TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Monte Carlo simulations of X-ray sources by machine learning approaches

Ausgeführt am Atominstitut der Technischen Universität Wien

In Zusammenarbeit mit der

Universitätsklinik für Radioonkologie

der Medizinische Universität Wien

unter der Anleitung von Univ.-Prof. Dr. DI Dietmar Georg

und Ko-Betreuung von DI Hermann Fuchs, PhD

und Mag. Peter Kuess, PhD

durch

Sanaz Alijani

[redacted]

Oktober 2021 Unterschrift (Student)

Contents

Acknowledgements

I would like to thank my thesis advisors Hermann Fuchs and Peter Kuess for their constant motivation and encouragement for my work. They consistently allowed this thesis to be our own work but steered us in the right direction whenever they thought we needed it. I would also like to thank Lukas Fetty who was involved in the validation survey for this research project.

Without their passionate participation and input, the validation survey could not have been successfully conducted. Also, Professor Dietmar Georg for providing me the chance to do my master thesis at the Medical University of Vienna. An especial gratitude to my family and my husband for unconditional support and care.

List of Figures

List of Tables

Abstract

This master thesis introduces an approach to using Generative Adversarial Networks for the generation of phase space to replace the generated phase space instead of large phase space datasets. The original approach was produced by Monte Carlo method of ImagingRing™ system at MedAustron. This is intended to create the generated particles that can be used in research areas, while creating the conventional sampling of phase space is time consuming and challenging.

To evaluate the outcome of GAN, some methods are proposed to validate the generated particles. The efficiency of the generated particles produced by GAN has been checked and satisfactory results have been gained for this research.

As the main result, the particles in the energy range of 20 keV-60 keV were generated with the maximum statistical and theoretical significance. In addition, the superimposition of the original phase space and generated one can be obtained in this given range of the energy. This study shows that no particles were generated in the energy above 70 keV for X-Y parameters and the particles in the energy range between 60 keV-70 keV were generated with lower superimposition of the generated particle and original one due to lack of the reference particles.

The generated particles by GAN requires only around 10 MB storage compared to the phase space produced by ImagingRing™ System which contains tens of Gigabyte data. Besides, the process of the particle generation is fast and it is efficient to use.

Moreover, a novel research pathway in the statistical techniques for validation of the generated phase space has been opened so that further research can be developed.

1 Introduction

In this chapter, the motivation and aim of this master thesis will be revealed first, and then a brief description of the thesis structure will be defined accordingly.

1.1 Motivation and objectives

The widespread use of Monto Carlo (MC) simulation is for modeling sophisticated radiation machines such as X-ray tubes or ion therapy beam lines [68][35]. At MedAustron a novel X-Ray imaging device, the ImagingRing™ System is used. A full Monte Carlo model of this system was already created. To create the precise Bremsstrahlung spectrum and accurate angular distribution, modeling of the X-ray tube by simulating the electron interactions is required.
The time of computation to implement such simulation is relatively high and phase space files have been acknowledged to reduce this time by pre-calculating and storing large numbers of particles [68][35].
Such a phase space file contains the information of particle position, energy and angular momentum distribution. However, these files contain around 50 gigabytes of data and are cumbersome and inefficient to use.
This master thesis aims to implement a novel approach to replace phase space files with a trained neural network and validation of model performance. For this purpose, the generated phase space has to be compared with the full Monte Carlo model of x-ray source provided at MedAustron. In particular, the questions are considered to what extend the generated phase space is similar to conventional one and about the feasibility of using the generated phase space instead of the one mentioned above.

1.2 Thesis structure

This thesis is structured into seven chapters. Chapter 2 describes the complete introduction of the X-ray tube and its applications in medicine and the basics of Monte Carlo simulations. Since our used method is a machine learning task, chapter 3 is designed based on the theoretical explanation of machine learning theory, neural networks, network training, and some basics of gradient descent as well as activation function. In chapter 4, the detailed explanation about Generative Adversarial Network (GAN) and Wasserstein GAN (WGAN) which are used as the neural network technique in this master thesis are introduced. Besides, a short explanation of the used technology and the training and optimization of GAN as well as the evaluation techniques of the results of the GAN are explained in chapter 5. In this thesis, these explained techniques were used to generate data by machine learning, that is supposed to match the actual data from measurements and evaluate to which extend the actual data is matched, by using machine learning techniques. Ultimately the feasibility of this new approach is assessed, taking multidisciplinary aspects into account.
All the results are shown in chapter 6. Finally, in the last chapter, the discussion and the conclusion of the project are given and the limitations which were faced with. Additionally, further recommendation is proposed.

2 Background and related work

MC simulations are extensively utilized to characterize complex radiation devices, such as X-ray tubes or ion therapy beam lines. Nevertheless, in general, MC simulations could be used to simulate brachytherapy radionuclide seeds, proton beam nuzzles, nuclear imaging process, etc. An obvious example is to calculate the dose in a patient CT image, in which the simulation is divided into 2 parts [35][68]. Firstly, detailed MC simulation is done in order to transmit particles through the accelerator treatment head elements (primary collimation, flattening filter, monitor chambers, mirrors, secondary collimation, etc.) to a virtual plane. The features of particles such as energy, position and direction which reach the plane are kept in the phase space file and base on detailed features of the treatment head components. The second part of the simulation is to track particles from the phase space plane through the multi-leaf collimator and the patient CT image to analyze and evaluate the distribution dose which is absorbed [68].

In the following, an outline of the principles of X-ray production and spectrum and the applications of X-ray in medicine to characterize and model the radiation output of X-ray tubes are given. Then, the purpose of the radiotherapy and the different treatment planning is discussed. Lastly, the basic introduction of the Monte Carlo simulation is introduced in this chapter.

2.1 X-ray history

X-rays could be produced by natural sources, such as radon gas or radioactive elements on earth, but can also be generated by technical means. The discovery of X-ray by Wilhelm Conrad Röntgen in 1895 marked the beginning of a revolution in Medical Imaging and Radiation Therapy. While applying a high voltage to a cathode tube, he observed that crystals near the tube started to glow. From subsequent experiments, Röntgen concluded that these have been caused by a radiation, that was able to penetrate through most substances, including human tissue. However there were also substances which X-rays could not pass through, like bones and metal, allowing multiple useful applications [45].

2.1.1 Fundamentals of X-ray production

An x-ray tube is an evacuated tube that contains a cathode and an anode, as shown in figure 2.1. The cathode is heated by a low voltage current, causing the thermal emission of electrons. The released electrons are then accelerated towards the anode by a high voltage. When hitting the anode, the electrons are abruptly decelerated, resulting in the emission of high-energy electomagnetic waves. Only a fraction of the kinetic energy is transfered into radiation, the rest is deposited in the anode as thermal energy, implying the need for cooling the anode [45].

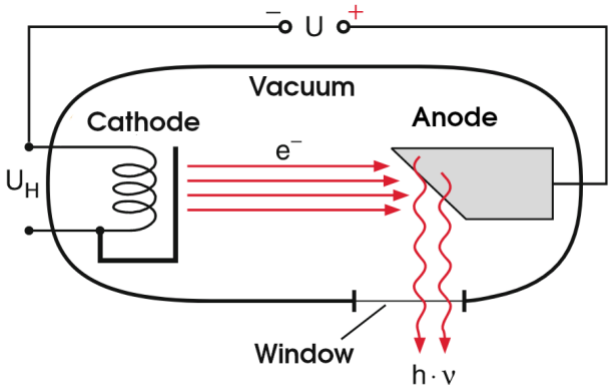


Figure 2.1: X-ray tube (modified from source [45] p.254). The cathode and the anode are seen inside the tube.

The following is based on the “Diagnostic Radiology Physics“ book [42].

2.1.2 X-ray spectrum

X-ray output can be plotted as a graph called spectrum. X-ray spectra are the results of the deceleration of incoming electrons on the anode when high energy is applied to the vacuum tube. X-ray spectrum is the result of attenuation of the X-ray beam, which is generated at the target. Two types of X-ray spectra as shown in figure 2.2, were defined: Continuous and Discrete. They both result of physical process which produce X-ray called Bremsstrahlung and characteristic radiation.

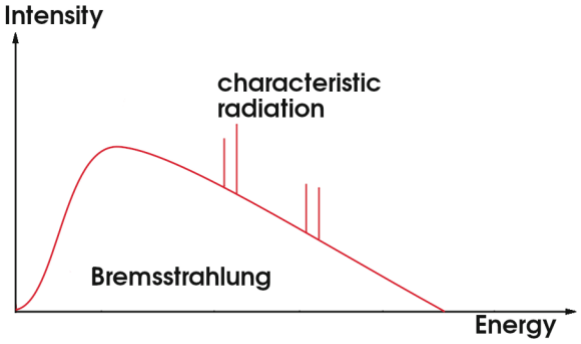


Figure 2.2: X-ray spectrum (modified from source [42] p.92): Continuous spectrum (Bremsstrahlung) and discrete lines (Characteristic radiation)

2.1.3 Bremsstrahlung

This type of inelastic interaction takes place when an incident electron interacts with the electric field of the nucleus. The bremsstrahlung spectrum results in the slowing down of incident electrons and their energy loss. The loss of energy from incident electrons due to bremsstrahlung is caused by inelastic scattering or by emission of an X-ray photon due to the radial acceleration in the electric field of the nucleus. The probability of bremsstrahlung highly depends on Z^2 (Atomic number); therefore, the efficiency of bremsstrahlung radiation for heavy material such as tungsten is better. The energy of the bremsstrahlung is subtracted from the kinetic energy of the electron. Additionally, the angle of photon emission is related to the energy of electron.

2.1.4 Characteristic radiation

The interaction of an incident electron with an inner shell electron causes the inner electron to get ionized and leave the atom if its kinetic energy exceeds the binding energy. The generated vacancy will be filled with an electron from a higher energy level. When the outer shell electron drops from a higher energy state into a lower energy one, the difference of the energy is released as either an X-ray photon or in the scattered electron's energy as it is shown in figure 2.3. These photons produce characteristic lines in the spectrum, such as K-lines and L-lines.

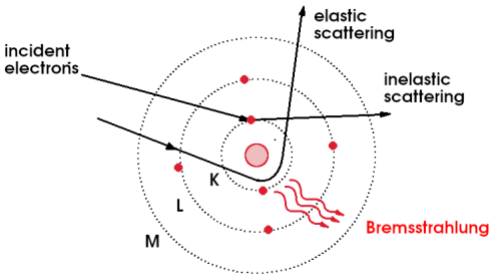


Figure 2.3: Schematic interaction of incident electrons on matter (modified from source [45] p.254)

The ideal X-ray spectrum without any filtration would be triangular. Different filtration such as beryllium and aluminum is used in the actual spectrum to illustrate the characteristic lines and continuous spectrum. At the lower portion of the energy of spectrum, for instance at 60 kV, as shown in figure 2.4, there are no characteristic lines.

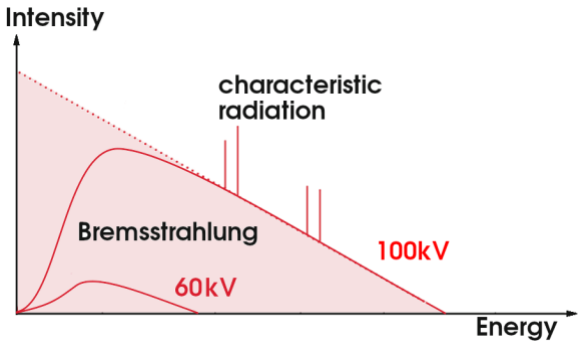


Figure 2.4: The ideal X-ray spectrum. The filtration technique provides the characteristic lines and continuous spectrum for the X-ray spectrum (modified from source [42] p.92)

2.2 X-ray sources in medicine

X-ray sources have various applications in medicine such as image-guided therapy (radiotherapy imaging) and medical diagnostic imaging. The latter denotes the wide range of multiple image modalities: mammography, computed tomography, systems for general radiography and interventional X-ray systems [4]. In radiation oncology departments, the aim for diagnostic imaging is to observe and detect the existing tumor, rather for radiotherapy imaging the exact position and the shape of tumor are searched and the plan of treatment based on the dose calculation is designed.

