostate gland is well delimited to its surroundings and is shaped rather simply and shows little to no variance of location in different patients. Therefore, they were all assigned with the label 0, indicating the highest GPRs. Intervals of GPR values that make up the labelled groups were to be chosen in a manner so that the numbers of plans of each label are as balanced as possible in order to avoid a bias towards a certain label in the model while still giving reasonable results. It was chosen to divide the input data into three labels according to their GPR according to Table 2.

Table 2: Distribution of all treatment plans to GPR values and consequent assignment of labels.

GPR	Number of Plans	Labels	Numbers per label
1	235	0	383
]0.99, 1[	148		
]0.98, 0.99]	75	1	130
]0.97, 0.98]	55		
]0.96, 0.97]	17	2	87
]0.95, 0.96]	21		
]0.94, 0.95]	13		
]0.93, 0.94]	11		
]0.92, 0.93]	6		
]0.91, 0.92]	10		
]0.9, 0.91]	4		
]0.89, 0.9]	2		
]0.88, 0.89]	2		
≤ 0.87	1		
Total	**600**		**600**

2.3.3 Dicom file structure and parameters

In order to generate images as data input for the neural network, treatment planning data is used. Medical data, as in the case of radiotherapy treatment plans, is stored in Dicom (Digital Information and Communications in Medicine) format, which can store image data as well as patient information. In this particular case – the Dicom variation for radiotherapy, Dicom RT– each file contains the complete treatment plan information, i.e. collimator positions and delivered dose, as well as patient data. The data is stored in form of attributes, which can be accessed via their associated tag ID. The extracted parameters include the Dicom tags noted in Table 3.

Table 3: Dicom tag IDs, names and descriptions [44]

Tag ID	Tag Name	Description
(300a, 00c0)	Beam Number	Number of the beams that are used to irradiate, equivalent to the number of arcs the gantry is performing.
(300A,0112)	Control Point Index	Consecutive numbers of control points along the gantry arc signifying points at which changes occur during the beam delivery, all the parameters below are associated to a certain control point.
(300A,0134)	Cumulative Meterset Weight	Fraction of total beam meterset that has been irradiated at the current control point.
(300A,0086)	Beam Meterset	Total amount of monitor units (MU) that are set to be delivered with the current beam.
(300A,011E)	Gantry Angle	Angle the gantry is positioned at, with 180° being the position where the gantry is in the most upright position.
(300A,011C)	Leaf Jaw Position	This entry appears two or three times per control point, corresponding to two or three different collimator types. The entries for the jaw collimators are described with 'ASYMX' or 'ASYMY', depending on the orientation, containing two values signifying the distance from the center to the two parts of each collimator. The entry for the MLC leaves includes the description 'MLCX' and contains either 80 or 160 values in a row, assigning each leaf a position given

		in mm distance from the origin. Indexing starts at the bottom left leaf and first covers all the leaves on the left, then continues at the bottom right (seen from BEV).

The gantry can perform a full 360° rotation around the patient couch and while there is a continuous irradiation, control points are set along the arc the gantry moves along. These can vary in number depending on the geometry and size of the target volume; furthermore, the gantry speed can be varied for additional dose modulation. In the Dicom file, the aforementioned parameters are defined for each control point. The single jaw and leaf positions are given in mm distance from the origin, making it necessary to compute the distance using the data from each corresponding leaf pair for further analysis. The origin of the coordinate system is positioned in the center of the beam.

2.4 Preparation of Patient Data

Patient treatment plans are archived in a raw text file format as the output of the treatment planning system (TPS), Monaco (Elekta, Sweden), and can be exported from the TPS as Dicom files. To avoid loading every single plan into the software and extracting it, the raw planning data was first transformed into Dicom format and then into images depicting the transmitted dose, the fluence maps. The used software resources and detailed transformations will be described in the following paragraphs.

2.4.1 Software

The necessary conversion of patient data to images was performed using several different software resources. In general, all coding operations were executed in Python, using PyCharm as integrated development environment (IDE). For transforming the patient data into structured XML (Extensible Markup Language) shape, the ElementTree Application Programming Interface (API) was used [45]. Additionally, further conversion into Dicom data format was achieved by using operating system functionalities, utilizing the os module [46] and the DCMTK toolkit

including the xml2dcm utility [47]. As a method to access the Dicom tags and read the Dicom files, the package PyDicom and more specifically the Dicompyler package was used [48], which is construed for dealing with the Dicom-RT format. This format is the extension of the Dicom type designed specifically for radiotherapy applications. Furthermore, the package NumPy was used for basic scientific computing and handling array objects [49], as well as Matplotlib as plotting library [50].

2.4.2 Conversion of Monaco Raw Data to Dicom

The transformation of the acquired raw plan data into Dicom files was the first step towards generating image data. The raw files were analysed and in a reverse engineering way, certain segments were characterized according to their values, which were compared to those of a corresponding exported Dicom file. Each Dicom checkpoint appears as a block of numbers in the raw data file, containing all the values for the MLC leaf and jaw positions. A characteristic starting sequence was identified, signifying the start of each checkpoint-block. However, the order of the leaf positions had to be changed in order to be consistent with the arrangement of values in the Dicom file, as the raw file listed the two positions of a leaf pair consequently, while the Dicom lists all the values of the left MLC leaves first, followed by all the positions of the right MLC leaves.

Furthermore, the values of the jaw collimator positions had to be manipulated. In the raw file, positions for each jaw pair were given using two values: the size of the opening, i.e. the distance between the two jaws, and the offset of the center of this opening with respect to the central axis, both in mm. This is depicted in Figure 14.

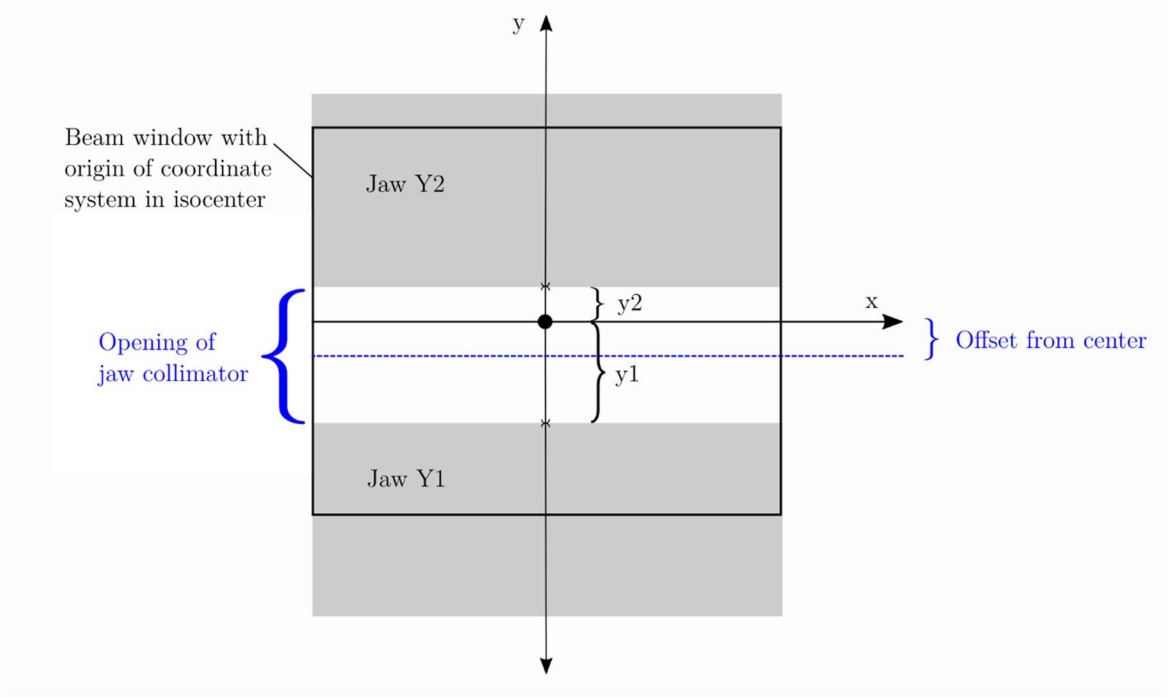


Figure 14: Schematic depiction of calculation of jaw positions y1 and y2 as present in the Dicom file using the values given in the raw plan file, opening size and offset (blue).

Therefore, the jaw positions for the y-jaw on the y-axis as given in the Dicom file can be calculated from those values according to the following equation. For plans designed for the Synergy linac, the positions of the x-jaw are obtained analogously, with $x_{1,2}$ representing the positions of the x-jaw along the x-axis, with x_1 describing the position of the left and x_2 the position of the right jaw, respectively.

$$y_{1,2} = offset_y \mp \frac{opening\ size_y}{2} \quad (1)$$

$$x_{1,2} = offset_x \mp \frac{opening\ size_x}{2} \quad (2)$$

To obtain the values for the cumulative meterset weight, a small computing effort had also been necessary. The total beam meterset can be obtained easily by parsing the raw file. Within the blocks containing the information for each checkpoint, a value corresponding to the dose that has been delivered between two checkpoints can be found. Therefore, the needed meterset weight, the fraction of dose that has been delivered between the preceding and the current checkpoint can be calculated using the following context.

$$Meterset\ Weight\ per\ Checkpoint = \frac{Delivered\ Dose\ per\ Checkpoint}{Beam\ Meterset} \quad (3)$$

As described in 2.3.3, the beam meterset parameter describes the total dose delivered, while the cumulative meterset weight gives the fraction of the meterset that has been delivered up until the regarded checkpoint. To achieve the same convention as in the original Dicom, the meterset weight obtained using (3) has to be summed up continuously for each checkpoint.

All the retrieved values are now arranged in a single ElementTree object in order to write an XML file, since it offers an intuitive way to incorporate a desired structure into a text file by defining elements and assigning correspondent sub-elements to mirror the Dicom file structure. The conversion from xml to Dicom format and vice versa is an easy task, accomplished by a simple operating system command.

The retrieved values for cumulative meterset weight deviated after the third decimal from the comparative values in the Dicom, as the value for the beam meterset did also differ. Therefore the Monaco-internal conversion to Dicom seems to have an influence on these values. For being used in the following tasks the obtained values were regarded as sufficiently accurate.

Using the values from the raw planning data file, a direct conversion to image would be possible instead of the intermediate step of generating Dicom data, but since this tool may be useful in future situations it was chosen to perform the two actions separately. Furthermore, creating two separate tools enables the generation of fluence maps from existing Dicom data.

2.4.3 Fluence Map Generation

After converting the complete patient dataset to Dicom files, the delivered dose and the collimator positions per checkpoint were read from the Dicom and used to generate an image output. The created image type is called fluence map, displaying the fluence of dose through every point of the beam window during a complete treatment. Making use of the previously described libraries and the tag structure of the Dicom file, the needed parameters could be accessed easily. The first action taken was to create an empty matrix with the desired dimensions of the output image. For

this, 400 x 80 pixels were chosen as uniform image size. The width of 400 pixels represents the size of the beam window, which yields 400 mm, therefore 1 mm of the beam window corresponds to one pixel. The positions of the collimators are given in sub-millimetre precision, but a resolution in the mm range was considered sufficient. The image height of 80 pixels was chosen according to the number of leaf pairs. Since the used linacs exhibit either 80 or 40 leaf pairs, the former was chosen and set as the dimension. In order to generate uniform output images, this image size was kept also for the Synergy linac with half the number of leaf pairs. In this case, two pixels correspond to the width of one leaf, while for Versa plans, one leaf is represented by one pixel. This leads to a reduced resolution in the fluence maps for Synergy plans, but was assumed to not influence the results since the fluence maps therefore mimicked the larger MLC leaf pairs found in the Synergy linac.

In order to fill the 400 x 80 matrix, the origin in the isocenter of the beam that is set to describe collimator positions in the Dicom is moved to the right upper corner. This is done in order to match the positional information of the collimators to the matrix indices and thus the pixels of the later generated image, i.e. equate the position in mm on the x- and y-axis to the matrix column and row indices. To move the origin, all x-values were shifted by +200 and the y values by -200, respectively. Since this conversion renders all y-values negative, the absolute value of positions on the y-axis was considered in order to fit the matrix index convention.