2.2.1 MedAustron

MedAustron is a center for ion beam therapy and research, located in Wiener Neustadt in Lower Austria and is one of the few centers around the world providing treatment with carbon ions. For the first time, a carbon ion accelerator for clinical usage was opened by the National Institute of Radiological Science (China and Japan in 1994) and since that time, thousands of patients have been treated [27] [26]. However, still, the number of centers providing carbon in ions is scarce. In total, there have been five countries and 12 centers using carbon ion radiation therapy so far [28][40].
A full Monte Carlo model of the ImagingRing™ System used in this master thesis were performed at MedAustron in the Irradiation Room1 (IR1) which is used for non-clinical research purposes. The ImagingRing™ System which will be described in detail in section 2.4, provided for in-room patient imaging and intended to be used as the equipment for creating phase space.

2.3 Radiation Therapy

Cancer is one of the most causes of mortality and morbidity in the recent century [43]. In order to treat the cancerous organs or tissue, various types of treatment can be considered. Radiation Therapy is one of the effective non-surgical cancer treatment modalities which uses high radiation doses to kill or shrink the tumor cells. Although radiation therapy can also be used for treatment of the non-malignant disorders [24][25].
Radiation is usually called ionizing radiation since the ionized charged particles deposit energy in the body. The aim is to deliver radiation to a target volume within the body without any damage to normal surrounding tissues [8]. High energy beams eliminate the tumor cell growth by stopping the ability of genetic material (DNA) to divide and proliferate further [9]. The treatment plan will be designed for each patient depending on the type of tumor and stage of cancer. There are two common ways to deliver radiation to the location of tumors: external beam radiation and internal beam radiation [8].

- External Radiation Therapy

The high-energy X-ray beams (4-20 million volt) are produced mostly by a linear accelerator (LINAC) passing through the collimators and then penetrate to the

most deep-seated tumors. This type of radiation therapy is planned to radiate the target tumor from outside to close location of tumor by aiming of high energy beams. The most effective clinically dose used is 6-18 MV which has an accurate balance between penetration and surface dose, while sparing the skin and normal surrounding tissues [6] [7] [8].

2.3.1 Volumes and margins

The aim of radiation therapy is to irradiate the cancerous cells by eliminating their ability to reproduce, while the exposure dose to healthy tissue in the surrounding needs to be as low as possible. However, it is not feasible to treat the malignant tumors without irradiating neighboring healthy tissue. In order to standardize the description and approach of the distributed dose, a number of different tumor volumes and margins were briefly introduced. In this case, the Tumor Control Probability (TCP) and the Normal Tissue Complication Probability (NTCP) will be improved which causes the optimization of the therapeutic windowing.

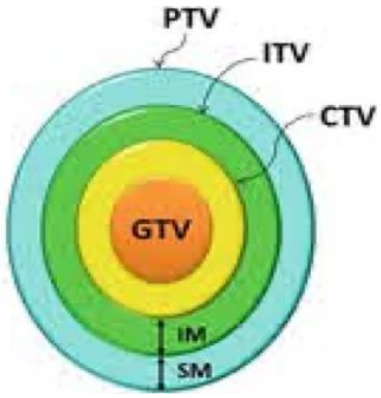


Figure 2.5: Definition of target volumes [43] in which PTV shows Planning Target Volume, GTV describes Gross Tumor Volume, ITV denotes Internal Target Volume and CTV defines Clinical Target Volume [31][32].

The following list of abbreviations:

GTV Gross Tumor Volume: "is the gross visible or demonstrable extent and location of malignant growth" [31].

CTV Clinical Target Volume: "is the tissue volume that contains a demonstrable GTV and/or subclinical/invisible malignant disease, which has to be irradiated. The aim of radiotherapy is to treat this volume " [31].

IM Internal Margin: "is responsible for considering the variations in size, shape, and position of the CTV in relation to anatomical reference points" [32].

SM Setup Margin: "is defined to specify the uncertainties in patient-beam positioning" [32].

ITV Internal Target Volume: "is the tissue volume that contains Clinical Target Volume (CTV) and Internal Margin (IM) for organ motion" [32].

OAR Organ at Risk: "are healthy tissues whose radiation can lead to influence treatment planning. OR are usually placed close to the clinical target volume (CTV)" [31,32].

PTV Planning Target Volume: "is a geometrical concept, which is defined for uncertainties in treatment planning and delivery. This will aid to choose suitable beam size and their arrangements which considers the possible variations in geometry and inaccuracies to provide certainty that does prescribed is in fact absorbed in the CTV" [31,32].

2.3.1.1 Treatment planning and dose calculation

In the first step for radiation therapy, a 3D model of the patient must be created. For this purpose, a computed tomography scan or magnetic resonance imaging is performed depending on the type and place of the tumor for delineating the margins and volumes. Additionally, soft tissues with high contrast can be captured. In the further step, the dose level, angle of the beams, and the number of the beams needs to be set. In other words, dose calculation shows the relation between the parameters of treatment explained in the process of planning, as well as their clinical results. The parameters of treatment require to be improved so that 95% of the PTV gets the determined dose of 95%. More than that, several limitations such as the Median Dose ($D_{50\%}$), Near-min Dose ($D_{98\%}$), and the Near-max Dose ($D_{2\%}$) to diverse volumes are explained, making the process more challenging [33][34]. This treatment planning is a repetitive process which finishes with the treatment plan to be optimized. The tumor has an irregular shape, to limit the excess irradiation to neighboring normal tissue, the conformal radiation therapy (CORT) is introduced which is divided into two main groups; 3D conformal radiation therapy (3D CORT) and Intensity-modulated radiation

therapy. 3D CORT uses the geometrical properties of the tumor to irradiate the high number of beams to target volume.

2.3.2 Treatment techniques

In the following subsections, some of the treatment methods used in radiotherapy will be explained.

2.3.2.1 3D Conformal Radiation Therapy

The 3D conformal radiotherapy is a technique that aims to shape the radiation beams to the PTV. This technique adds CT planning in which the volume to be dealt with explained on a 3D data set. As a result, organs in danger are depicted and defined to protect them and to reduce side effects. In order to design the complicated arrangement of beams, Radiotherapy planning software is utilized to evaluate the dose-volume [36].

2.3.2.2 Intensity Modulated Radiation Therapy (IMRT)

Intensity modulated radiation therapy (IMRT) regulates the shape and intensity of the beam during treatment by aiming to transfer the higher dose to CTV and spare OAR better. For this issue, the angle of the beam will be varied most of the times during the treatment. One type of IMRT in an advanced kind is known as Volumetric Intensity Modulated Arc Therapy (VMAT). In this type, the radiation is sent by a linear accelerator machine (LINAC) which is revolving around the patient. Both the speed of rotation and the intensity of it are in the process of modulation in this regard. The field shape is constantly adapted and adjusted by an Multi-leaf Collimator (MLC) that allows flexibility to some degree during treatment process. The advantage of using VMRT in contrast to IMRT is, that it is less time consuming in terms of the treatment planning process [37].

2.3.2.3 Image Guided Radiation Therapy (IGRT)

The therapy which uses imaging during radiotherapy is called image guided radiotherapy (IGRT). Its aim is to enhance the level of validity and it corrects the mispositioning of the patient by acquiring daily imaging of the patient. IGRT is a great advantage to be utilized to treat various kinds of tumors; however, it is

useful when tumors are placed closely to highly sensitive organs. Moreover, it can also be beneficial regarding tumors which have the tendency to be located near vital organs such as lungs and livers during the process of treatment planning. Because of this, positions of OR and tumor are determined straight away before the beginning of treatment fraction, and comparisons in terms of the patient's positions are made during planning treatment. For this reason, IGRT needs to use one imaging system which is built into the system of treatment [38].

2.3.2.4 Adaptive Radiation Therapy (ART)

Due to the fact that the plan of treatment is designed a couple of weeks prior the Radiation Therapy, anatomical changes, such as losing weight cause changes in volume and size of the internal organs. These changes require an adaption of the treatment plan. Adaptive Radiation Therapy (ART) is defined to modify the plan of treatment by evaluating the errors between CTV and delivered dose distribution to the patient. Then, images must be acquired in order to observe and verify the possible variations. Finally, the parameters of treatment planning will be estimated in order to modify the PTV [39].

2.3.2.5 Four-Dimensional Radiation Therapy (4DRT)

Respiratory motion can lead to organs displacing, especially in tumors which are located within the thoracic cavity or near the diaphragm. For tumors located around the diaphragm, motion amplitude can be as large as three centimeters. Due to this fact, Four-Dimensional Radiation Therapy (4DRT) is designed to evaluate the tumor movement. This technique helps to eliminate unwanted damage to the healthy surrounding tissues. In this type of radiation therapy, the radiation is adapted to the patient's breathing.

2.3.3 Proton beam therapy

The idea of using protons in therapeutic medicine was first proposed by Robert Wilson in 1946 to treat tumors located deep in the body with accelerated proton beams [19].
Proton therapy is a type of external radiation therapy, which aims to irradiate and treat target tumors with high energy beams and high accuracy [21]. Protons

deposit their maximum energy in the Bragg Peak and reach their maximum penetration depth as they decelerate [20]. In proton therapy, the energy deposits quickly to zero in the process of slowing down and thus the maximum dose is delivered to the target tumor and less spread on near-normal tissue [21]. These features allow the tumors to be irradiated with maximum efficiency, while healthy surrounding tissues and organs are less irradiated and more critical structures spared [23]. The difference of the dose-depth plot for X-rays, electrons, and protons is illustrated in figure 2.6. Recent studies demonstrate that 50% of spreading of dose on normal tissue is reduced in proton therapy in comparison to photon therapy [22].

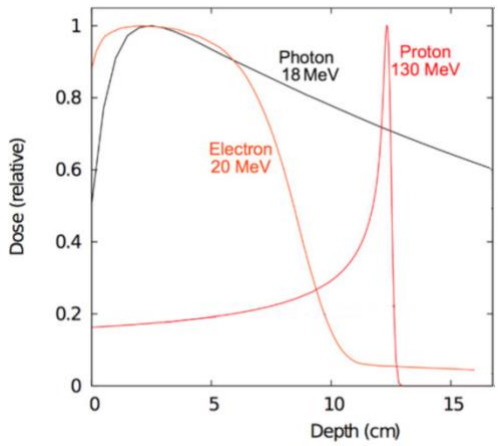


Figure 2.6: The dose-depth curve (modified from source [46]) for electron (20 MeV), photon (18 MeV) and proton (130MeV). Note the different shapes of the mentioned curves.

2.4 ImagingRing™ System

The ImagingRing™ System, which is operated at MedAustron, as shown in figure 2.8, is a special type of cone-beam computed tomography for medical imaging [11][12][14]. It resembles an industrial robot, designed for medical use and includes two robotic arms: a monobloc system (X-ray tube and generator) and a flat-panel detector. The detector and the X-ray tube are assembled in an ImagingRing™ System, which can be rotated independently by aiming to optimize the field of view [16].

The position of the patient was designed to move or tilt, additionally to have millimeter precision and closely observe the whole patient body during treatment [13]. Due to the construction of the ring and movement of the patient through the couch, the maximum field of view is capable of imaging, and the maximum

precision is expected for releasing of the exposure. In the ImagingRing™ System, the ring is mounted on the rail to simplify the imaging without changing the position of the patient [15].
The ImagingRing™ System at MedAustron has also provided a full Monte Carlo model of this system. This full Monte Carlo model known as phase space contains important characteristics such as position, energy among others.

Figure 2.7: ImagingRing™ System at MedAustron [13]. The X-ray head and detector which are mounted on a robotic arm are positioned below the patient couch

2.4.1 Movement details

The X-ray head can be rotated independently around the patient with a rotational movement of about 480°. The longitudinal movement of the ring is designed 125 cm along the couch with a speed of 10 cm per second.

2.4.2 X-ray head and detector

The X-ray radiation penetrates through a glass layer followed by oil and an exit window made of polycarbonate. This process results in a built-in self-filtration equal to 1.4mm of aluminum. The located X-ray source in the X-ray head pilots a photon energy ranging from 40keV to 120keV, which can be varied to 80-120keV for clinical use. The X-rays are emitted in termed pulses. The time

intervals of these pulses have a defined constant length. For each required exposure to radiation, the number of pulses and their respective time intervals can be determined. The spinning rate is about 0476.21 [14][16].
The XRD detector is a silicon flat panel X-ray detector with no shape and features which is based on CsI scintillator. The XRD 1642 model utilized in the ImagingRing^{TM} System at MedAustron provides a resolution of 1024 x 1024 pixels and high frame rates of up to 100fps. It is equipped with an active sensor area of 41 x 42 cm². The detector can rotate up to 481.5 degrees around the patient [14] [16].

2.5 Monte Carlo methods

The Monte Carlo method was developed in the 1940's, although the idea of MC is traced back to the eighteenth century [1][2]. Modern MC simulation was first used to model random diffusion of neutrons traveling through radiation shielding in the Manhattan Project by Neumann and Ulam [2]. Scientist who worked on this method gave the name "Monte Carlo" after the city in Monaco and its many casinos [1].
The Probability Density Function (PDF) is the base of MC simulation which is used to model the system. A large amount of data will be sampled from the series of PDFs by using random numbers. The technique which is used for sampling the data is the Cumulative Probability Function (CPF) which is defined as an integral of PDF. Then the final data is computed by the result of sampling data [2].
Particle transport is a kind of physical process that can be described by interaction probabilities per unit of distance. The complex and mathematical problems which are comprised of various random events, such as particle transport, can be simulated with the probability density function [3]. Monte Carlo methods are a clarified technique which is used for generation of X-ray spectra.
The phase space is the outcome of sampling the distribution randomly, instead of integrating probability functions which are created by Monte Carlo methods [10]. The phase space files contain the information of particle position, energy, and angular momentum distribution. However, these files contain typically up to several tens of gigabytes of data and are inefficient to use. An inaccuracy that has to be accepted for the sake of simplification is, that in reality X-ray emission is not originated from one certain point, but rather from the irregular shaped area of the anode, called focal spot.

2.5.1 Simulation

Geant 4 (v10.5) is the simulation toolkit that is used in this thesis to create a Monte Carlo simulation model of ImagingRing™ System [16]. Geant 4 is a geometry based tracking software to simulate particles traveling through matter. In other words, the particles can be tracked along their path through a specified geometry. This simulation toolkit was developed by a collaboration of physicists and software engineers from around the world at CERN and was written in C++ [18]. The source code of Geant 4 can be downloaded freely. It is widely used in nuclear physics, high energy physics, accelerator physics as well as medical applications, due to its wide range of functionalities, including electromagnetic, hadronic, and optical processes as well as a wide range of energy from 250 MeV to 1 TeV [41][30].
In Geant 4, each particle is tracked along its way and is simulated individually. During this process, the length of tracks is defined by the parameter called step size, and based on each step, all interactions are evaluated by sampling from the appropriate probability distribution. In order to decrease the computation time, another parameter, called track cut ,is specified by removing the particles which have no chance of reaching the desired detection space. The average length of the removed particles is shorter than their track size [18][41].

2.5.2 GATE

GATE (the Geant4 Application for Tomographic Emission) is based on the GEANT4 framework, which is a toolkit to simulate the physics of radiation [29]. Although GEANT4 offers a large number of patterns and models for simulating, it provides a user-friendly macro mechanism for controlling sophisticated geometries. This feature is considered to be quite helpful for creating brand-new devices in medical applications. In addition, for the image acquisition optimization, it can be used for reconstructions of an image and haphazard reduction of noise [41][29].
GATE is a program written based on C++. The main function is able to explain its core. Based on the core, there is an application layer which is a set of C++ classes. However, the layer for user is designed and made up in an easily operated way in which C++ skills for programming are not needed. It also permits a control through an extended version of the Geant4 for scripting language [29][41].

A device geometry, in GATE, can be modeled in a stepwise fashion through binding objects which are basic and geometrical like boxes, spheres, cylinders. Then, a material for each object is able to be explained. Likewise, one source of particle is supplemented. Particle types along with the distribution of energy, as well as its direction are defined for reflecting the source in a realistic way [41].

3 Machine learning

In Machine Learning (ML) computers learn how to solve problems by trial and error. It became one of the most attractive branches of artificial intelligence. The learning process can be realized by mathematical algorithms and is capable of producing a reasonable output from input data automatically. Machine learning models are designed to generalize the learned experiences in order to predict new data during training. The training process is required to repeat continuously and the final model will be evaluated after the multiple iterations of training [48]. The process of generalization is difficult and has different designs for each type of application. The relation between input and output data can be presented in the function $y = f(x)$, which defines the correlation between two variables x and y. In most types of machine learning algorithms, such as regression and classifications, the function f is required to approximate the value of y for each new value of x. In this case, it is suitable to represent an arbitrary function that is reliable for universal approximation. One of the most popular and universal predictors is an artificial neural network (ANN) which uses a classifier to model the decision [47][49].

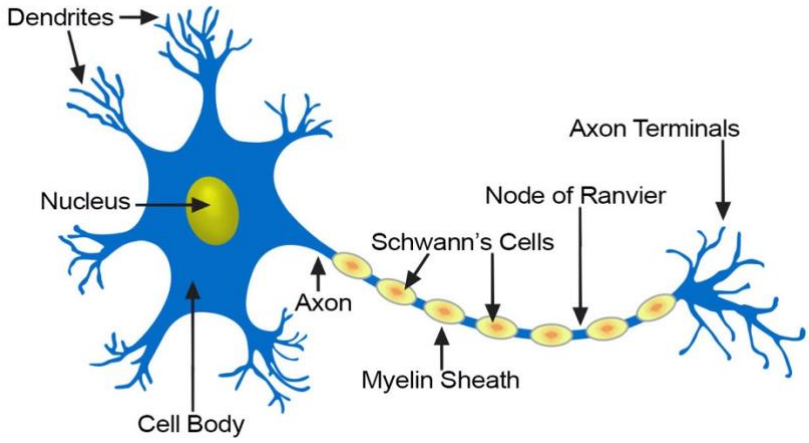


Figure 3.1: A diagram of a neuron [44]. Neurons are inspired by biological nervous system.

3.1 Neural networks

The neuron is the fundamental unit of a Neural Network and the neurons in ANNs are inspired by the biological nervous system such as the brain [51]. Figure 3.1 shows the diagram of a human neuron. ANN consists of simple computing elements that are interconnected together, with the aim to solve complicated patterns. The complexity of problems can be learned by examples. The selected examples must provide useful information, otherwise they cause misguiding and incorrect functionality for the network.

Each neuron in ANN can be considered as a classifier that receives multiple inputs x_1, x_2, … and creates a weighted sum for these inputs using the weight vector w = (w_1, w_2, ..w_n). Besides, the bias b is added to the weighted sum for these inputs. Then a non-linear activation function $h(x)$ is applied to the neuron to model the decision [51]. Figure 3.2 illustrates an example of a neuron which is derived from the human neuron. Each neuron receives the input from the previous neurons through the dendrite, then compares the threshold value through a series of mathematical calculations. If it exceeds a certain limit, an output can be produced. The output reaches further neurons through the axons. There is a synapse between each neuron output and the input of the subsequent neuron [74][51].

$$a = h\left(\sum_i w_i x_i + b\right) = h(z) \qquad (3.1)$$

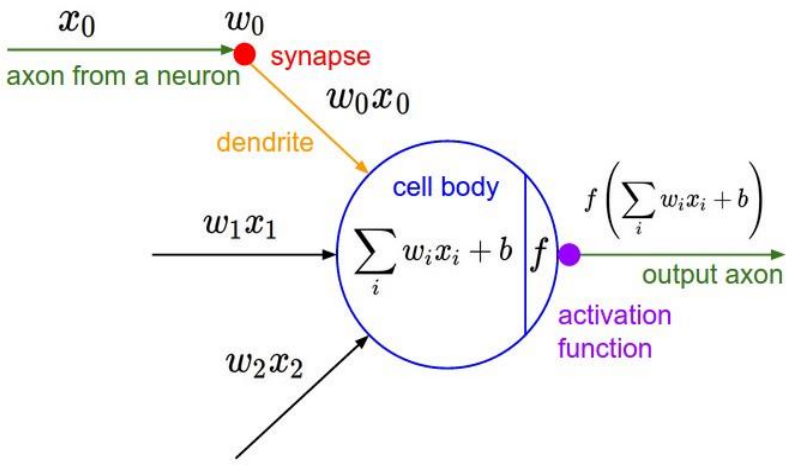


Figure 3.2: An example of a neuron showing a couple of inputs and their corresponding weights, a bias, and the activation function f applied to the weighted sum of inputs [74].

The function (3.1) can be simplified by using the dot product of vectorized inputs $\boldsymbol{x}$ and vectorized weights w instead of the summation function. It can be shown in function (3.2).

$$a = h\,(w^T x + b) = h\,(z) \quad (3.2)$$

All trainable parameters can be summarized as vector θ.

$$\theta = (b_0, w_1, w_2, \ldots, w_n) \quad (3.3)$$

3.2 Feedforward Neural Network

Feedforward Neural Network (FNN) is a type of ANN which is used to approximate the function given an input x [50]. The goal of a FNN is to combine multiple neurons and create the directed associated graph in the form of a layer without cycles. To summarize, each layer can be assumed as a single function consisting of these neurons [50].
Every FNN has a similar architecture which is shown in figure 3.3. It contains three different layers: the input layer, intermediate layer or hidden layers, which contain an arbitrary number of layers of neurons, and the layer computing the output, called the output layer. The data is entered into the input layer and then will be passed on to one or more hidden layers. The predications of networks occur before the data is transferred to the output layer.

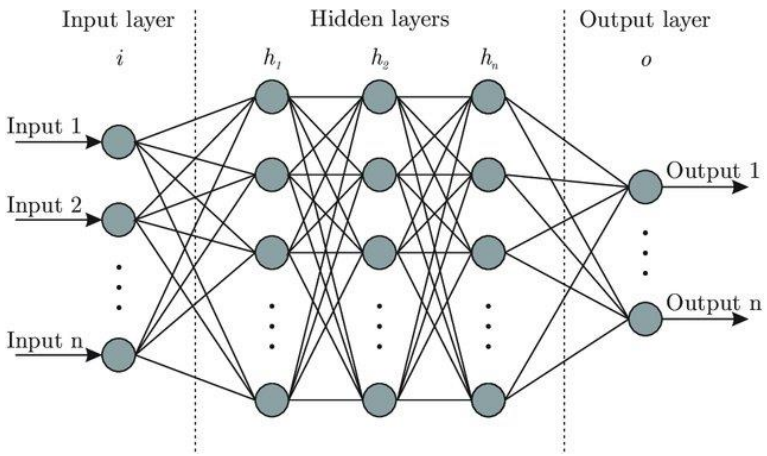


Figure 3.3: Artificial neural network architecture [5]. It consists of three different layers: one input layer, three hidden layers and one output layer.