For generating an image, a few more complex computational steps were necessary due to the definition of the checkpoints in the Dicom. Since the irradiation is performed continuously throughout the treatment, the checkpoints only give fixed intermediate positions of the collimators. For acquiring the total transmitted dose for the whole treatment, the movement of the collimators has to be accounted for, so therefore for each computation step the differences in cumulative meterset weight and in the collimator positions of two checkpoints were considered. The assumed model to calculate the transmitted dose in between two checkpoints by means of one MLC leaf pair is depicted in Figure 15.

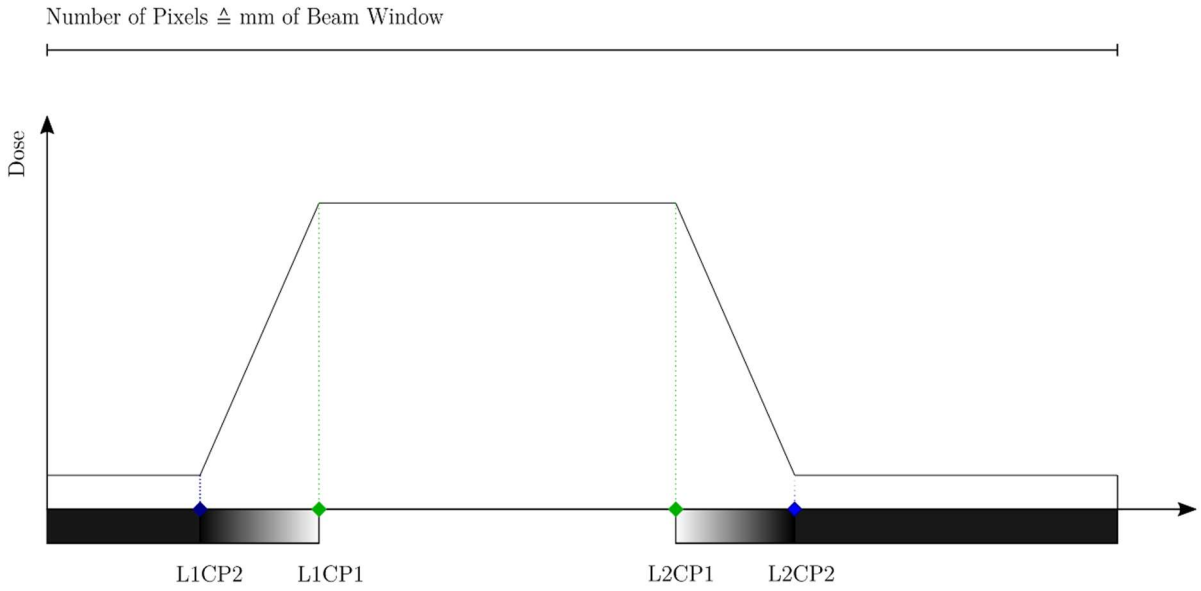


Figure 15: Schematic representation of the calculation of the transmitted dose with the aim of generating fluence maps. The bars below the diagram illustrate one MLC leaf pair at their positions along the x-axis within the beam window. L1 represents the left leaf, L2 the corresponding right leaf of the pair. Both are represented at two different checkpoints, CP1 (marked by green symbols) and CP2 (marked by blue symbols), moving from the position at CP1 to CP2.

The calculation requires the observation of two checkpoints at a time. Therefore, the dose delivered in the time between checkpoint 1 (CP1) and checkpoint 2 (CP2) is calculated using the cumulative meterset weights (CMWeight) of the two checkpoints as well as the total beam meterset:

$$dose = (CMWeight_{CP2} - CMWeight_{CP1}) * Beam\ Meterset \quad (4)$$

Then, the regions of different dose levels can be divided into five sections depending on the coverage of the beam. The left and right outer regions are completely covered by a MLC leaf during the time between CP1 and CP2. Since the MLC leaves do not shield 100% of the dose, a certain fraction of dose that is still transmitted has to be considered, as depicted in Figure 15. For this model, a transmission coefficient for the MLC of $t_l = 0.004$ was used. The opening between left and right leaf that is not covered at any time denotes the region where the full dose is transmitted. Lastly, the movement of the MLC leaf pairs is accounted for in the regions with varying dose. To describe the partially covered regions, a simple linear correlation of dose and

position on the x-axis x was assumed, with Δx denoting the difference in leaf positions of the two checkpoints, i.e. the distance the leaf has moved in mm. Regarding the shape of the MLC leaf tips, a simplifying assumption was made. The tips present in the linacs are rounded, whereas in this model, a rectangular shape has been used. This leads to a disregard of the parts of the incoming beam that are slightly oblique and would still contribute to the transmitted dose in reality.

Again, the fraction of dose that is transmitted through the partly closed collimator leaves described by the leaf transmission t_l has to be considered and leads to the following formula.

$$dose(x) = \frac{dose}{\Delta x} * x * (1 - t_l) + dose * t_l \tag{5}$$

$$x = \begin{cases} n - \min(L_1CP_1, L_1CP_2), & for\ L1 \\ \max(L_2CP_1, L_2CP_2) - n, & for\ L2 \end{cases} \tag{6}$$

Depending on the movement from left to right or vice versa, the dose shows either an increasing or decreasing slope, requiring a different definition of the parameter x. The column indices of the matrix are denoted by a single parameter n, which is used for the whole beam window. Since the filling of the matrix happens from left to right, an increase of n is inversely correlated to the dose transmitted by movement of the right MLC leaves. To describe the different behaviour, the minimum or maximum positions of the leaf at the two regarded checkpoints (L_iCP_i) are considered, respectively.

An even finer distinction of regions has to be made once the jaw collimators are considered as well. The calculation of transmitted dose follows the same mechanism, using $t_j = 0.001$ as jaw transmission coefficient. Furthermore, the width of the MLC lamellae has to be taken into account when translating the positions of the y-jaw along the y-axis to rows of the matrix. Depending on the used linac and MLC type, either 5 or 10mm correspond to one row and therefore a width of one pixel. Achieving uniform images of the same size as output required the duplication of each row for plans designed for the Synergy linac which incorporates only 40 leaf pairs, doubling the number of rows to 80 to match the Versa linac plans. Making the transmitted dose dependent on the x-jaw, y-jaw and the MLC positions and using the according

dose values obtained by the model described before, looping over all the checkpoints of all used beams fills the matrix.

After completing the matrix using the full planning data, it is converted into a greyscale PNG image with a simple Python command. This conversion lead to an addition of a white frame around the actual matrix size, resulting in a final image size of 526 x 127 pixels. The name of the saved image is put together by the patient ID and the label, which is retrieved from an additional text file that lists all the regarded patient IDs and their corresponding GPRs. A few examples of the image output are given in the following figures.

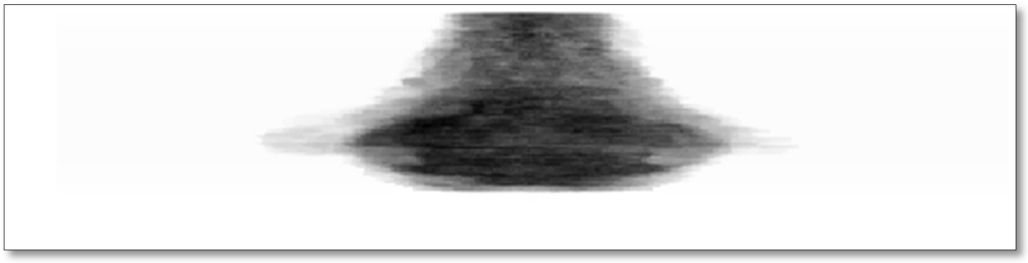
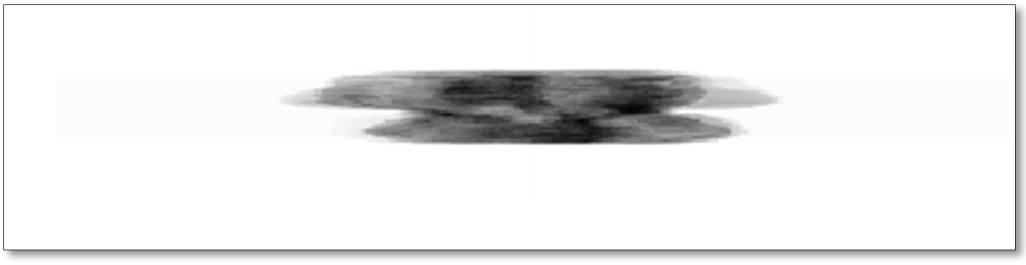


Figure 16: Fluence map of a gynaecological treatment plan.

Figure 17: Fluence map of a prostate boost treatment.

a)



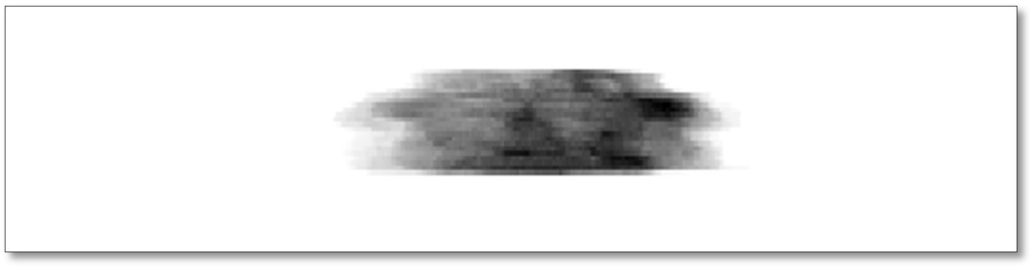
b) 

Figure 18: Fluence maps for 2 different HN plans. Image a) shows a plan for a treatment performed on a Versa linac, while image b) depicts the fluence map for a plan performed on a Synergy linac. The reduced resolution in b) is evident, originating from the larger width of MLC leaf lamellae.

The fluence maps portray the fluence of dose through the beam window during a whole treatment. The darker areas show regions of higher dose deposition, while white areas do not receive any dose. It is evident that fluence maps of prostate boost plans are easy to distinguish from HN and gynaecological plans by their smaller irradiated volume and simpler geometry and dose distribution. Due to the former described selection of image size the fluence maps appear pinched, as the length one side of the beam window is decreased, leading to a rectangular representation of the square beam window.

2.4.4 Random separation into three datasets

The last action performed on the input data before it was used to train the neural network was to randomly shuffle the sample order and subsequently splitting the complete dataset into three groups – the training, testing and evaluation dataset, each containing 200 plans. For that, a text file was created combining the complete data paths of all the generated images to their labels. After shuffling the lines of that list, it was separated into three different text files, serving as an input to determine the three different datasets for the neural network. All the datasets were checked for relatively balanced distribution of the three labels, in order to avoid generating a dataset with only one prominent label, which would create a bias of the neural network towards one label.

Table 4: Distribution of the labels and tumour groups to the three datasets

	Set 1	Set 2	Set 3
Prostate	64	75	83
HN	79	78	70
Gynaecological	57	47	47
Total	**200**	**200**	**200**
Label 1	119	127	137
Label 2	48	44	38
Label 3	33	29	25
Total	**200**	**200**	**200**

2.5 Neural Network

After generating the input data, a neural network was set up using [51] as template, adapting the suggested neural network to suit the task presented in this work. The mentioned Github repository introduces a basic network structure and presents a strategy to transform image input data into usable tensors, which is also adjusted to the generated fluence maps.

2.5.1 TensorFlow

The neural network was built in Python using the open source framework TensorFlow developed by Google [52]. It provides Python and C++ APIs and includes simple functions to arrange the layers of the network, network parameters and layer details and perform mathematical operations to generate output data. TensorFlow supports high level interfaces like Keras and can run on a CPU (central processing unit) as well as multiple GPUs (graphics processing unit). As suggested by the name, TensorFlow performs all the operations on multidimensional data organised as tensors. The network models generated in TensorFlow are organised as graphs following the graph theory of mathematics, consisting of so-called nodes and edges which represent the mathematical operations and the connecting dataflows, respectively. After the building of the graph is complete, the execution happens only in form of a session, during which the previously defined functions and operations are called.

2.5.2 Visualisation using TensorBoard

TensorBoard is a collection of tools able to visualise the built graph in a TensorFlow procedure as well as displaying images that are passed through the network and illustrating obtained scalar variables [53]. In order to select which nodes or values should be displayed, summary objects have to be added during the model build, which gather the selected values during the training process. This data is collected in TensorBoard event files, which can then be visualised in the web browser.

The complete graph is shown in Figure 19. The boxes correspond to the nodes and therefore to collections of operations combined in one namespace, while the edges are shown as arrows, depicting dependencies and dataflows. Auxiliary nodes, i.e. nodes of a high degree with many connections to other nodes (e.g. the used optimiser) are displayed in the upper left corner to keep the diagram arranged clearly.

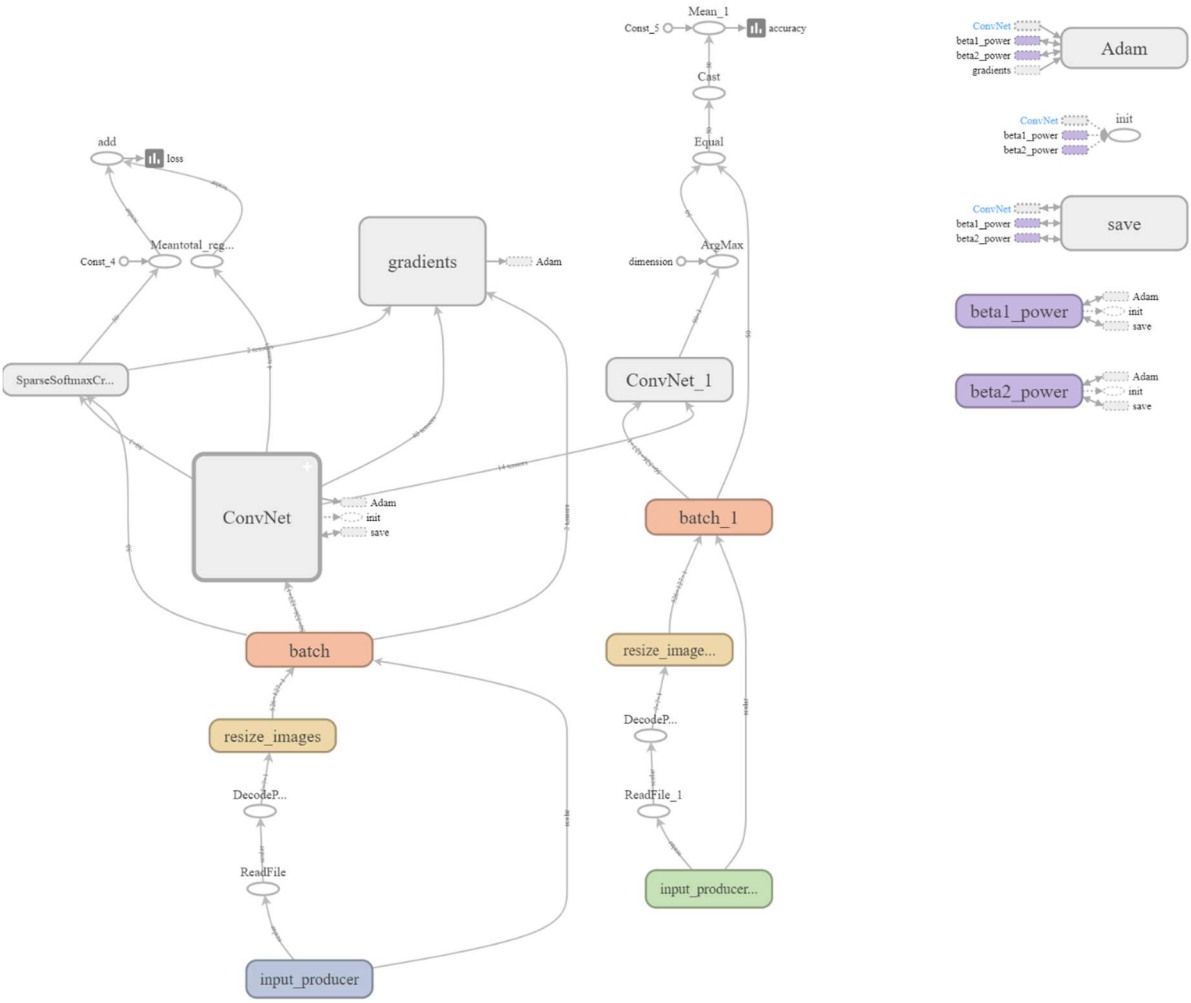


Figure 19: TensorBoard visualisation of the complete model, data flows of tensors are shown with arrows and operations as boxes.

In order to debug and analyse the neural network, TensorBoard can display the inputs and outputs of every node, also showing the single operations performed inside of larger, more complex nodes.

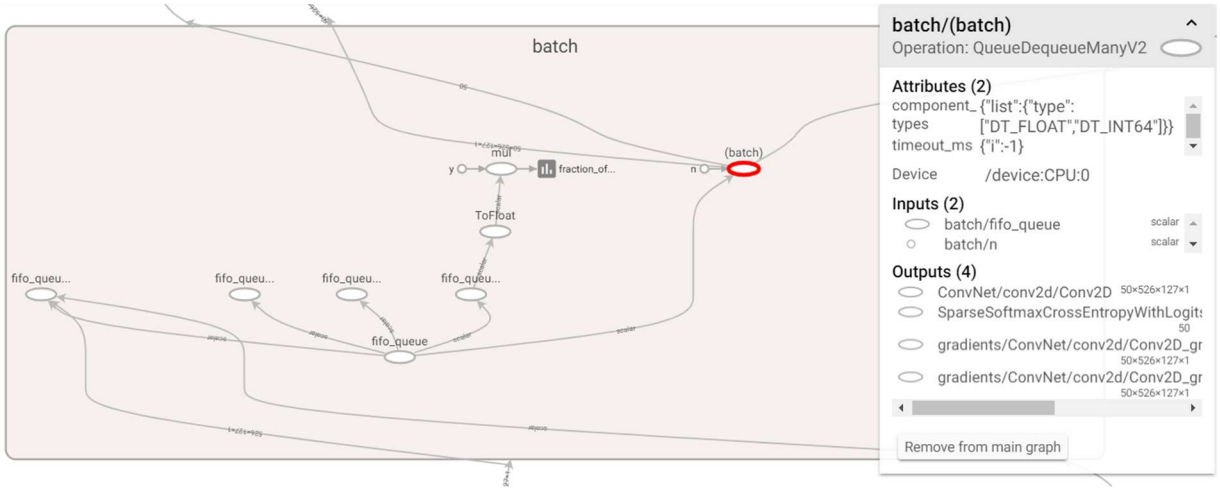


Figure 20: Display of an expanded node in TensorFlow, providing information about the inputs and outputs of the selected node.

2.5.3 Used Mathematical Operations in Neural Network

Loading the input data into the model can be realised using two different approaches. Firstly, the input images may be organised in three folders according to their labels. During loading of the data into the network, two tensors are built, one containing the image data paths while the other lists the labels corresponding to the folder the images are in. The other method, which was chosen for this work, involves a file which lists the data paths of the input data and their corresponding label. Upon reading the file, the paths and labels are separated and assigned to separate tensors. Both strategies now convert the PNG image data into uint16 tensors, i.e. the values of the tensors are integer values with a size of 16 bit. The number of colour channels is set to 1, since only greyscale images are regarded. The size of the tensors amounts to 526 x 127 x 1, representing the greyscale 2D images.

As activation function of the neurons, the previously discussed ReLU was used. This function sets negative activations to zero and outputs a linear function if there is a positive input.

$$f(x) = x^+ = max(0, x) \tag{7}$$

Therefore, the output increases with the positive input leading to neurons reacting stronger to larger input values and enabling the calculation of more complex, non-linear models.

After each training cycle of loading training data into the network and setting the weights, a prediction of the label of the subsequently fed in testing data is made using the previously determined weights. After the last fully connected layer, i.e. the output layer, the softmax function is applied to determine the probabilities of the tested image belonging to one of the possible labels. The softmax function converts an n-dimensional vector termed logits, which is received as input from the previous network layer, to a decimal probabilistic value for each of the possible classes. The output is therefore a scalar value between 0 and 1, the sum of the probabilities for all labels naturally yielding 1.

Further, the loss function (generally termed cost function) which describes the difference between estimated and actual label is determined by calculating the cross entropy. The cross entropy serves as a method of obtaining the difference between the probability distribution given by applying the softmax function on the output logits, i.e. the output of the neural network, and the actual distribution of labels. For two discrete probability distributions p_i and q_i the cross entropy is given as follows.

$$H(p, q) = -\sum_i p_i log(q_i) \tag{8}$$

The last performed step before the optimisation of the loss function is to determine the mean of the acquired cross entropy tensor, which is reduced to one single element in the process. This obtained value is later used as the scalar value determining the loss and should converge to 0 during the training process. The optimisation of the loss function is performed using the Adam (Adaptive Moment Estimation) optimiser [54]. This metric combines the benefits of two other optimisers, Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) and serves as an extension of the conventional stochastic gradient descent (SGD). In contrast to SGD, which uses a fixed learning rate, Adam determines a learning rate

for each parameter i.e. each weight separately and adapts the learning rates according to the found gradient of the loss function. Contrary to other optimisation techniques, Adam uses estimations of both the first and second stochastic moments of the gradient, corresponding to the mean and uncentered variance of the gradient. A hyperparameter that can be set is the step size α, indicating the size of the steps that are taken along the gradient in order to find a minimum value. It is generally also referred to as learning rate but has to be distinguished from the individual learning rates set for each parameter when describing the Adam optimiser. However, since the framework TensorFlow denotes α as learning rate, this convention will be kept for further investigation performed in this thesis.

As described in 1.2.1.3, complex models may learn details of the training data and are therefore prone to overfitting. To avoid that, weight regularization was applied in order to keep the model simple and therefore improving the generalisation of features. This was done by adding an L2 regularizer to each convolutional layer, which retrieves the squared feature weights of each layer. All these values are summed up, multiplied by a factor λ called regularization parameter, and added to the loss function, serving as method to consider the complexity of the model [55]. The loss function to be optimised has now a shape as shown in (9).

$$minimise\big(Loss(Data|Model) + \lambda * complexity(Model)\big) \tag{9}$$

This additional term penalizes large weights in the model and therefore regulates the distribution of weights, reducing large positive and increasing small negative outlier weights. The loss function includes the additional L2 term, therefore the optimisation compensates the complexity of the model, therefore counteracting overfitting. The regularization parameter λ determines the fraction of the L2 regularization term that is added to the loss function and is set to 0.01 for this work.

$$L2\ regularization\ term = ||\boldsymbol{\omega}||_2^2 = \omega_1^2 + \omega_2^2 + \ldots + \omega_n^2 \tag{10}$$

In order to quantify the results of the training, the parameters accuracy and, as previously described, the loss are calculated. The accuracy is determined by obtaining the ratio of correct to total number of predictions made by the network.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Number\ of\ Examples} \tag{11}$$

In the network this is realised by applying the argmax function to test logits, i.e. the vectors obtained in the last fully connected layer, hereby – in contrast to the softmax function – only obtaining the labels with highest probability. Comparing the result to the actual labels of the test dataset yields a tensor of Boolean values, therefore determining the number of correct predictions. Similarly to obtaining the scalar value for the loss function, the reduced mean of the tensor of correct predictions is obtained in order to describe the accuracy.

2.5.4 Terminology of Neural Networks

In order to provide a summary of the terms and definitions used in the previous sections, the following table provides short descriptions of the parameters and properties of neural networks in respect to TensorFlow terminology [56].

Table 5: List of used terminology of neural networks.