In artificial neural networks, in order to predict the output of new entered data, the classifier infers the real and correct class y by a new function called estimated function.

3.3 Gradient descent

In neural networks algorithms, to find how well the output for given input x is approximated, the loss function or cost function, which measures the quality of the parameters θ, is defined [51]. The gradient descent is introduced as an iterative technique to find the parameters θ by moving in the direction of steepest descent which is defined as a negative gradient. During this process, the model of performance is gradually increased. The goal of gradient descent is to find the local minimum of the loss function f(θ). The computation of the minimum of the simple loss function would be feasible analytically, but for complex functions, such as loss function of FNNs, it would be complicated [50].
The minimization of loss function f(θ) where θ contains all trainable parameters is the goal of all training processes in NN [50]. For this purpose, the gradient descent which uses the gradient of loss function ∇f(θ) for all trainable parameters θ is used. During this process, the model performance is gradually increased by sequentially updating the θ. The values in the vector θ and the f(θ) are constantly changed during full batch training for regular gradient descent, although the f(θ) remains static for the same value of θ. In the full batch learning, the gradient of each training set is computed independently, then they are summed up together.
Gradient descent can be also introduced as a useful technique in NN which requires the time-consuming computation of all gradients for whole training examples in one single update step. The Stochastic Gradient Descent (SGD) is also defined to increase the speed of the learning process, which corresponds to mini-batch learning. At each update step, SGD estimates the gradient of the cost function, denoted as f_t. As a result, each single update in SGD is less time-consuming in comparison to the full batch learning. The other advantage of SGD is that it is capable of performing a frequent update which causes the function to fluctuate and converge to the minimum. This fluctuation provides the condition in which the function jumps to the better local minima [58][59].

$$\theta_{t+1} = \theta_t - \eta \nabla f_t(\theta_t) \quad (3.4)$$

Here, θ_t denotes the trainable parameter θ at time step t and η denotes the learning rate. It can be written as:

$$\Delta\theta_t = -\eta\nabla f_t(\theta_t) \quad (3.5)$$

when $\Delta\theta_t = \theta_{t+1} - \theta_t$ [17].

3.3.1 Momentum

As described in section 3.3, gradient descent is used to find the minimum of the loss function. By using the regular SGD, the $\nabla f_t(\theta)$ may change strikingly for subsequent steps. This can lead to a large oscillation of the values in θ overtime which causes the slowing down of convergence [47]. To counter this, a momentum term can be added to equation 3.5. The major advantage of momentum is, that the short-term memory, called acceleration, is used to boost the learning process [17].

$$\Delta\theta_t = \mu\,\Delta\theta_{t-1} - \eta\nabla f_t(\theta_t) \quad (3.6)$$

Where μ is he momentum coefficient which holds a value between 0.5 to 1[47].

3.4 Adaptive Learning Rate Optimizers

SGD is defined as a powerful optimization technique for training NNs; however, the choice of learning rate can influence the result of optimization. If η is chosen too large, the training might fluctuate and skip over desired local minima. If it is too small, the learning rate causes significant delays of the convergence process. To deal with this, a common technique, called learning rate decay, is utilized [54]. This method is a hyperparameter in itself, which needs to be designed for each network. It also allows a larger η at the beginning of training and a smaller η towards the end of training. Therefore, η must be tuned by some factors every

few epochs. The aim of an adaptive learning rate optimizer is to find the best learning rate depending on the application. There are a couple of methods which are defined in the following subsection and in these approaches, the learning rate is not a global variable. They need the hyperparameters tuning such as Resilient backpropagation (Rprop) [61], Root Mean Squared backpropagation (RMSprop) [59] as well as Adaptive moment estimation (Adam) [62].

3.4.1 Rprop and RMSprop

Rprop training algorithm aims to eliminate the negative effects of the magnitude of the partial derivates. During this process, the direction of the weight update is determined only by the sign of the derivates while the magnitude of derivate is defined separately without any effects on the weight update [61]. This technique is known as a robust algorithm in ANNs and corresponds to full-batches. The main disadvantage of Rprop is, that it does not work when training with mini-batches. To counter this, RMSprop is an adaptive learning rate method which is a mini-batch version of Rprop, proposed by Geoff Hinton in his course [57]. It works by aiming the normalization of the gradient for all parameters θ, at each update. In order to improve learning, RMSprop divides the gradient descent for each learnable parameter by the accumulated magnitude of the gradient [59].

$$\Delta\theta_t = -\eta \frac{\nabla f_t(\theta_t)}{v_t} \quad (3.7)$$

where $$v_t = \beta_2 v_t - 1 + (1 - \beta_2)\left(\nabla f_t(\theta_t)\right)^2 \quad (3.8)$$

Here β_2 is an additional hyperparameter called decay rate. The division and squaring operations denote their elementwise versions.

3.4.2 Adam

Adam is an SGD method which estimates the first and second-order moments of the gradients by computing individual learning rates for different parameters [62].

Adam algorithm consist of three hyperparameters including the learning rate (η) and decay rate for first- and second-order moments, which are defined by β_1 and β_2. v_t, m_t: first- and second-order moments are given in the equation 3.8 and 3.9.

$$m_t = \beta_1 m_t - 1 + (1 - \beta_1) \nabla f_t(\theta_t) \quad (3.9)$$

When mt and vt values are loaded as vectors of zeros, they are biased towards zero. This is especially the case when t or their respective decay rates are little in amount. Thus, the algorithm uses bias corrected variants of these moments:

$$m_{tt} = \frac{1}{1-\beta_1}$$

and

$$v_{tt} = \frac{1}{1-\beta_2}$$

The update rule for Adam becomes:

$$\Delta \theta_t = -\eta \frac{m_{tt}}{v_{tt} + \varepsilon}$$

Here, ε is a small constant that avoids division by zero.

3.5 Activation function

Activation functions (AF) are non-linear functions that are applied to the weighted sum of the inputs. It is added to each neuron in the ANN and denotes whether the neuron is activated or not. The activation function is the only source of nonlinearity in the NN and has sufficient ability to tackle the complex algorithms and learn challenging data.

3.5.1 Sigmoid function

One of the simplest and most popular activation functions used in this work is the sigmoid function $f(x) = 1/(1 + e^{-x})$. The output range of this function is between 0 and 1, where 0 represents tthat the neuron is not activated at all, while 1 denotes the activating rate at maximum level. The main disadvantage of using the sigmoid function is, that for a large value of x, the gradient will vanish and cause the slowing down of learning, if x is placed in the saturated area. The other drawback is defined because it is not zero-centered and causes inadequate results during gradient descent [50]. Although the sigmoid function is not a useful method for neurons in hidden layers, it can be interpreted as a predictor of probabilities in output neurons.

3.5.2 ReLU variants

Another popular activation function is the Rectify Linear ReLU [52] (ReLU) which is defined as ReLU(x)=max(0,x). Due to its function, it can result in the faster learning of deep ANNs, with gradient descent in comparison to sigmoid function, since it is not saturated for a large value of x [53]. Due to the form of the ReLU, that is linear and non-saturated, the vanishing of gradient descent in a large value of x is omitted [54]. The main problem in this activation function is that no gradient is provided when x is negative. A new version of ReLU called Leaky ReLU [55] is presented and defined in the following to solve this problem:

$$f(x) = \begin{cases} x & if\ x \geq 0 \\ 1/\alpha x & otherwise \end{cases}$$

Where α can be set for different layers in NNs and if it sets to a large value, such as 100, it behaves similar to the original ReLU. The small value of α causes an increment accuracy in classification [56]. Figure 3.4 shows the two different types of activation functions, the right figure represents RELU and the left one represents the sigmoid function.

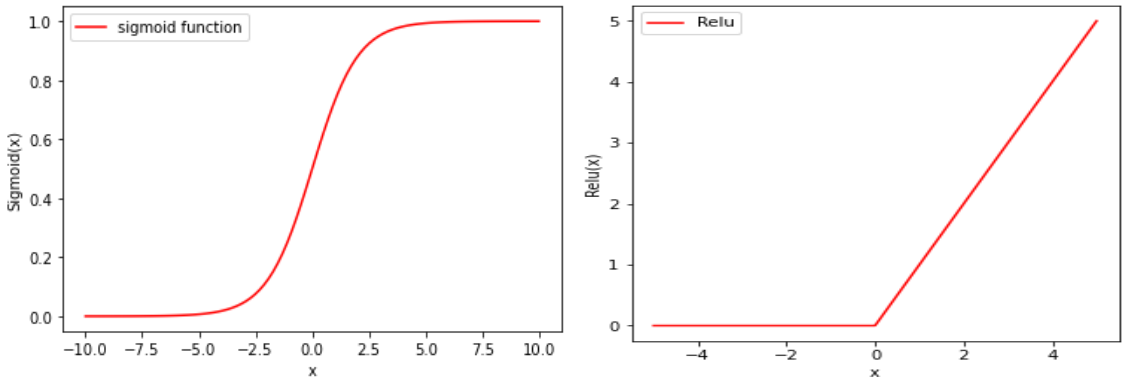


Figure 3.4: The right figure represents RELU and the left one represents the sigmoid function. The popular activation functions used in ANNs. Own work by Matplotlib[1] library.

[1] https://matplotlib.org

4 Generative Adversarial Networks

The idea of Generative Adversarial Networks [63] was proposed by Goodfellow in 2014 as a method for data generation. GAN consists of two networks: Generator (G) and Discriminator (D), which are often implemented as NNs. In this code, G and D designed in a three-hidden layer network.

In all generative models, the Gaussian noise distribution z is applied to $G(z;\theta_G)$ to model the distribution and produce fake data, in our case the distribution of particles. D is also designed to estimate the probabilities of realness and fakeness of a sample data. In order to make a convenient training for each model, the equilibrium between two models has to be found and maintained.

Training of GAN is a challenging process. Generative and Discriminative model are training simultaneously in a so called zero-sum game. In a zero-sum game with two players, in this case the D and G, act as maximizer and minimizer, striving to increase their own score at the cost of the other one's. The Generator attempts to maximize the probability of its output deceiving the discriminator. The discriminator on his part has to keep the probability as low as possible [64]. Ideally, an equilibrium between the performance of G and D is reached, called the Nash Equilibrium [63][80]. It is mathematically described in (4.1)[63].

In case the performance of G and D is unbalanced, the entire training process might stagnate with the output being improved, resulting in a vanishing gradient.

This expression or the GAN purposes in the following defines the complete work of the network:

$$\min \max v(D,G) = E_{x\sim Pdata(x)}[log\ D(x)] + E_{z\sim \mathrm{Pz(z)}}[\log\,(1-\mathrm{D(G(Z))})] \quad (4.1)$$

Where:

$P_{z(z)}$: Denotes the noise distribution which is considered as input of G
$P_{data(x)}$: Denotes the real data distribution
x, z : Denote the inputs of Discriminator and Generator

The aim of the mentioned formula, which is a function of both G and D, is to maximize the D loss, or the log D(x) and minimize the G loss, or log (1- D(G)). In this case, expected probability from various outputs of real and fake data are defined as: E_x, E_z.
Besides, this $v(D, G)$ is the summation of expected probabilities for real and generated data.