Accuracy	Fraction of correct predictions made by the network
Activation function	Mathematical function describing the activation of the neurons in the network, i.e. how the input values of each neuron relate to the output
Batch	Number of input data fed into the network at a time, if the batch size is smaller than the total number of input data it is termed mini-batch, often a factor of the total number of inputs is used for a faster computation process
Convolution	Mathematical operation performed in the convolution layer in order to reduce the number of parameters which allows faster computation and more complex networks without reducing performance
Dropout	Method to reduce overfitting, deliberate deactivation of a certain fraction of neurons to improve generalisation of network
Epoch	Process of all training instances passing through the network once. One epoch consists of the number of iterations it takes to pass the whole input dataset through the network once, i.e. number of input data divided by batch size

Fully Connected Layer	Layer in which all the neurons have connections to all the neurons of the previous layer; performs classification of data
Feature	Elements of input vectors, i.e. number of neurons of the first layer
Graph	Complete structure of neural network
Hyperparameter	Property of network model that is to be set manually
Iteration	Training step; passing of one batch to the network and setting the weights according to the gradient
Label	Output of network, group the input belongs to
Learning rate	Size of steps taken along the gradient of the loss function in order to reach minimum
Logits	Output of last layer of the network, non-normalised vector of predictions made by the network
Loss (Cost) function	Main objective to be optimised during training by following the gradient of the loss function along its descending slope to find a minimum
Max Pooling layer	Layer of neural network that selects the most active neuron in a cluster of chosen size in order to improve computation time and reduce overfitting by focusing on the most relevant features
Model	Neural network after training, can be used to evaluate unlabelled input data
Node/ Neuron	One computational unit in the network, organised in layers, connected to other neurons
Optimisation function	Algorithm used to minimise/ optimise the loss function by adjusting the weights of the neurons
Overfitting	Process of very flexible adjustment of network to the training data leading to worse generalisation and therefore worse performance on unknown data
Parameter	Variable of network model, to be set during training by the network itself, i.e. weights
Regularization	Another method of avoiding overfitting by adding an additional expression to the loss function in order to improve generalisation of model
Session	Object that runs the built TensorFlow graph
Step size	Equivalent to learning rate
Training	Process during which the weights are adjusted in a way that the output of the network model matches the know input label
Weights	Parameters of the connections between neurons, to be set during training

2.5.5 Structure of network

In order to represent the final used layer structure of the neural network, Figure 21 shows the complete network including all data flows as well as the properties of the layers.

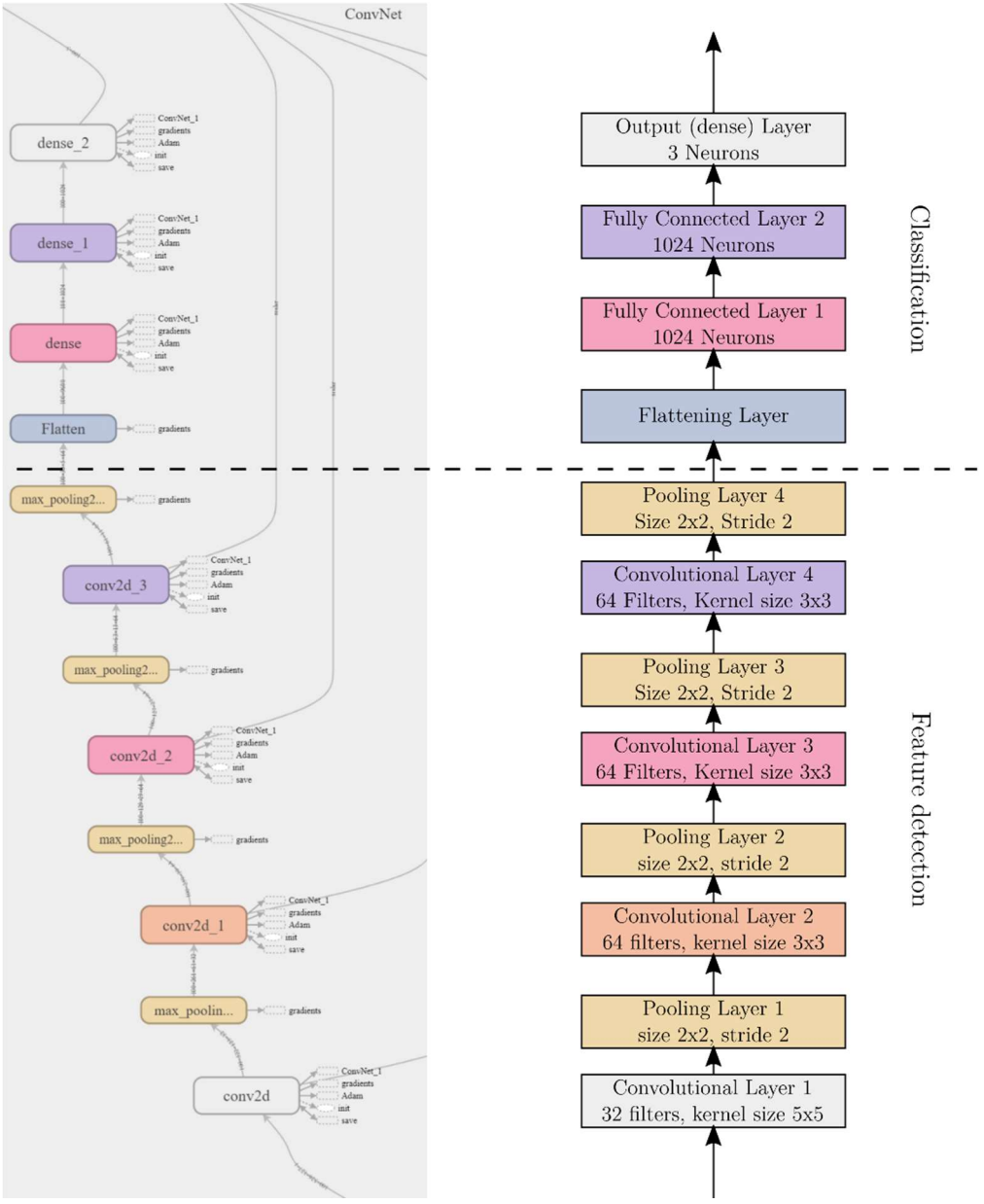


Figure 21: Structure of the used convolutional neural network as portrayed by TensorBoard (left) as well as the equivalent structure presented as boxes with description for better visibility (right). The arrows show the order the layers are passed, starting at the bottom with the first convolutional layer and ending at the last fully connected and therefore the output layer containing three neurons.

The function of the layers can be divided into feature detection and classification, the former accomplished by the convolutional and pooling layers, and the latter done by the flattening and the fully connected layers.

As depicted in the figure, the network consists of four convolutional layers, each followed by a pooling layer. The number of filters or kernels used amounts to 32 for the first layer, using a kernel size of 5x5. The input of this layer is formed from the pixels of the actual input image. The other three convolution layers make use of 64 filters, with a size of 3x3 each. The stride was kept at the default value for all the convolution layers, causing the filter to move one pixel or neuron at a time. Each convolution layer is followed by a pooling layer utilizing the max pooling technique, therefore selecting the neuron with maximum activity within a selected field size. These four layers exhibit the same parameters, choosing a field size of 2x2 and a stride of 2.

For executing the classification process, a flattening layer organises the retrieved neurons into a one-dimensional vector in order to serve as input for the following three fully connected layers. The first two of these consist of 1024 neurons, each connected to all the neurons of the previous layer, hence the nomenclature. The last of the fully connected layers serves as the output layer and consists of only three neurons, the same number as labels are present in the model. The values retrieved in this layer are the non-normalised predictions of the probabilities of the input data belonging to each of the labels. This output can be seen as a vector termed logits and is used to obtain the probability distribution.

The structure as it is elaborated here was obtained after testing and improving the original neural network, which is to be described in the following sections. The performed optimisation can be thematically divided into three topics: the optimisation of the network itself by adding more layers, the optimisation of parameters used during training and the testing of robustness of the network against different input data.

2.6 Optimisation of Arrangement of Network Layers

One strategy to evaluate improvements was to increase the depth of the network, i.e. adding more convolutional and fully connected layers.

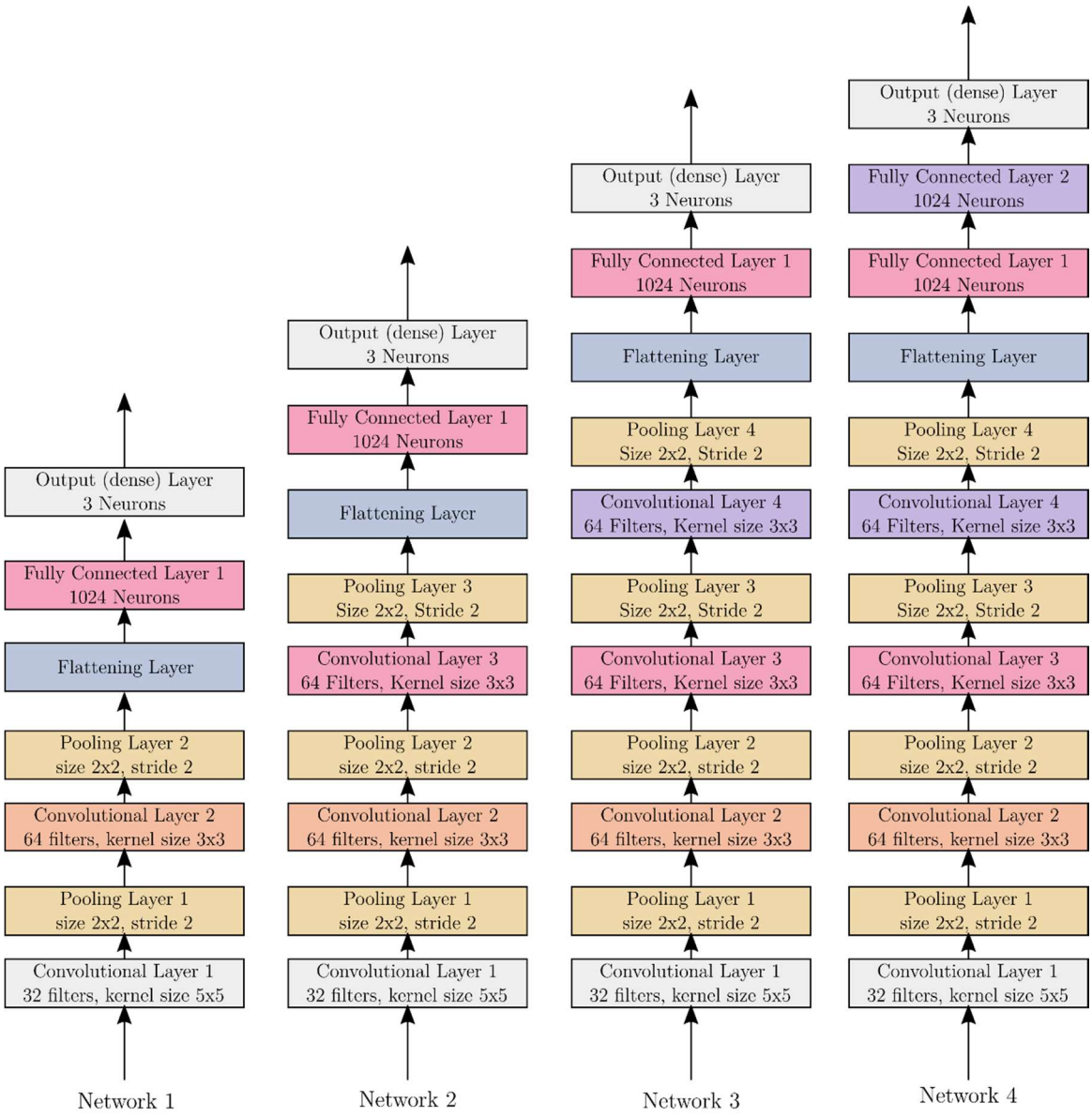


Figure 22: Different tested layer sequences. Network 1 represents the same network as suggested in the template, successive addition of convolution and fully connected layers lead to Network 4, which is the final used structure.

Usually, adding convolutional layers improves the accuracy of neural networks as more features are extracted while keeping the computation time rather low. This has a limit, since at a certain point the network will tend to overfit the data since detailed

features are detected by the convolutional layers. Four different layer sequences were tested, depicted in Figure 22.