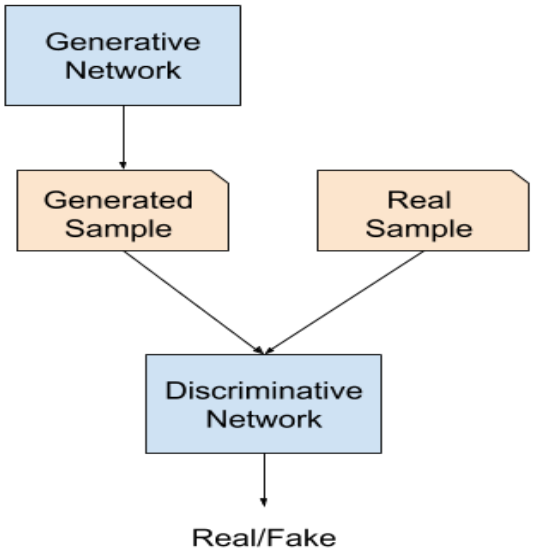


Figure 4.1: A simple graphical presentation of the GAN setting. The generator produces the fake data from the random noise which must convince the discriminator. The discriminator gets input as either real data or generated data to distinguish whether its input is real or fake [79].

More than that, for training both discriminator and generator, resent flaws are the gradients of the following function loss are sent back into the model.

Updated discriminator rule:

$$\nabla \theta_d \frac{1}{m} \sum \left[\log D \left(x^{(i)} + log \left(1 - D \left(G\left(z^i \right) \right) \right) \right) \right] \quad (4.2)$$

Updated generator rule:

$$\nabla \theta g \ \frac{1}{m} \sum log \left(1 - D \left(G\left(z^{(i)} \right) \right) \right) \quad (4.3)$$

Where m shows the whole number of tested samples in batch prior to being updated and with both θ_d and θ_g showing each model weight.

4.1 Iterative process of GAN

The purpose of this section is to explain the process of training. Every step of one training process is known as epoch, a term in machine learning, indicating the number of passes of the whole training dataset through the system. Throughout every epoch data are processed in a batch, a random fixed size subset (called batch size) from the samples in the dataset which are real. The number of epochs and the number of iterations is the same, given that the entire training dataset file in one batch [78]. Steps inside each batch are as following:
Firstly, the discriminator receives a real data batch. After that, incoming errors from the discriminator output are worked out. Simultaneously, the generator produces a fake data batch, using noise vectors batch. In the third place, the discriminator receives fake data batch for the second time. Next, the discriminator output is calculated either (0) or (1), meaning fake data and real data, respectively. Both of these kinds of errors calculated by the discriminator are then propagated to the model. Now, errors for the generator are supposed to be calculated, meaning that outputs coming out of the discriminator ought to be 1, because they are ideally imitated forms of real data. Output errors are supposedly to be propagated to the generator model and this cycle is again restarted with another batch from this epoch [78].

4.2 Difficulty of training GAN

Training of regular GANs is always accompanied by instability. The cause of this instability is the fact that the generator and discriminator are designed with different contradicting goals. This can prevent each other's progress during training [64]. Additionally, when the generated data is similar to only a part of the real data distribution, failure occurs. This failure mode is called mode collapse [64]. Depending on the severity of the mode collapse, the training process could be considerably compromised. When mode collapse occurs to a mild extent, the model still converges and the training process is applicable. In a severe case of mode collapse, only a few samples or even just a single sample is generated, which is far from the goal of GAN [65].

4.3 Wasserstein generative adversarial nets

The goal of the original GAN is to minimize the JS divergence in a min-max game [63]. In most cases, the JS divergence will not provide the beneficial gradients and cause it to fail to converge. To counter this, Wasserstein GANs [66][67] were proposed. WGAN minimize the Earth mover distance instead of minimizing the JS divergence between two probability distributions. Besides, the training of WGAN is leading to improve the original GAN in different ways and cause robustness to mode collapse [66]. The introductory of this section will be started by an introduction of Earth Mover (EM) distance. Then this section will be finalized by the explanation of WGANs.

To find the perfect P_θ density model in the given real data x, mastering probability distribution using maximum probability estimation is vital. Considering the real data distribution p_R and the model distribution p_G, the restriction (amount of real data) of the maximum likelihood estimate is equivalent, so as to minimize the Kullback-Leibler (KL) divergence $KL(p_R||p_G)$. Nonetheless, in case P_θ is not obvious, the KL divergence is inexplicable or infinitive in this regard. For example, if the distributions are governed by manifolds dimensions, so the intersection of model manifold, as well as real distribution support, will not occur [67][82]. That is why various generative models strive to put an extra noise term to the model distribution, which results in indistinct generated data. To counter this, GANs can be trained by randomly feeding variable z through a parametric function G, usually applied as a neural network. Therefore, through taking

samples directly from the model distribution p_G, and changing the model parameters θ_G, adjusting p_G to be as similar as p_R is possible real distribution p_R.

The major advantage to use WGAN is the different approach, so that distances between the model distribution p_G and the real distribution p_R can be measured. In the authentic formulation for GANs, which is proposed by Goodfellow et al., the aim function of GAN for optimizing is the Jensen-Shannon (JS) divergence. In WGAN, the Earth-Mover (EM) distance or Wasserstein-1 distance is utilized to estimate distance function, which results in computing the optimal transport plan loss for transforming the distribution p_R into p_G. The Wasserstein-1 distance is explained as following:

$$W(p_R,p_G) = \inf_{\gamma \in \Pi (p_R ,p_G)} E_{(x,y)\sim\gamma} [||x-y||], \quad (4.4)$$

$\Pi(p_R, p_G)$ is the number of all binding distributions $\gamma(x, y)$ when the amount for p_R and p_G, is small respectively. As the infimum in Equation 4.4 is out of control, the binary form of the Wasserstein-1 distance [69][82] is utilized to collect the scheme optimization:

$$W (p_R, p_G) = \sup_{||D||_L \leq 1} E_{x\sim p_R} [D(x)] - E_{x\sim p_G} [D(x)] , \quad (4.5)$$

When 1-Lipschitz functions are less than the supremum, this constraint can be alternated with a K-Lipschitz theory. The binary form only undergoes changes by a multiplicative constant of K, that does not alter the optimization problem. Here K is able to be absorbed into the learning rate hyperparameter. In case D functions family, which are parameterized with parameters $\theta_D \in W$, has functions, which are all K-Lipschitz, the subsequent optimization problem, leading to the primary training objective for WGAN, can be obtained. Theoretically speaking, the supremum in Equation 4.5 can be gained for some $\theta_D \in W$ and through merging the generator network G(z) as the model distribution p_G. The equation can be reformed to reach at:

$$\max E_{X\sim p_R}[D(x)]-E_{Z\sim p_Z}[D(G(z))], \quad (4.6)$$

$$\theta_D \in W$$

By the time it is optimized, the Wasserstein-1 distance can show greater features in comparison to the JS divergence, as distributions, which are supported by low dimensional manifolds, are learned. The result of such events is, that WGAN has the capability for learning likelihood distributions in which other learning goals derived by the JS and KL divergences fail to do so. Because of the benefit of EM distance, to have two main advantages as being continual and distinguishable, the functionality of the discriminator is proposed once by Arjovsky et al. to be called as a trained critic as it can produce valid gradients. Moreover, while the discriminator improves in the standard JS objective of the original GAN, causing the JS distance to be saturated locally, brings about disappearance of gradients [81][82], as opposed to the WGAN having gradients elsewhere. It is recommended that WGAN can tackle the collapse mode problems, because of the critic which is about to be trained prior to optimality. As well as that, for approximating EM distance, the optimized critic will give permission for a relevant explanation of the critic loss [82].

4.3.1 Gradient penalty

The existence of some factors can cause quality of samples to be lost as well as a lack of full convergence, whereas GANs training are being enhanced by WGAN. Gulrajani et al. believe these modes failed because of the way the 1-Lipschitz restriction is put in the original WGAN. The weight clipping presented in the original WGAN means, that the critic has difficulty finding the functions optimality, thus its capacity is underused. In addition, norms of gradients are affected due to weight clipping, which might also cause disappearance or explosion of gradients. Hence, gradients are unsteady as a result, which could cause the training to be inefficient and unstable [70]. They also provide some proof that the critic in the framework of the WGAN, which is optimal, has a normal unit gradient almost everywhere under p_R and p_G. This can differently encourage them to impose the 1-Lipschitz restriction of the original WGAN, through utilizing gradient penalty on the critic gradient. Using WGAN-GP (gradient penalty) on the contrary to the weight clipping can help to implement that Lipschitz is only continuous if the critic is constrained [70][71]. WGAN-GP

reaches this through utilizing a gradient penalty to the critic function value. WGAN-GP is far better in terms of performance than the regular WGAN because of its various generator and critic forms [71].

The result of that is that the gradient norm is motivated towards 1, forming the ground for the Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) training procedure [70]. Gulrajani et al., for this gradient, take a uniform sample from a distribution $p_{\hat{x}}$, defined along the lines between sample pairs from both the real data distribution p_R and the model distribution p_G. When applying this gradient penalty, the original WGAN loss function is altered as shown below:

$$L=E_{x\sim p}[D(x)]-E_{z\sim p_Z}[D(G(z))]+\lambda E_{\hat{x}\sim p}(\|\nabla_{\hat{x}}D(\hat{x})\|-1)\,, \quad (4.7)$$

As it is shown above, λ introduces the penalty coefficient, thus, they discovered it can perform well for a number of experiments if the value of λ is 10.

5 Method

This chapter, first, outlines the most important soft and hardware that used to develop, train and evaluate GAN in this thesis and to prepare input data. Then it is followed by a comprehensive explanation of the employed GAN details and the ending part of this section will be finalized by the introduction of the evaluation of the used methods.

5.1 Used Technology

In order to describe the complete documentation of used technology, a short introduction of popular tools is given below.

Python[2] is a high-level and interactive programming language which is designed by Guido van Rossum and released in 1991. Python allows to execute the codes without the need to compile it. Due to powerful libraries, Python is used widely in many scientific projects. Over the recent years, python has been the most used language to develop Neural Networks and deep learning algorithms.

NumPy[3] plays an important role for scientific computing in python. In this master thesis, its features were used for editing, removing unnecessary data and in evaluating n-dimensional array of objects.

Pandas[4] is a software library for manipulation and analysis of data in python. It provides the data frame by aiming to simplify data analysis and manipulating of large numerical tables. The output matrix of Neural Networks can be evaluated by “pandas”.

[2] https://www.python.org
[3] https://numpy.org
[4] https://pandas.pydata.org

Matplotlib[5] is the comprehensive library in python which is used to create static, interactive and animated visualization of numerical data.

Pytorch[6] is an open-source machine learning library which provides two high-level features: tensor computation with strong GPU acceleration and deep neural networks built on a tape-based autograd system (automatic differentiation).

Scikit-Learn[7] is a software library for machine learning. It provides many convenient methods for preprocessing data and evaluating results.
Principle Component Analysis (PCA) is the statistical technique which converts high dimensional data to low dimensional data by selecting important features. PCA can be implemented by using python's Scikit-Learn library.

Root[8] is a high-performance and open-source software which is written in C++. Root has been integrated over years with python and the dynamic and unique connection between python and C++ is considerably enhanced. Analyzing and visualizing a large amount of data can be executed by root. Thus, the phase space data, like the one processed in this thesis, is usually stored in root format files.

Uproot[9] allows to read and write ROOT files using only Python and Numpy. The standard uproot is only an input/output (I/O) library. Uproot does not depend on C++ root, instead it unlocks data from root files as Numpy arrays. In the last few years, uproot has been widely used in different branches of physics, specifically in particle physics.

Colab notebooks are Jupyter Notebooks which run on a cloud and provide GPU-powered notebooks for free. It is highly integrated with Google Drive. **Jupyter Notebook**[10], an open-source web application, prepares a practical learning environment which simplify the running, modifying and analyzing data in deep learning and artificial intelligence.

[5] https://matplotlib.org
[6] https://pytorch.org
[7] http://scikit-learn.org
[8] https://root.cern
[9] https://pypi.org/project/uproot/
[10] https://jupyter.org

All training of the ML was done on a computer with following specification in table 5.1:

Table 5.1: Hardware used for training the data by GAN with the high provided GPU of Google Colab.

Type	Model
CPU	1.8 GHz Dual-Core Intel Core i5
GPU	12 GB NVIDIA Tesla K80 (based on Google Colab
RAM	8 GB 1600 MHZ DDR3
Hard Drive	1 T
Operating System	MacOS 10.15

Thus, the given training times for the GAN in section 5.2.2 are referring to this set of hardware. The most effective hardware part is a GPU, which was provided remotely by Google Colab (about 10 hours each time). Finding the best input data for training from main datasets, which was done in Jupyter Notebook, was time-consuming. The splitting of the large phase space into the smaller files took around 36 hours. In addition, this process was not feasible in Google Colab with fast GPU, because the amount of the reference dataset (50GB) was higher than the free space provided by Google drive (15GB).

5.2 Used Method

The code used in this thesis [73] is the Generative Adversarial Network written by Sarrut, which was extracted from GitHub to model the large phase space files used in Monte Carlo Simulation [68].
In order to describe the complete documentation of the used method, a short introduction of my approach, given in the subsections below, consists of the pre-processing the dataset, training the data and tunning the parameters.

5.2.1 Training data

A large phase space dataset of around 50 GB in root file format was provided. It was produced by a Monte Carlo model of ImagingRing™ System, containing a

set of parameters including position, energy, direction, weight, etc. After removing unnecessary details, the rest of data should be converted into the numpy files to be used for the GAN training. The numpy file size was selected to be convenient to train GAN. The final training datasets used in the training part was around 1.4 GB which contained 4.8×10^7 of the particles. A screenshot of training data extracted from a Jupyter Nootebook is shown in figure 5.1.
The details of the particles are input parameters (seven). The first one for energy, the next three of each particle for the particle position, the last three ones for the direction of particle.

```
In [15]: 1 df=pd.DataFrame(data=C,columns=C.dtype.names)
         2 df
```

Out[15]:

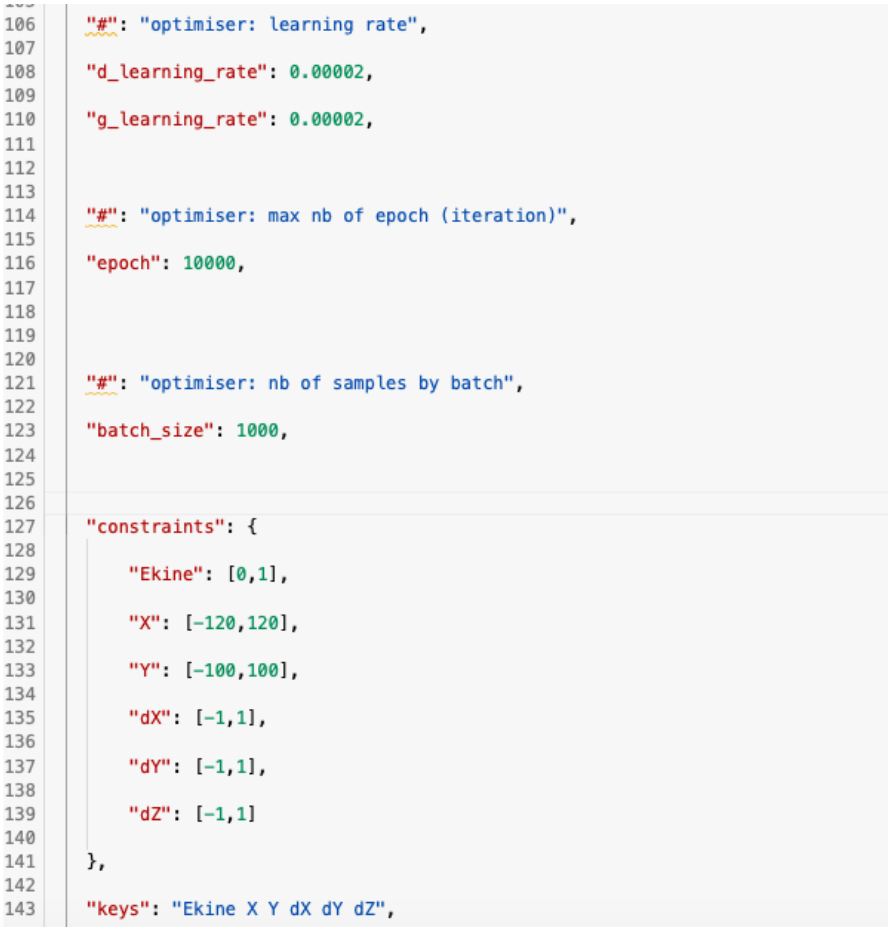
	Ekine	X	Y	Z	dX	dY	dZ
0	0.032058	45.027584	7.428838	540.005005	0.561044	0.091002	-0.822769
1	0.036867	-45.081795	12.311944	540.005005	-0.464899	0.128622	-0.875971
2	0.040421	39.591923	-25.274616	540.005005	0.606988	-0.579361	-0.543973
3	0.044133	-8.746800	-22.260082	540.005005	-0.100999	-0.250439	-0.962850
4	0.046160	-24.896723	7.216946	540.005005	-0.281961	0.083027	-0.955827
...	...	...	...	...	...	...	...
23999995	0.030102	14.531175	15.869899	540.005005	0.116769	0.210399	-0.970617
23999996	0.039191	22.223034	-10.449576	540.005005	0.251940	-0.116333	-0.960725
23999997	0.030377	-0.247046	27.979050	540.005005	-0.070833	0.403236	-0.912350
23999998	0.029725	1.482368	9.687625	540.005005	0.017985	0.115148	-0.993185
23999999	0.032626	-25.475288	-9.827997	540.005005	-0.284956	-0.108356	-0.952396

24000000 rows × 7 columns

Figure 5.1: A screenshot of training data obtained from Jupyter Notebook. Panda package was used to generate the data frame to be convenient to evaluate.

5.2.2 GAN architecture and parameters

Training of our GAN requires a json file described by implementing a set of dependent hyperparameters. A screenshot of the json file is shown in figure 5.2. Some of these parameters are designed empirically by trial and error. The architecture of both Generator and Discriminator Network is the following: 400 neurons for each of the three hidden layers. The value of weights and biases was set empirically based on literature data [68].



```
"#": "optimiser: learning rate",

"d_learning_rate": 0.00002,

"g_learning_rate": 0.00002,


"#": "optimiser: max nb of epoch (iteration)",

"epoch": 10000,



"#": "optimiser: nb of samples by batch",

"batch_size": 1000,


"constraints": {

    "Ekine": [0,1],

    "X": [-120,120],

    "Y": [-100,100],

    "dX": [-1,1],

    "dY": [-1,1],

    "dZ": [-1,1]

},

"keys": "Ekine X Y dX dY dZ",
```

Figure 5.2: A screenshot of the json file obtained from GAN's code [73]. All the hyperparameters used for the training, existed in json file. Note the intervals (constraints) of the six parameters which have been specified.

Two different types of activation functions are used in this WGAN, A Rectified Linear Unit (Relu) and sigmoid function. The sigmoid function is just used in the last hidden layer in generator. The total number of weights for both Generator and Discriminator is around 5×10^5 [68][66]. The RMSProp optimizer [59] has been used instead of the conventional Adam optimizer [62] to avoid instability during the training process [68][66].

The batch size is set to around 10^3 per iteration and the number of epochs is set to 10^4. Each iteration took around 3 seconds, using Google Colab one run took about 10 hours in total. The learning rate is the most important part during training, I used 2×10^{-5} to achieve the best superimposition of fake data on real data. Figures 5.3 - 5.8 show the marginal distributions of the six parameters gained from the reference phase space and from the GAN produced by the described hyperparameters, whereas one of the parameters of the input is considered constant. The 10^4 particles obtained from Gaga_Generate are

compared to original particles. The bin size is set to 200. The mentioned figures have been obtained from Gaga_plot code which is described in code description.

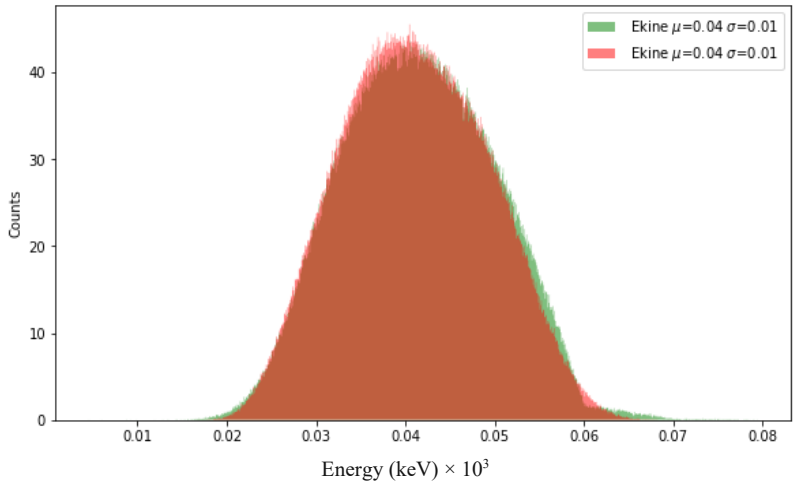


Figure 5.3: Histogram of the photon beam energy for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using 10^4 photons and 200 bins. Mean and standard deviation for two distributions are indicated in the figure legend. Note the similar shape of the distributions.

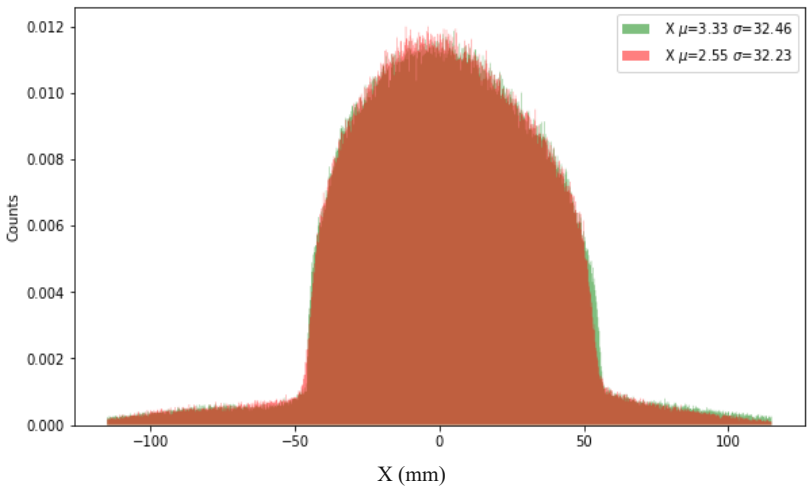


Figure 5.4: Histogram of the position (X) for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using 10^4 photons and 200 bins. Mean and standard deviation for two distributions are indicated in the figure legend. Note the slight differences in the mean value of two distributions.