2.7 Optimisation of Training Parameters

In order to further improve the obtained model and check the response of the model to different parameter changes, selected training hyperparameters were varied in order to identify optimal values and achieve sufficiently accurate results. These include the learning rate and the batch size. Other network parameters that were not changed over the course of the training processes are the number of epochs as well as the regularization parameter.

2.7.1 Learning Rate

The learning rate is a parameter that describes the step size that is taken along the gradient in the direction of the steepest descent during backpropagation in order to correct the network weights and determine the minimum of the loss function. In case of the used optimiser in this work, the learning rate set upon calling this function is a hyperparameter describing only the step size, since the learning rate of each parameter is adjusted separately, as described in 2.5.3.

A very small learning rate causes the weights to be adjusted very slowly, therefore increasing the time until a minimum is reached. On the other hand, a large learning rate takes steps that may be too large and therefore never reach the minimum, as illustrated in Figure 23.

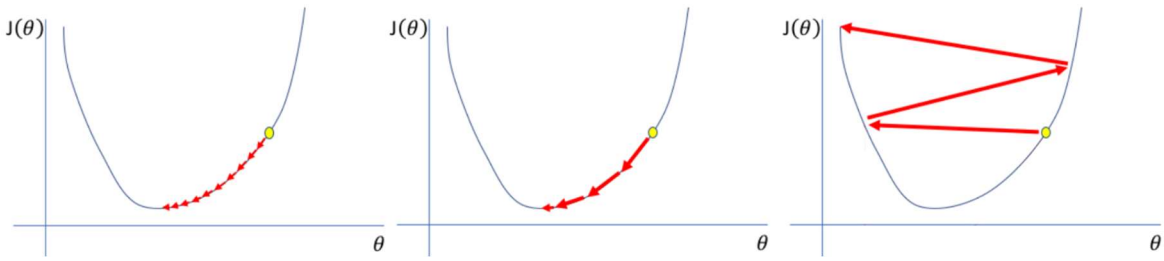


Figure 23: Depiction of the learning rate as the size of steps that are taken along the found gradient. The curve on the left shows a very small learning rate while the curve in the middle presents an optimal value and the right curve depicts a learning rate that is too large [57].

For this investigation, the values for the learning rate were varied from the recommended default value of 0.001 to 0.01 and 0.0001.

2.7.2 Batch size

As previously described, the batch size determines the number of input images that is passed to the network at once, influencing the gradient of the loss function that is obtained after one iteration. The gradient that is formed is averaged over all the predictions made, therefore, if more predictions are made, i.e. more images are passed through at once, the gradient is more precise. However, it has been shown that performance actually decreases when using large batch sizes due to worse generalisation, therefore it was assumed that the used amount of images per batch was sufficient [58].

In order to compare the performance of the model with different batch sizes, mini-batches of different sizes were built, using 50, 100 or 150 images per batch. Since the total dataset size is 200, the last case performed only one iteration since there is a residual amount of 50 images. One iteration describes the pass of one mini-batch to the network and one backward pass of weight correction. The relation of iterations and epochs, i.e. the cycle of passing all the input data once and setting the weights accordingly, is given as follows.

$$1\,Epoch = \frac{Number\ of\ input\ instances}{Size\ of\ mini-batch}\ Iterations \tag{12}$$

2.8 Testing of Robustness of Training

An important way of determining validity of a model is to test it for its robustness. To do that, two different approaches were taken: firstly, the three different datasets were given to the network in different combinations. Secondly, the datasets itself were shuffled again and compared.

2.8.1 Robustness against Switching of Datasets

One robustness testing involved the switching of the three prepared datasets, which are simply enumerated for the sake of a general view.

Table 6: Combinations of the three obtained datasets

Combination	Training dataset	Testing dataset
1	Set 1	Set 2
2	Set 2	Set 1
3	Set 3	Set 1
4	Set 3	Set 2
5	Set 1	Set 3
6	Set 2	Set 3

2.8.2 Robustness against Shuffling of Datasets

In order to investigate if the order of the list objects within the datasets has an influence on the training, the before investigated combinations of datasets were tested again after using a short script to shuffle the lines within the files containing the image paths and labels.

Table 7: Combinations of shuffled datasets in order to test robustness

Combination	Training dataset	Testing dataset
1	Set 1 shuffled	Set 2 shuffled
2	Set 2 shuffled	Set 1 shuffled
3	Set 3 shuffled	Set 1 shuffled
4	Set 3 shuffled	Set 2 shuffled
5	Set 1 shuffled	Set 3 shuffled
6	Set 2 shuffled	Set 3 shuffled

The described models were compared with the corresponding models from the previous robustness test.

The trainings were performed on an Acer Aspire E5-575G laptop based on an Intel® Core™ i7-7500U CPU with 2.90GHz and 8GB RAM. Additionally, an NVIDIA GeForce 940MX graphics card is available as GPU.

3 Results

After setting up the neural network as stated, the described variations of parameters and layer sequences as well as the robustness tests were performed in order to design an algorithm that is able to classify the input data accordingly and with the highest possible accuracy.

The epochs performed during each training were set to 500, i.e. the whole dataset is passed to the network 500 times, with a consequent backpropagation calculation and adjustment of weights after each forward pass. Another parameter that has not been varied is the regularization parameter α, which was set to 0.01 for all the performed network training processes. For the models built for this thesis it was chosen to display the value of accuracy and loss of every 10^{th} epoch using TensorBoard. TensorBoard provides a slider to adjust smoothing of the obtained curves to improve visibility of the trend of the graphs. The transparent curves display the actual obtained values, while the lines with high opacity represent the smoothed values.

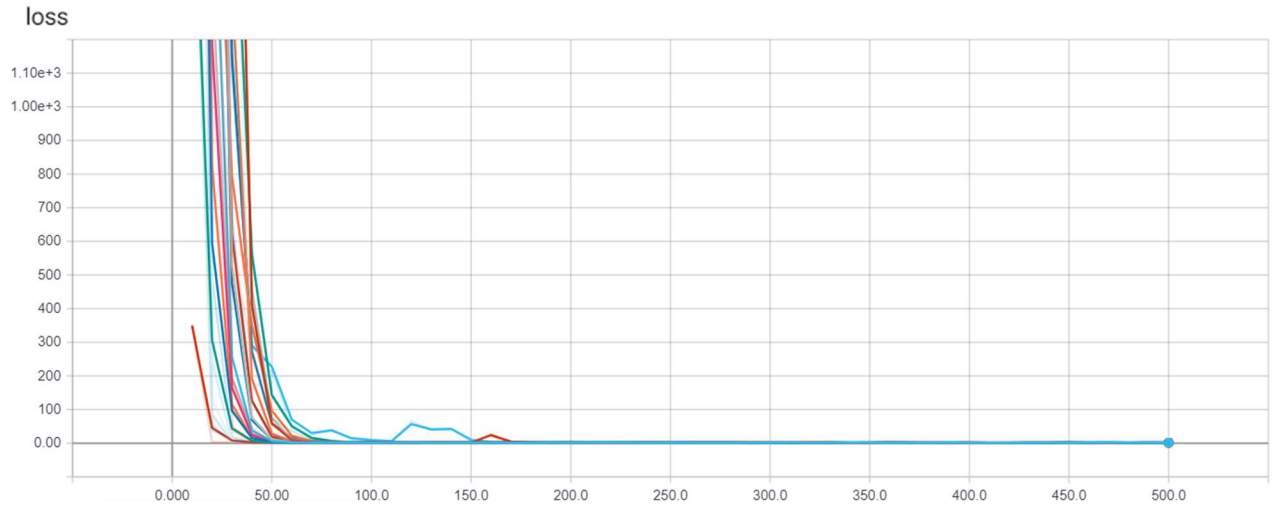


Figure 24: TensorBoard visualisation of the loss for multiple different network models. The x-axis represents the numbers of epochs while the y-axis shows the value of the loss.

The scalar loss values obtained as described before are displayed in Figure 24. Since the value showed the desired behaviour of converging to zero rapidly in all performed trainings, this parameter is not further discussed in the following optimisations.

3.1 Network Layer Structure

In order to determine the optimal layer structure, more convolutional and fully connected layers were added to the basic structure, leading to the results presented subsequently. All network trainings were performed using a learning rate of 0.001 and a mini-batch size of 100.

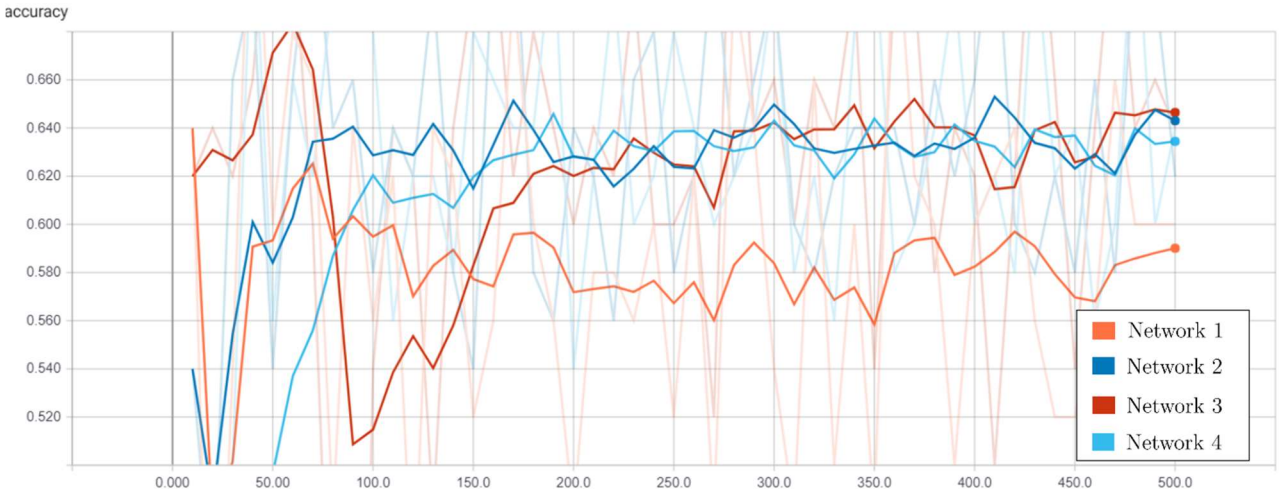


Figure 25: Achieved accuracy of four different layer sequences, represented as training accuracy over number of epochs.

Network 1, which was suggested in the used template, contains the least amount of layers and achieves the lowest accuracy of ~59%. Adding a third convolutional layer increases the achieved accuracy substantially to just over 64%. No changes were observed upon adding a fourth convolution layer and a third fully connected layer, however the layer sequence of Network 4 was kept for all further training processes, since no consequential increase of computation time was monitored (see 3.7).

3.2 Learning Rate

The first hyperparameter to be varied was the learning rate. As shown in Figure 26, the accuracy of the model varies considerably with changing learning rate. The lowest tested value was 0.01, exhibiting the predicted behaviour of not converging to a minimum, since the taken steps are too large. Reducing the learning rate to 0.001 already leads to a relatively fast convergence to an accuracy of around 64%. Further reducing that value to 0.0001 yields only a small increase of accuracy. However, again the observed computation time did not increase substantially.

Figure 26: TensorBoard graph displaying the training accuracy for different learning rates over the number of epochs.

3.3 Batch Size

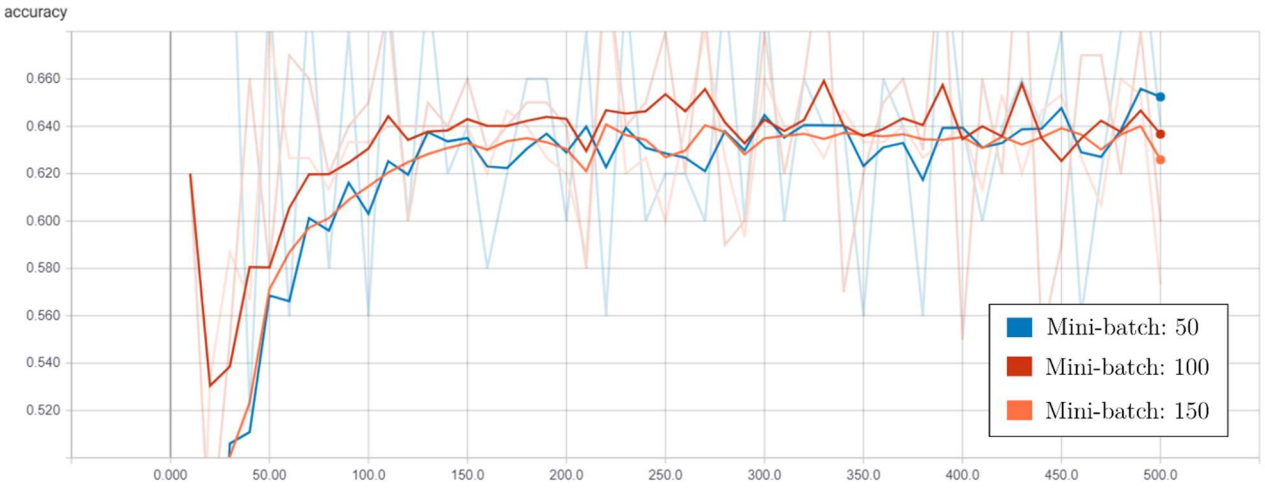


Figure 27: Training accuracy of trained model for three different mini-batch sizes.

Three different models were trained using three different batch sizes, i.e. 50, 100 or 150 images per mini-batch. The obtained curves all converge similarly to a value between 63% and 65 %. While a batch size of 150 produces the smoothest curve, the achieved accuracy values for the smallest batch size of 50 vary to a larger extent, causing the graph to appear less smooth especially for the first 150 epochs.

3.4 Robustness against Switching

To test the robustness of the model found in the last optimisation steps, the three datasets were passed to the network in different combinations according to 2.8.1. This test was again performed using a learning rate of 0.001 and a batch size of 100.

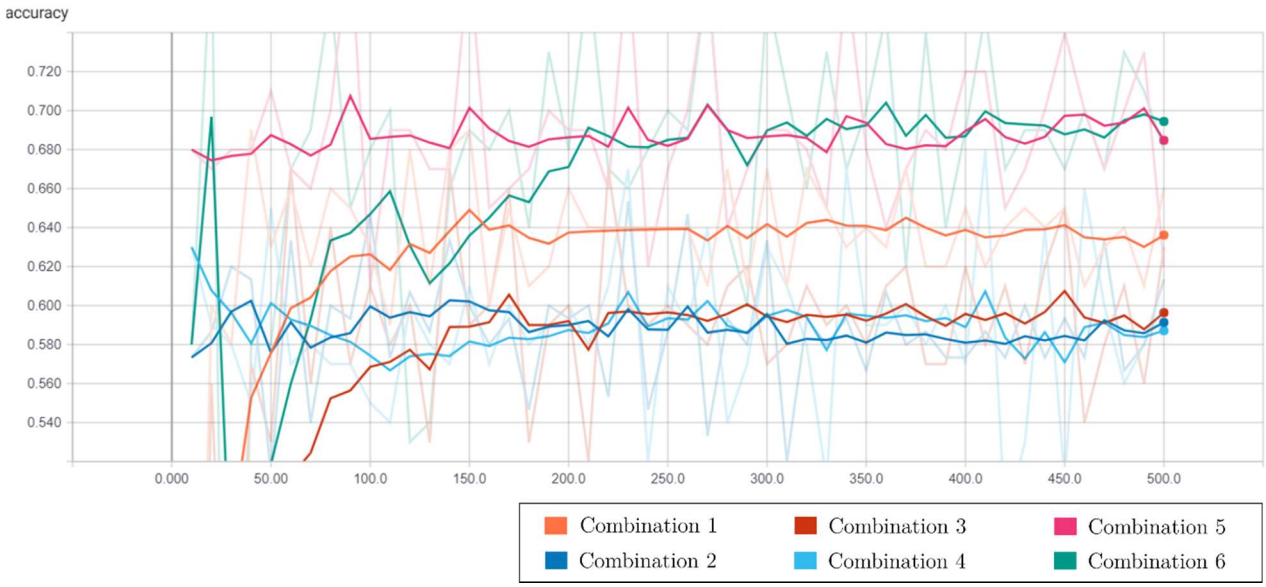


Figure 28: Accuracy of networks for six different combinations of datasets, given as training accuracy over number of epochs.

The hereby achieved training accuracies reach from ~59% to ~69%, therefore a large variance can be observed. The resulting curves seem to follow three different branches, converging either to a value below 60% for combinations 2, 3 and 4, around 64% for combination 1 or ~69% for combinations 5 and 6.

3.5 Robustness against Shuffling

The second robustness test performed was to shuffle the datasets and compare those to the same unshuffled combinations from the previous testing. Again, a learning rate of 0.001 and a batch size of 100 images was used.

Figure 29 and Figure 30 show the first three and the last three combination in comparison with their shuffled variations, respectively. Overall, the shuffled combinations yield similar values as the non-shuffled versions, following the observed

three branches. A deviation can be observed in combination 4, where the shuffled version provides an increase of accuracy of around 4%.

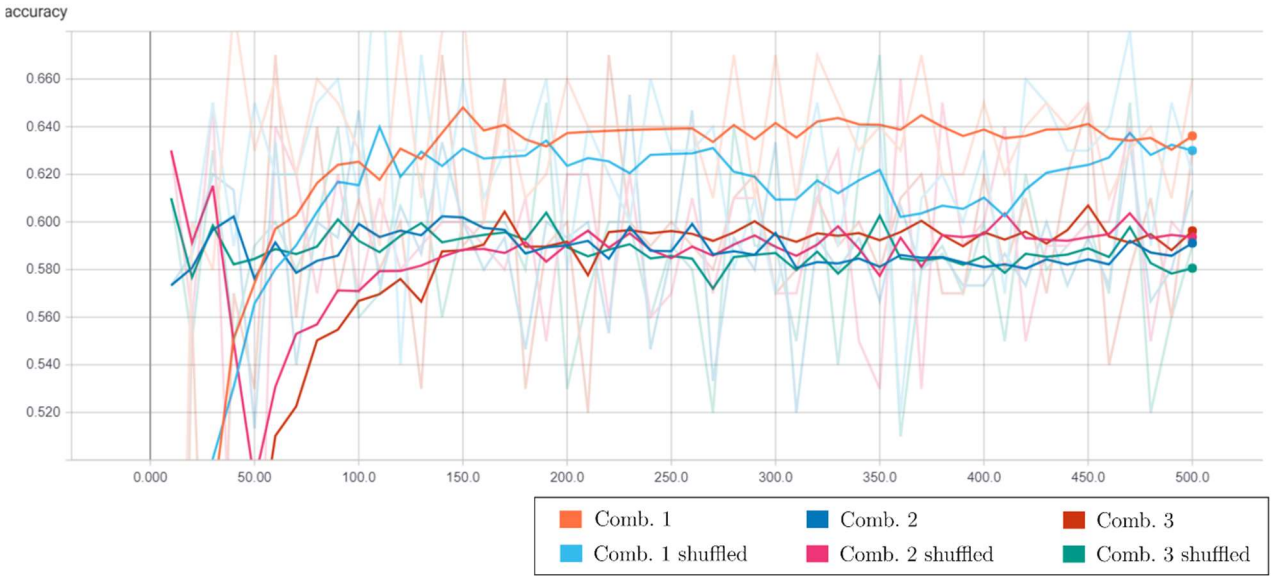


Figure 29: Training accuracy of networks over number of epochs to compare original datasets with shuffled ones.

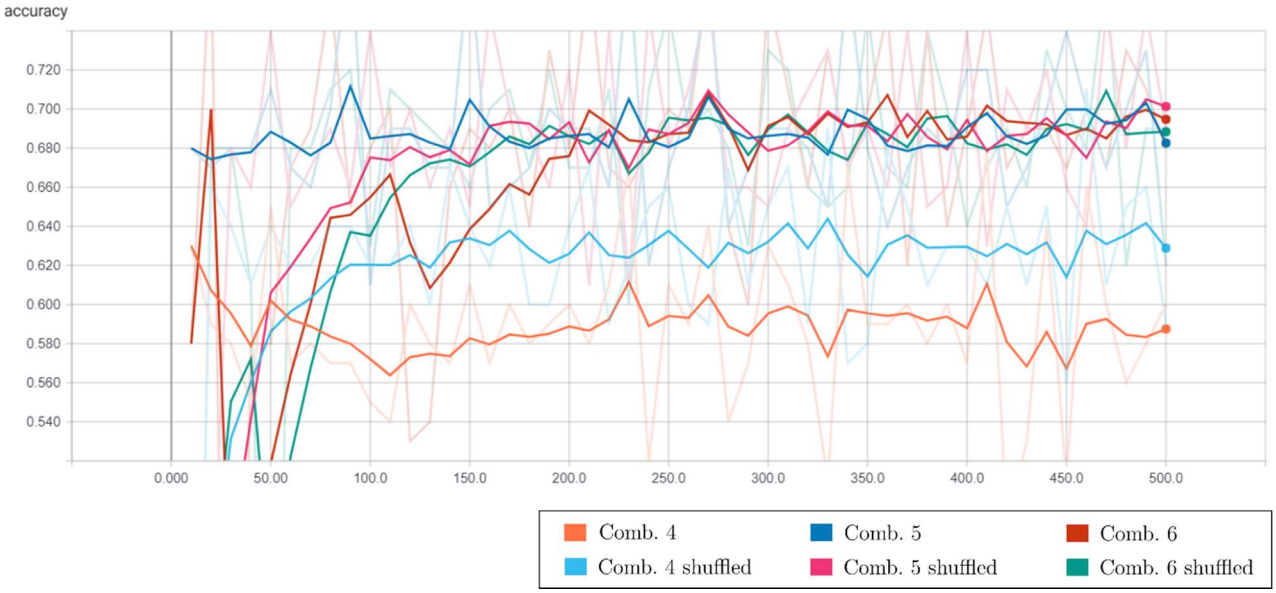


Figure 30: Training accuracy of networks over number of epochs to compare original datasets with additionally shuffled ones.

3.6 Evaluation of unknown data

The model delivering the best results as presented before consists of four convolutional and pooling layers, three fully connected layers, with the learning rate set to 0.001 and a mini-batch size of 100 images. Upon using dataset 2 and 3 for

training, an accuracy of 69% is reached. The final operation performed on the completed model was the classification of unknown data. For this purpose, the evaluation dataset, i.e. one that has not been used during training is passed to the best performing model described before. In contrast to training, the weights are now fixed and only predictions are made. During this process, an evaluation accuracy of 59.5% was achieved.

3.7 Computation times

Since no extensive increases in computation times upon changing parameters while performing network trainings on the CPU were observed, models were not optimised to that effect. For the sake of completeness, all runtimes are listed below. Calculation times ranged from 51 minutes and 26 seconds for Network 3 to 2 hours, 39 minutes and 35 seconds for the largest tested mini-batch size of 150 images.

Table 8: List of the computation times of the respective trained models

Model	Time
Learning rate 0.01	1h 49min 33s
Learning rate 0.001 = Combination 1	2h 7min 3s
Learning rate 0.0001	1h 45min 28s
Mini-batch 50	55min 51s
Mini-batch 100	2h 2min 58s
Mini-batch 150	2h 39min 35s
Network 1	1h 0min 0s
Network 2	1h 4min 59s
Network 3	51min 26s
Network 4	58min 51s
Combination 1 shuffled	1h 44min 12s
Combination 2	2h 36min 36s
Combination 2 shuffled	1h 44min 9s
Combination 3	1h 48min 51s
Combination 3 shuffled	2h 24min 4s
Combination 4	1h 54min 35s
Combination 4 shuffled	1h 53min 13s
Combination 5	1h 41min 30s
Combination 5 shuffled	2h 7min 16s
Combination 6	2h 11min 38s
Combination 6 shuffled	2h 8min 10s

4 Discussion

A neural network able to classify treatment plans into groups according to their GPR was established and optimised. As a first approach of assessing the performance of neural networks concerning the task mentioned above one possible implementation of a convolutional neural network was presented, which demonstrated the ability to produce reasonable output. The generation of input data, i.e. the transformation of radiotherapy treatment plans to fluence maps was accomplished as desired.