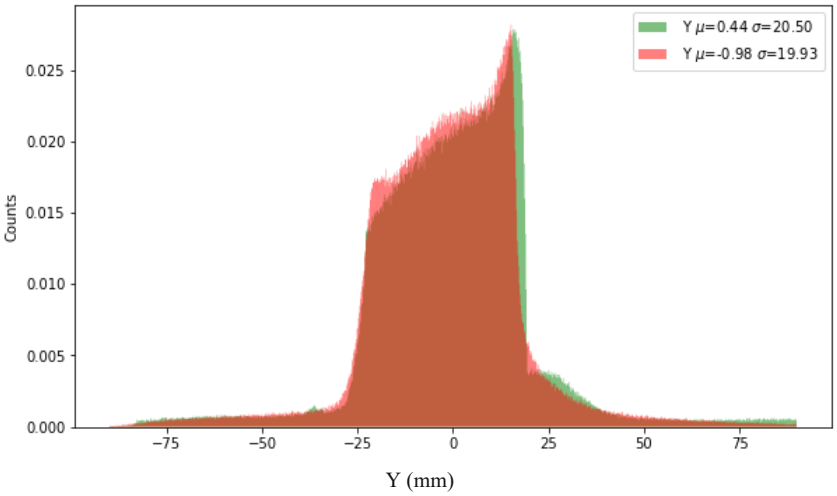


Figure 5.5: Histogram of the position (Y) for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using 10^4 photons and 200 bins. Mean and standard deviation for two distributions are indicated in the figure legend. Note the slight differences in the mean value and standard deviation of two distributions.

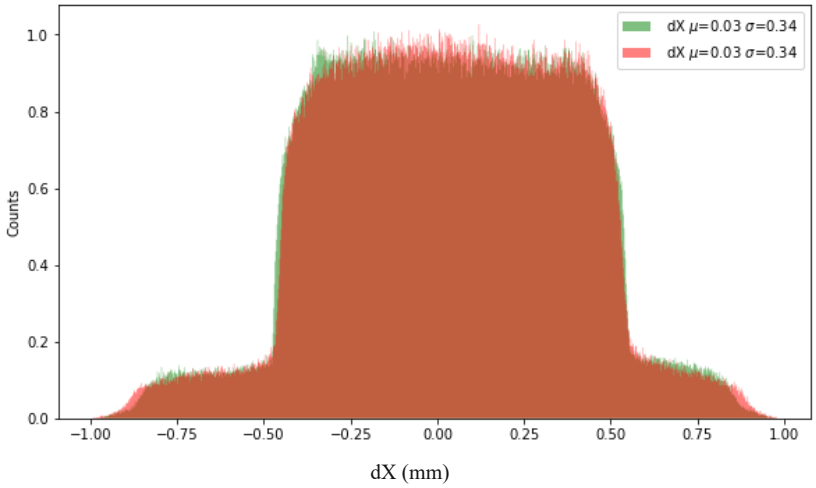


Figure 5.6: Histogram of the direction (dX) for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using 10^4 photons and 200 bins. Mean and standard deviation for two distributions are indicated in the figure legend. Note the similar shape of the distributions.

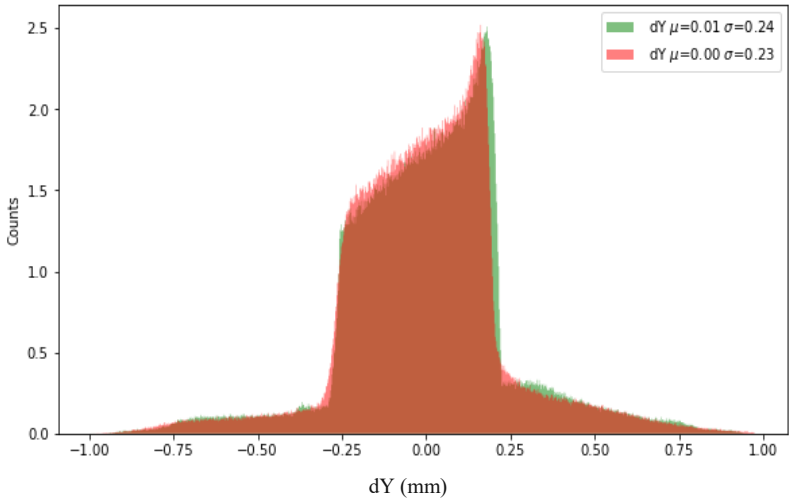


Figure 5.7: Histogram of the direction (dY) for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using 10^4 photons and 200 bins. Mean and standard deviation for two distributions are indicated in the figure legend. Note the similar shape of the distributions.

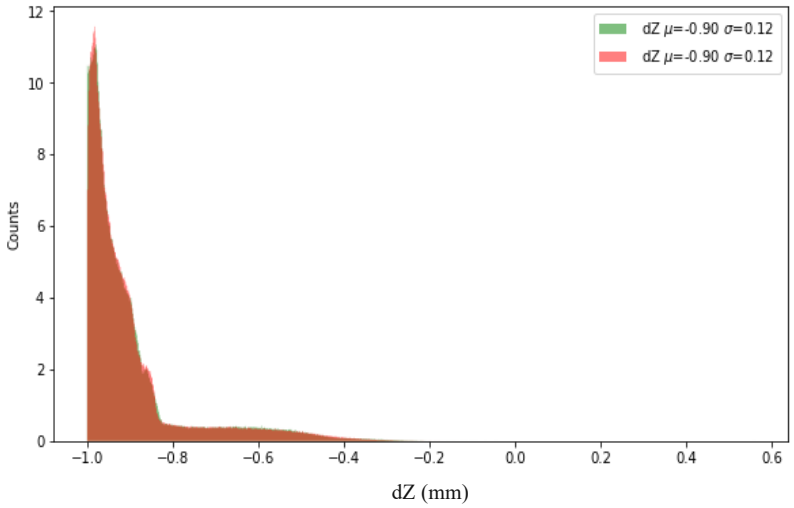


Figure 5.8: Histogram of the direction (dZ) for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using 10^4 photons and 200 bins. Mean and standard deviation for two distributions are indicated in the figure legend. Note the similar shape of the distributions.

5.2.3 Implementation

The GAN for training and the particle generation was a WGAN with gradient penalty. It was implemented in Pytorch with the Pytorch framework [68][83]. The generated particles are produced by running the python's file called Gaga_train which is described in code description.

5.3 Experiments and Evaluation methods

Two phase space files, generated using a Monte Carlo model of the ImagingRing™ System at MedAustron, were provided for this thesis: one file was used for training the GAN and the other for evaluating [68]. The file used for training was split into smaller numpy files. The generated particles can be obtained by Gaga_generate after training the ANNs. Gaga_generate gives this feasibility to get as many arbitrary numbers of particles as needed, whether more or less through using its coding [68].
In order to answer the aim of this thesis question, the generated data must be compared with real data. In this regard, the evaluation part of the implementation in the generated phase space file needs to be compared with the real phase space. Due to the fact that the interpretation of large datasets is demanding and challenging, two different methods were considered to evaluate the results: 1- the visualized process and 2- the numerical process. One way to interpret the visualized process can be defined as Principle Component Analysis.

5.3.1 Principle Component Analysis

Principal Component Analysis [72] is a method which is used to visualize variables of multiple dimensions by creating new variables. These new variables are uncorrelated linear functions derived from the respective eigenvalue/eigenvector problem of the data. They represent the best linear approximation of the given dataset in a lower dimensional subspace. Thus, the variance is maximized and loss of information minimized. The dimension of the dataset is reduced by these new variables, allowing an easier interpretation. This technique reduces the dimension of the data to two in order to project them on two orthogonal axes. Since PCA is also an unsupervised process, the number of features $x_A,..,x_Z$ can be explained and defined [72].
The phase space file in numpy format needs to be converted to a data frame for the first step of performing principle component analysis. The main prerequisite, as for any machine learning task, is that the data has to be scaled. One way to perform the scaling is using the StandardScaler class from the Scikit-Learn package. StandardScaler is used to standardize the input features.

5.3.2 Marginal Correlation

The second type of visualized method is termed Marginal Correlation. In this type, each variable of fake data and real data in a plot is compared to see, if they are superimposed or not. The amount of energy for both real data and fake data is considered stable and constant. The superimposition of real and fake data on each other can be seen for the position of parameters and direction of particles. The number of input data, which is extracted from real and fake data, is about one million particles, and the bin size is set to 200.

5.3.3 Correlation of X-Y plots

Two of seven input parameters were chosen and extracted for both real particles and fake particles. Two characters of particles which are X-Y are plotted for any specific energy range. This method might have a higher loss of information compared to PCA due to the fact that the left parameters are not considered. However, they can show how particles in X-Y coordination behave and are distributed on the surface.

5.3.4 Validation of data with p-value

Karl Pearson was the first pioneer of proposing the P-value in his Pearson's chi-squared test [76]. In order to use this method, a null hypothesis and a level of significance are required. If the null hypothesis is not rejected, it can be concluded the null hypothesis might be statistically significant. But if it is rejected, the conclusion can be drawn that the null hypothesis is not statistically significant [76]. This is a numerical method which shows how similar fake and real data are; however, the code written by myself can be used as a visualized method for various energy ranges or (bins of energy) for different number of particles.
The null hypothesis shows the fact that both real data and fake data are similar and alpha, which defines the level of significance, is set to 0.05, meaning that the confidence level is at 95%. A P-value below this range illustrates the inequality of real data and fake data at the mentioned confidence level. In this case the null hypothesis is rejected and an alternative hypothesis is needed. Each p-value was

obtained from the two-sample Kolmogorov-Smirnov test (ks_2samp) in python with the SciPy[11] library.

5.3.4.1 Visualized p-value

The studies with extremely large samples are associated with problems in which p-value goes quickly to zero and results in no practical usefulness [77]. In order to compare the fake and real data in large scale, the visualized p-value is devised. The function in the visualized p-value was implemented in python code in order to obtain the amount of the p-value for each bin for specified energy range. The mentioned method was used to define the given amount of the energy and to design the p-value plot of the rest of the parameters in different values for bins.

5.3.5 Correlation matrices

The correlation matrix aims to identify the correlation coefficients between variables used as a statistical technique to define the covariance normalized by the product of their standard deviations between the six parameters obtained from training and from the full Monte Carlo model.

[11] https://www.scipy.org

6 Results

The value of discriminator loss at the beginning of training, for 10^4 iterations and 10^3 batch size, for given data (4.8×10^7) is negative and larger than final discriminator loss. As soon the generator is sufficiently trained, the final discriminator loss converges to zero, as it was expected according to previous work [68]. The final D-Loss and G-Loss are shown in table 6.9.
The first result of training the particles, as it is shown in figure 5.3 - 5.8, illustrate the distributions of the six parameters (Ekine, X, Y, dX, dY, dZ). In each figure, the x-axis denotes the intervals of each parameter and the y-axis shows the counts gained from the reference phase space compared to the ones from the GAN. The 10^4 particles obtained from Gaga_Generate are compared to the original particles. The bin size is set to 200. The ideal marginal distribution of each parameter obtained from GAN and reference dataset is supposed to have an equal value for both standard deviations and for both mean values [68]. For each parameter, the obtained mean and standard deviation were obtained almost equal.
Figure 6.1 below shows the overlay of 500 samples from each the real data, produced by the full Monte Carlo model and the fake data, generated by the GAN. The plot was obtained by PCA method as it was mentioned in section 5.3.1. This superimposition is the result of well-matched generated data by training with the real data, for small number of particles.