4.1 Interpretation of Results

Overall, accuracy values in the range of 57%-69% were observed upon training different network layer sequences with varying hyperparameters. The obtained results will be discussed and interpreted below.

The increase of network depth, i.e. the addition of convolutional and fully connected layers lead to the predicted improvement of network performance of ~5% to up to 64% accuracy compared to the smallest network presented in the template. This increase originates from the higher number of features that are identified and used for classification. However, the expected large increase of computation time with depth of network due to an increased number of parameters was not observed or only minimal.

Furthermore, the achieved results for learning rate and batch size variation supported the theoretical background and underlying fundamentals of machine learning as presented in 2.7. The largest learning rate exhibited the expected behaviour of oscillating around the minimum but not reaching it, due to the step size being too large, while the smaller learning rates converged to values of ~ 64%. The variations upon changing mini-batch size support the theoretical predictions, as the observed curves appear smoother with increasing batch size, probably related to the gradient of the loss function. The gradient is formed by the average of the differences of all predictions. Increasing the number of predictions – as it is the case when increasing the number of images passed to the network at once – enhances the precision of the gradient.

While the built network is robust concerning shuffling of input data, switching of datasets causes a discrepancy of accuracy of 10% (e.g. Combination 4 and 6 in Figure 28). The testing of robustness against shuffling behaved as desired, supporting the assumption that the order the input data is passed to the network does not influence the outcome. However, the large difference in accuracy observed when switching datasets seems to originate from the unbalanced distribution of input data to the three labels and thus the three datasets and indicates the necessity to improve the network in this context. Suggestions on how to overcome this limitation are described in section 4.3.

In general, the relevance of the analysis of computation times is questionable, since the runtime also depends on the workload on the CPU, which varied with the time of day and the personal activity and therefore the observations did not match the theoretically expected one. In contrast to the predictions, decreasing the learning rate did not always lead to a longer computation time and in some cases even to a decrease in runtime. However, the observed increase of computation time with batch size is in line with the fact that more images are used per computation step.

4.2 Comparison to Literature

The obtained results are compared with one achieved by Interian et al., since it was aimed to replicate and improve their stated outcome [42]. The best performing network, i.e. the CNN with four convolutional layers and three fully connected layers, reached an accuracy value of 69% upon training with a learning rate of 0.001 and a mini-batch size of 100. In comparison to that, Interian et al. achieved an accuracy of 0.70 ± 0.05 upon predicting GPR values of fluence maps. However, since the aim was to determine continuous values rather than distinct classes, the mean absolute error (MAE) was used as evaluation metric in order to measure accuracy. In contrast, the metric that is commonly used in classification problems and was therefore applied in this work is the cross entropy, making the direct comparison of the results challenging. However, the results achieved in this thesis are of the same order of magnitude as the achieved accuracy using a CNN to predict GPR based on fluence maps presented in literature and are therefore considered as plausible.

Another approach of using deep learning methods to evaluate QA data was investigated by Nyflot et al., using a convolutional neural network to identify errors in QA measurements and comparing the performance to manually selected features [59]. In contrast to Interian et al., the group focused on the detection of errors rather than predicting a parameter of the plan using gamma images displaying the gamma index of each point of the irradiated field. In this case, a two- or three-class classification problem was investigated, respectively. Their deep network achieved a maximum accuracy of 64.1% upon dealing with three classes, using the max margin loss function. The results suggest the achieved accuracies in this work hold up to the current findings of image classification of QA plans related to GPR in literature, even though again a different metric was used to compute the loss function.

What appears to be a satisfactory result at first glance has to be treated with caution: upon closer inspection of the evaluation of unknown data it was evident that the network classified all the input data as label 0, probably stemming from the large proportion of input images that incorporate the label 0. Furthermore, the networks showed largely varying accuracies upon switching of datasets, which leads to the evaluation of the encountered limitations within the presented work.

4.3 Limitations and Improvements

Despite the encouraging comparison with literature, further improvements are necessary in order to reach the final goal of implementing the proposed machine learning routine into the standard clinical workflow. The results show a wide range of different accuracy values upon testing different network structures and especially large variations when assessing the robustness of the built networks against switching of datasets. Therefore, the findings require identification and examination of possible errors, as well as a discussion of other approaches for future investigations of that matter. The limitations encountered over the course of this work occurred in different aspects of the input dataset, network building and training processes. The main problem encountered was the faulty classification of treatment plans in the group with highest GPR values, due to the predominant assignment of label 0 to the acquired treatment plans.

4.3.1 Dataset

As it is the case for all machine learning methods, increasing the number of patient data and therefore increasing the training examples is a method of improving the resulting accuracy. While the 600 used images show to be of the same order of magnitude as the one compared with literature, results are expected to improve with increasing number of input data. Increasing the number of images would in this case require to obtain older plans, with the current planning dates of the retrieved patient data for this thesis already spanning over three years. Including even older plans would reduce the uniformity and therefore quality of data, since planning and treatment techniques have changed and improved over time. Therefore, new data has to be acquired.

On this note, the improvements in delivery systems and planning software lead to enhanced GPR values for more recent plans. This causes the number of plans with lower GPRs and therefore plans with label 1 or 2 to be found rather sparsely, hindering the building of a valid CNN model. Therefore, the desired balanced distribution of data to labels was not possible, since almost 64% of all the plans were categorised with the label 0. Despite best efforts to adjust the intervals determining the labels to distribute the input data evenly to all labels, the number with images of associated label 0 was still the largest and the trained networks were prone to identifying most data with label 0, as it was observed upon evaluating unknown data. Since the GPRs were obtained using the 3mm/3% criteria, a method to discern planning data more precisely would be to apply the 2mm/2% criteria, further distributing the obtained gamma passing rates to lower values and increasing the variance in the dataset. Also, it might be necessary to re-investigate the data set for drifts of the linac and the detector, which could also influence the classification.

Other approaches of labelling the patient data were also considered. Separating the data into 4 classes did not yield valid output, since the resulting accuracy figures only equalled random guessing. Distinguishing only two classes and in that way introducing the pass-fail criteria of the QA measurement was considered as not fine enough for this application. This approach would also worsen the distribution of data to labels, since only plans with a GPR of $<90\%$ fail the QA measurement, which

amounts to only 5 plans out of 600. Again, in this scenario, a network that only predicts label 0 would yield a high accuracy but would not offer a feasible model. The chosen intervals of GPR determining the labels in this thesis were somewhat arbitrary and served mainly the goal to achieve a balanced distribution to the labels than a statement about failing or passing the QA measurement.

The differing number of images in each class also seems to be the factor influencing robustness, since different datasets lead to very different outputs. In line with that argument, datasets 2 and 3, the combination that yielded the best accuracy, also contain most of the images classified with label 0. This leads to the assumption that the high training accuracy is caused by the predominant classification of label 0, which is correct in most of the cases. The substantially worse performance of the evaluation dataset on the trained model of only 59.5% evaluation accuracy compared to 69% training accuracy can be explained by the lower ratio of label 0 to label 1 and 2, therefore a larger portion of predictions are incorrect.

However, using non-balanced distributions of input data to labels is the general situation in current research and is therefore a problem that can be circumvented by choosing appropriate calculation metrics and other methods to prevent the network leaning toward a predominantly present label. Methods that have been proposed to counteract unbalanced distributions include oversampling, i.e. artificial increase of numbers of underrepresented labels, therefore either rotating or mirroring images to generate more input data and thus balancing the distribution in that way. This method of data augmentation is also generally used to increase the number of input data, but makes the network more prone to overfitting. On that note, applying a different technique to counteract overfitting could also be tested in the future. Another obvious method of balancing the number of labels would be to omit data from the predominant label, i.e. undersampling. This method therefore reduces the number of total data instances, and hence reduces the network performance. Finally, choosing more sophisticated metric to even out the unbalanced distribution enables the setup of a more feasible network without further manipulating the input data.

4.3.2 Metric

The used metric to calculate the loss function as previously discussed was the TensorFlow function sparse softmax cross entropy, allowing the classification of mutually exclusive labels. However, in this case other metrics such as the softmax cross entropy would have offered the possibility to put weights to the different classes, therefore enabling the prioritisation of images with labels that only occur sparsely and therefore compensating the imbalance in the distribution. A built-in function is available in TensorFlow, which requires the specification of a tensor containing a weight for each sample of the passed data [60]. In this context, the framework TensorFlow itself is currently worked on and updated frequently, therefore possibly offering future improvements in mathematical implementations as well as more efficient computation.

4.3.3 Fluence maps dimensions

As described, the dimensions of the matrix and therefore the size of the generated fluence maps images was set to 400x80 pixels. Nevertheless, since the positional information of the collimators is available in sub-millimetre precision, a finer image generation would be possible by increasing the image width to 4000 pixels, therefore increasing the precision of the network predictions. However, this would also drastically increase the parameters of the input and consequently all following layers. Subsequently, the increased number of necessary computations of parameters are more expensive concerning memory of the machine and could therefore be investigated on a system with higher computing power.

4.3.4 Physical machine memory

Another problem that was encountered that limited the possibilities of further research was the limited memory available in the used PC. Therefore, sampling with a batch-size of the complete dataset of 200 images was not possible due to a limited RAM of 8 GB. While computation time itself had not posed any obstacles, the models trainings aiming to investigate batch sizes of 200 images were terminated due to a lack of memory. All operations were performed on the included CPU with 2.90 GHz clock frequency, which was sufficient for the performed tasks since computation times

were in the range of one to three hours. The option of using the built-in GPU was considered, but due to lacking memory, not even small, simple networks were able to be trained on the graphical device. One suggestion for improvements for the future is therefore the use of GPUs with larger memory, since the computation operations on tensors can be parallelised and therefore runtime is improved. Alternatively, using a PC with increased RAM would already enable testing of wider and deeper networks.

4.4 Further improvements

Another concept that may offer further potential but was not applied in this work was to include the information of the total number of monitor units and checkpoints per plan in the input to the network in order to consider not only the relative but the absolute dose output of different plans. Since the used fluence maps show only the relative fluence distribution of each plan, i.e. the darkest region of prostate plans does not correlate to the same dose as the darkest region of HN plans, considering the full delivered dose as input parameter for the network may contribute to the assessment of complexity and GPR. Additionally, the number of checkpoints may also hold valuable information that could be used assessing the complexity and GPR, since a slight correlation of the number of checkpoints and GPR values is expected. A method to circumvent the convolution blocks and feed these parameters directly to the network needs to be developed.

Expanding the network size, i.e. the number of neurons, filters and kernel sizes per layer was not investigated in this thesis for multiple reasons. In order to understand the fundamental mechanisms and relations between parameters and the behaviour of the network, other parameters that were easier to vary and more straightforward were tested first. Additionally, the aforementioned memory issues prevented the further widening of the network.

The application of the investigated CNNs is not limited to VMAT treatments only. Since the fluence maps present the input information and may be created from TPS output for all the different radiation methods, the concepts may also be adapted for particle therapy.

Finally, the performed optimisations provided a first insight into neural networks and the possibility of incorporating machine learning algorithms in the clinical workflow to improve patient care and reduce time for quality assurance measurements. Using the proposed approaches to improve the CNN and the input data likewise, a constant improvement of the achieved results in this work can be expected.

5 Conclusion and Outlook

For applying new machine learning algorithms in the clinical workflow and substituting a well-understood patient specific QA process, high reliabilities and accuracies of results are required. This thesis set the first step into investigating neural networks designed for the special purpose of identifying the complexity of fluence maps for Versa HD and Synergy linacs allowing a classification according to the gamma passing rate.

In conclusion, incorporating machine learning algorithms and especially neural networks into the clinical workflow to estimate GPR values of treatment plans is a promising approach for future improvements of the QA procedure. Strategies on how to overcome the encountered limitations have been proposed and may be implemented in the future. Enlarging and improving the dataset is one of the key aspects expected to improve model performance, along with choosing a computation metric more suitable for the task and increasing the resolution of input images. That, in addition to further optimisation of the network dimensions is possible on a PC with larger computing power. Nevertheless, the results achieved in this work are comparable to current findings in literature and show a high potential of improvement that may be investigated in the future.

References

[1] "Krebs im Überblick." [Online]. Available: http://www.statistik-austria.at/web_de/statistiken/menschen_und_gesellschaft/gesundheit/krebserkrankungen/krebs_im_ueberblick/index.html. [Accessed: 13-Dec-2018].

[2] "Todesursachen im Überblick." [Online]. Available: http://www.statistik-austria.at/web_de/statistiken/menschen_und_gesellschaft/gesundheit/todesursachen/todesursachen_im_ueberblick/index.html. [Accessed: 13-Dec-2018].

[3] F. M. Khan, "Physics of Radiation Therapy Third Edition," *J. Am. Med. Assoc.*, p. 1138, 2003.

[4] D. Radstone, "Radiotherapy: principles and applications," in *Gynaecological Oncology for the MRCOG and Beyond*, N. Acheson and D. Luesley, Eds. Cambridge: Cambridge University Press, 2011, pp. 95–102.

[5] L. Freund, "Elements of General Radiotherapy for Practitioners," *Rehman, New York*, 1904.

[6] A. T. Porter and J. D. Forman, "Prostate brachytherapy. An overview. [Review]," *Cancer*, vol. 71, no. 3 Suppl, pp. 953–958, 1993.

[7] P. Mayles, "Handbook of radiotherapy physics: theory and practice," p. 1425, 2007.

[8] E. D. Podgorsak and International Atomic Energy Agency., *Radiation oncology physics : a handbook for teachers and students.* International Atomic Energy Agency, 2005.

[9] E. E. Klein, M. Vicic, C. M. Ma, D. A. Low, and R. E. Drzymala, "Validation of calculations for electrons modulated with conventional photon multileaf collimators," *Phys. Med. Biol.*, vol. 53, no. 5, pp. 1183–1208, 2008.

[10] European Respiratory Society., *Breathe.* European Respiratory Society, 1969.

[11] A. Taylor and M. E. B. Powell, "Intensity-modulated radiotherapy - What is it?," *Cancer Imaging*, vol. 4, no. 2, pp. 68–73, 2004.

[12] M. Teoh, C. H. Clark, K. Wood, S. Whitaker, and A. Nisbet, "Volumetric modulated arc therapy: a review of current literature and clinical use in practice," *Br. J. Radiol.*, vol. 84, no. 1007, pp. 967–996, 2011.

[13] S. Rana, "Intensity modulated radiation therapy versus volumetric intensity modulated arc therapy," *J. Med. Radiat. Sci.*, vol. 60, no. 3, pp. 81–83, 2013.

[14] D. A. Low, W. B. Harms, S. Mutic, and J. A. Purdy, "A technique for the quantitative evaluation of dose distributions," no. March, pp. 656–661, 1998.

[15] M. Hussein, C. H. Clark, and A. Nisbet, "Challenges in calculation of the gamma index in radiotherapy - Towards good practice," *Phys. Medica Eur. J. Med. Phys.*, vol. 36, pp. 1–11, 2017.

[16] W. Ertel, *Introduction to Deep Artificial Intelligence, Undergraduate Topics in Computer Science*. 2018.

[17] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT Press , Cambridge, Mass. [u.a.], 2012.

[18] S. M. Shamim, M. Badrul, A. Miah, A. Sarker, M. Rana, and A. Al Jobair, "Handwritten Digit Recognition using Machine Learning Algorithms," no. July, pp. 15–23, 2018.

[19] H. Kulkarni, "Unconstrained Facial Recognition using Supervised Deep Learning on Video," no. May, 2018.

[20] J. H. Lee, J. Shin, and M. J. Realff, "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Comput. Chem. Eng.*, vol. 114, pp. 111–121, 2018.

[21] R. J. Reynolds and S. M. Day, "The growing role of machine learning and artificial intelligence in developmental medicine," *Dev. Med. Child Neurol.*, vol. 60, no. 9, pp. 858–859, 2018.

[22] W. . Baxt, "Application of artificial neural networks to clinical medicine," *Lancet*, vol. 346, no. 8983, pp. 1135–1138, 1995.

[23] C. Park, C. C. Took, and J.-K. Seong, "Machine learning in biomedical engineering," *Biomed. Eng. Lett.*, vol. 8, no. 1, pp. 1–3, 2018.

[24] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[25] C. E. Cardenas *et al.*, "Auto-delineation of Oropharyngeal Clinical Target Volumes Using Three-Dimensional Convolutional Neural Networks," *Phys. Med. Biol.*, 2018.

[26] R. Vivanti, L. Joskowicz, N. Lev-Cohain, A. Ephrat, and J. Sosna, "Patient-specific and global convolutional neural networks for robust automatic liver tumor delineation in follow-up CT studies," *Med. Biol. Eng. Comput.*, vol. 56, no. 9, pp. 1699–1713, 2018.

[27] Y. Wang *et al.*, "Automatic Tumor Segmentation with Deep Convolutional Neural Networks for Radiotherapy Applications," *Neural Process. Lett.*, pp. 1323–1334, 2018.

[28] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[29] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[30] "A Gradual and Gentle Introduction to Deep Learning — AI-SS: Artificial Intelligence Software Solutions." [Online]. Available: https://www.ai-ss.org/a-gradual-and-gentle-introduction-to-deep-learning/. [Accessed: 16-Dec-2018].

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[32] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," 2014.

[33] "Machine learning fundamentals (II): Neural networks." [Online]. Available: https://towardsdatascience.com/machine-learning-fundamentals-ii-neural-networks-f1e7b2cb3eef. [Accessed: 05-Dec-2018].

[34] M. Early and A. Program, "Machine Learning with TensorFlow Version 10 MEAP Edition Manning Early Access Program Copyright 2017 Manning Publications."

[35] "Understanding activation functions better - bobdc.blog." [Online]. Available: http://www.snee.com/bobdc.blog/2017/09/understanding-activation-funct.html. [Accessed: 19-Dec-2018].

[36] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks Xavier," *Proc. Fourteenth Int. Conf. Artif. Intell. Stat.*, vol. 15, pp. 315–323, 2011.

[37] "Convolutional Neural Networks (LeNet) — DeepLearning 0.1 documentation." [Online]. Available: http://deeplearning.net/tutorial/lenet.html. [Accessed: 21-Dec-2018].

[38] "2: An example of convolution operation in 2D 2 . | Download Scientific Diagram." [Online]. Available: https://www.researchgate.net/figure/An-example-of-convolution-operation-in-2D-2_fig3_324165524. [Accessed: 20-Dec-2018].

[39] G. Carlsson, "Using Topological Data Analysis to Understand the Behavior of Convolutional Neural Networks." [Online]. Available:

https://www.ayasdi.com/blog/artificial-intelligence/using-topological-data-analysis-understand-behavior-convolutional-neural-networks/.

[40] Y. LeCun, L. Bottou, L. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[42] Y. Interian *et al.*, "Deep Nets vs Expert Designed Features in Medical Physics: An IMRT QA case study," *Med. Phys.*, 2018.

[43] H. Li, L. Dong, L. Zhang, J. N. Yang, M. T. Gillin, and X. R. Zhu, "Toward a better understanding of the gamma index: Investigation of parameters with a surface-based distance method," *Med. Phys.*, vol. 38, no. 12, pp. 6730–6741, 2011.

[44] "All CIODs — DICOM Standard Browser." [Online]. Available: https://Dicom.innolitics.com/ciods. [Accessed: 19-Jun-2018].

[45] "lxml.etree." [Online]. Available: https://lxml.de/api/lxml.etree-module.html. [Accessed: 08-Jan-2019].

[46] "os — Miscellaneous operating system interfaces — Python 3.7.2 documentation." [Online]. Available: https://docs.python.org/3/library/os.html. [Accessed: 08-Jan-2019].

[47] "DCMTK: xml2dcm: Convert XML document to DICOM file or data set." [Online]. Available: https://support.dcmtk.org/docs/xml2dcm.html. [Accessed: 10-Jan-2019].

[48] "Dicompyler." [Online]. Available: http://www.Dicompyler.com/. [Accessed: 11-Jun-2018].

[49] "NumPy — NumPy." [Online]. Available: http://www.numpy.org/. [Accessed: 08-Jan-2019].

[50] "Matplotlib: Python plotting — Matplotlib 3.0.2 documentation." [Online]. Available: https://matplotlib.org/. [Accessed: 08-Jan-2019].

[51] A. Damien, "https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/5_DataManagement/build_an_image_dataset.py." Github repository, 2015.

[52] "TensorFlow." [Online]. Available: https://www.tensorflow.org/. [Accessed: 12-Jan-2019].

[53] "TensorBoard: Visualizing Learning | TensorFlow." [Online]. Available: https://www.tensorflow.org/guide/summaries_and_tensorboard. [Accessed: 12-Jan-2019].

[54] D. Kingma and J. Lei Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6, pp. 58–62, 2015.

[55] "Regularization for Simplicity: L_2 Regularization | Machine Learning Crash Course | Google Developers." [Online]. Available: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization. [Accessed: 15-Jan-2019].

[56] "Machine Learning Glossary | Google Developers." [Online]. Available: https://developers.google.com/machine-learning/glossary/. [Accessed: 15-Jan-2019].

[57] "Setting the learning rate of your neural network." [Online]. Available: https://www.jeremyjordan.me/nn-learning-rate/. [Accessed: 17-Jan-2019].

[58] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima," pp. 1–16, 2016.

[59] M. J. Nyflot, P. Thammasorn, L. S. Wootton, E. C. Ford, and W. A. Chaovalitwongse, "Deep learning for patient-specific quality assurance: identifying errors in radiotherapy delivery by radiomic analysis of gamma images with convolutional neural networks," *Med. Phys.*, vol. 0, no. 0, p. mp.13338, 2018.

[60] "tf.losses.softmax_cross_entropy | TensorFlow." [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/losses/softmax_cross_entropy. [Accessed: 23-Jan-2019].

List of Tables

List of Figures

Appendix: Documentation

monaco2dicom.py

- Takes path of folder that contains subfolders (name = patient ID) containing tel.1-files, creates a list of full paths for each tel1-file
- Identifies sections in tel1-file that contain wanted information and writes those in ElementTree-element
- Performs necessary calculations on jaw positions and value that is used to infer cumulative beam meterset weight
- Rearranges values of leaf positions
- Assigns label to each file depending on the GPR of the plan, to be found in a txt file containing strings connecting patient ID and GPR
- Writes XML and further converts it to Dicom using Patient ID & label as name and saves it to given path

fluence_maps.py

- Takes path of folder containing Dicom files
- Scans through Dicom; using tags and DicomParser to find Checkpoints and needed values to calculate transmitted dose
- Checks if used linac is of type Versa or Synergy and therefore differentiates needed values
- Reads positions of collimators and transforms to match chosen coordinate system
- Calculates dose for different regions: covered by collimators, not covered by collimators, party covered since collimator move, considering both jaws and MLC leaf pairs
- Writes matrix with 400 columns and 80 rows
- For linacs with only 40 leaf pairs, every other row is copied to fill matrix
- Matrix is converted to grayscale PNG image, named after input Dicom name (Patient ID & label)

neuralnet1.py

- Reads images either from a txt file (give path) containing full image paths and corresponding label or from folder containing subfolder with images (names contain their corresponding label)
- Converts input PNGs to tensor

- Builds network model in TensorFlow using convolutional, pooling, flattening and fully connected layers
- Forms regularizer and add to convolutional layers and loss function
- Performs training using given training and testing datasets
- Calculates accuracy and loss and writes values as well as model graph to TensorBoard
- Saves model with tf.Saver

randomise.py

- Takes txt file containing full image paths as rows as input, outputs shuffled list in a new file

group_data.py

- Takes txt file containing full image paths as rows as input, looks up patient ID (=name of image) in txt file containing IDs and corresponding GPR values
- Writes full path and new label according to chosen GPR intervals into new output file