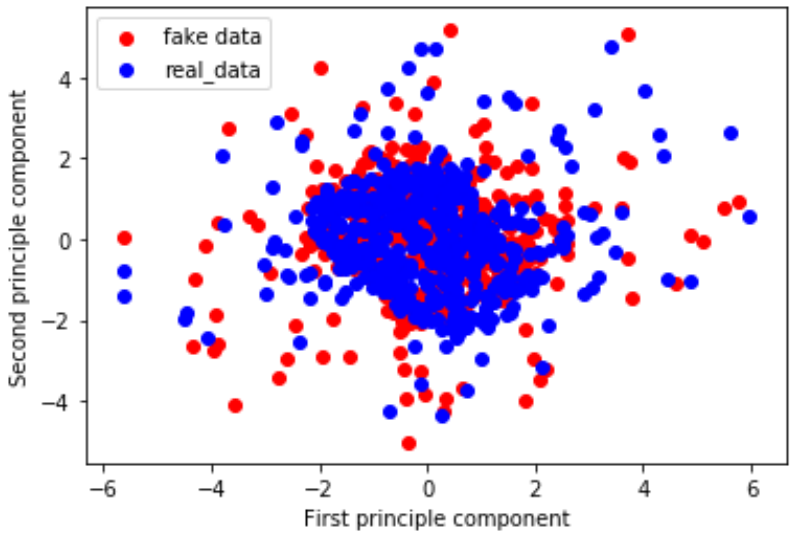
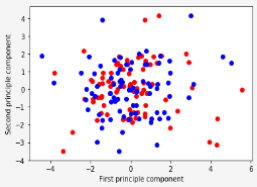
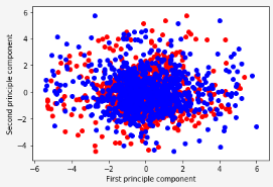
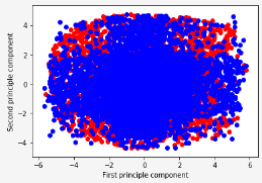
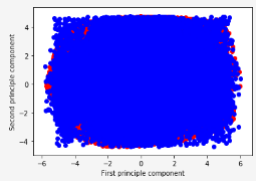
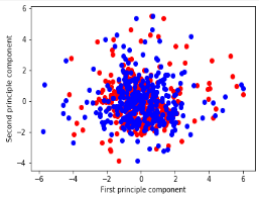
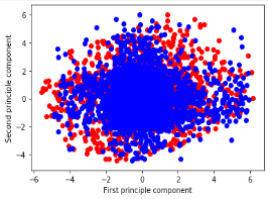
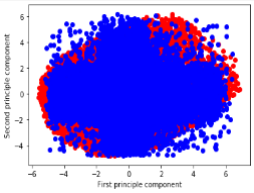
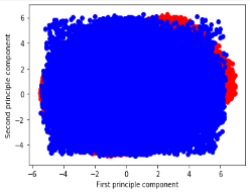
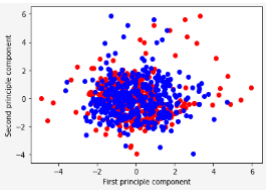
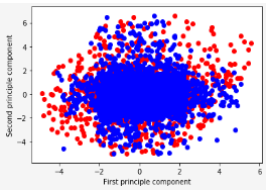
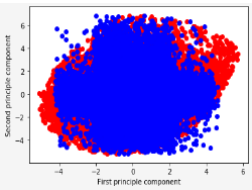
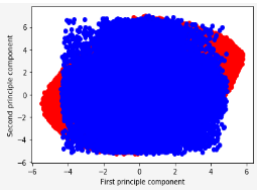
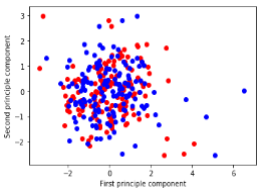
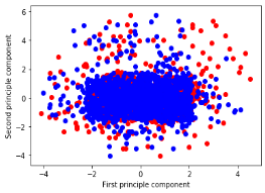
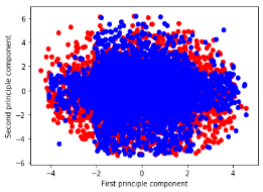
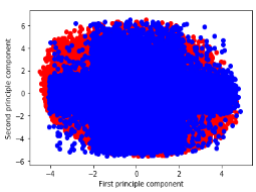
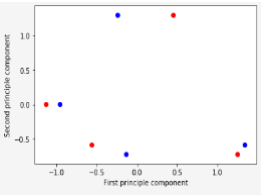
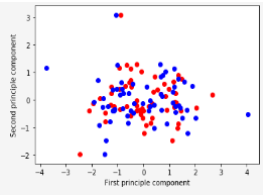
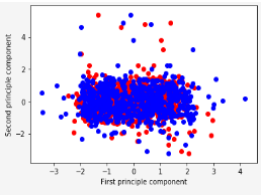
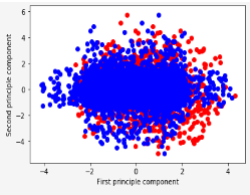


Figure 6.1: PCA plot for GAN generated fake data (red) and Monte Carlo simulated reference data (blue) suing 500 photons. Note the superimposition of real data and fake data. Most of the photons have been projected at the center.

In table 6.1, the PCA method was implemented for different energy ranges. For a higher number of particles, PCA plots demonstrate slight differences between fake data, depicted in red, and real data, in blue. This table also shows that, PCA plots for all different energy ranges, from 20kV to 70kV, for 10^4 number of particles denote the well-matching of real and fake particles.

Table 6.1: PCA plot for GAN generated fake data (red) and Monte Carlo simulated reference data (blue) using (10^3,10^4,10^5,10^6) photons and different energy ranges from 20keV to 70 keV. Note the best matching of the PCA plots for the energy ranges between 20keV to 60keV for small number of photons.

Energy/ particles	10^3	10^4	10^5	10^6
20 keV- 30 keV				
30 keV - 40 keV				
40 keV - 50keV				
50 keV - 60 keV				
60 keV- 70 keV				

In the following figures, the marginal correlation of the six parameters obtained from the reference phase space file and from the GAN is shown. In each plot in the following figures (6.2 - 6.7), the x-axis shows the interval of each parameter and the y-axis denotes the counts of particles. The number of input data, which is extracted from real and fake data, is about one million particles, and the bin size is set to 200.

The plots of parameters in the energy range of 20keV- 60keV in figures 6.2-6.5 show that the fake data and real data are superimposed on each other without considering the slight differences in some areas. The obtained corresponding standard deviation and the mean of each plot are approximately equal. As it is shown in figure 6.6, the superimposition of particles has not been seen, for the energy range of 60keV-70keV.

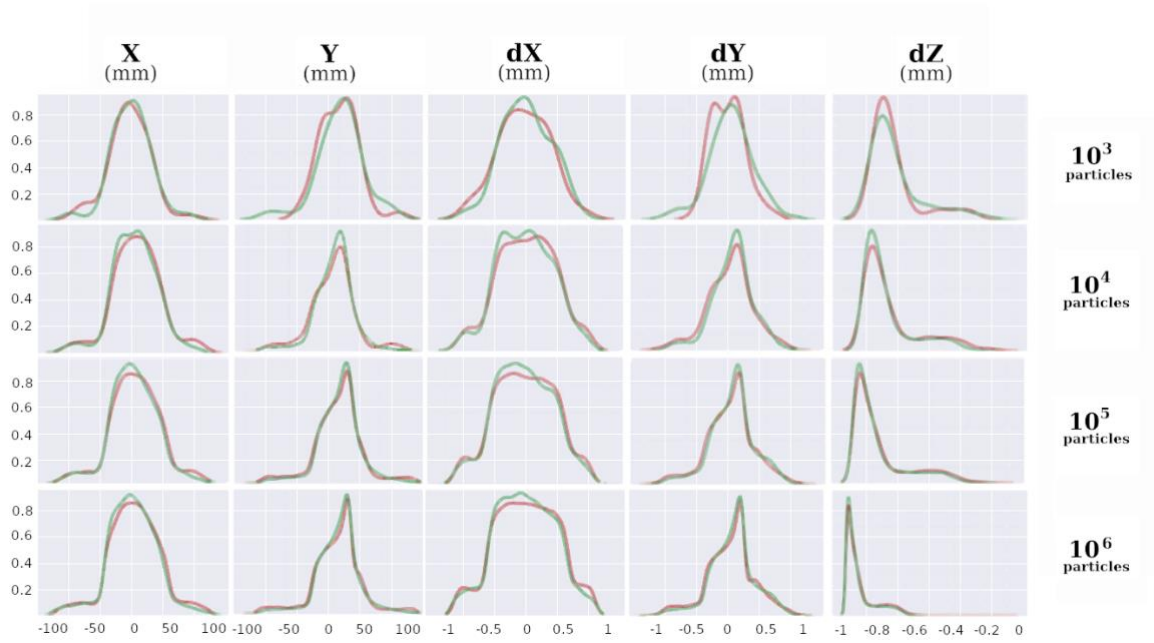


Figure 6.2: Marginal correlation plots for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using (10^3, 10^4, 10^5, 10^6) photons and the energy ranges between 20keV-30keV. Note the slight differences in terms of superimposition of real data and fake data for 10^3 photons.

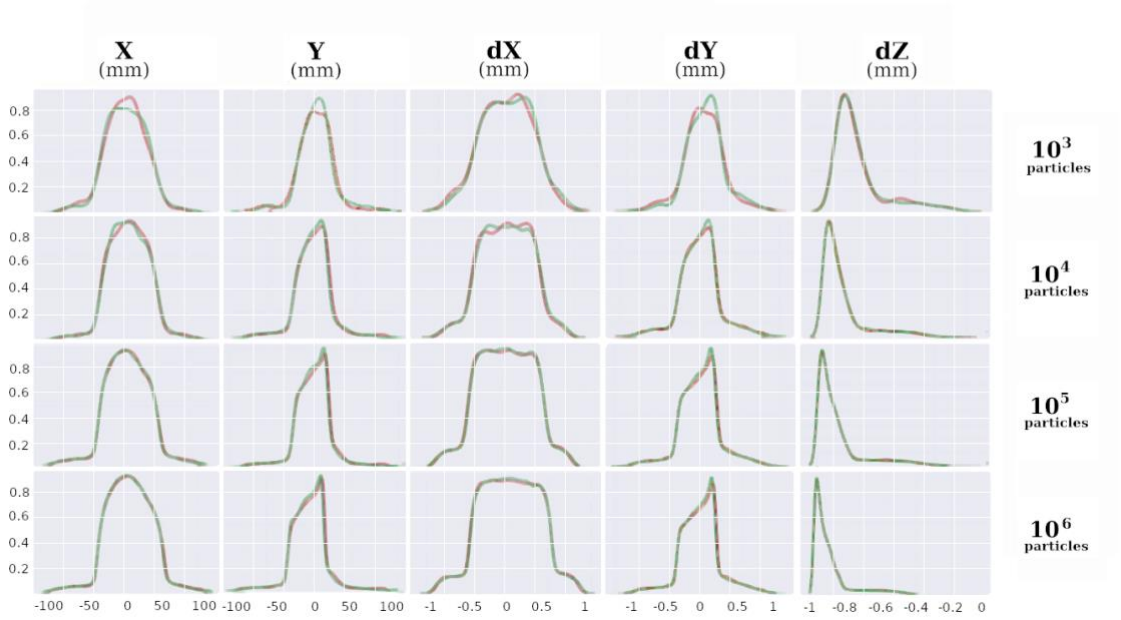


Figure 6.3: Marginal correlation plots for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using (10^3, 10^4, 10^5, 10^6) photons and the energy ranges between 30keV-40keV. Note the slight differences in terms of superimposition of real data and fake data for 10^3 photons (exception for dZ).

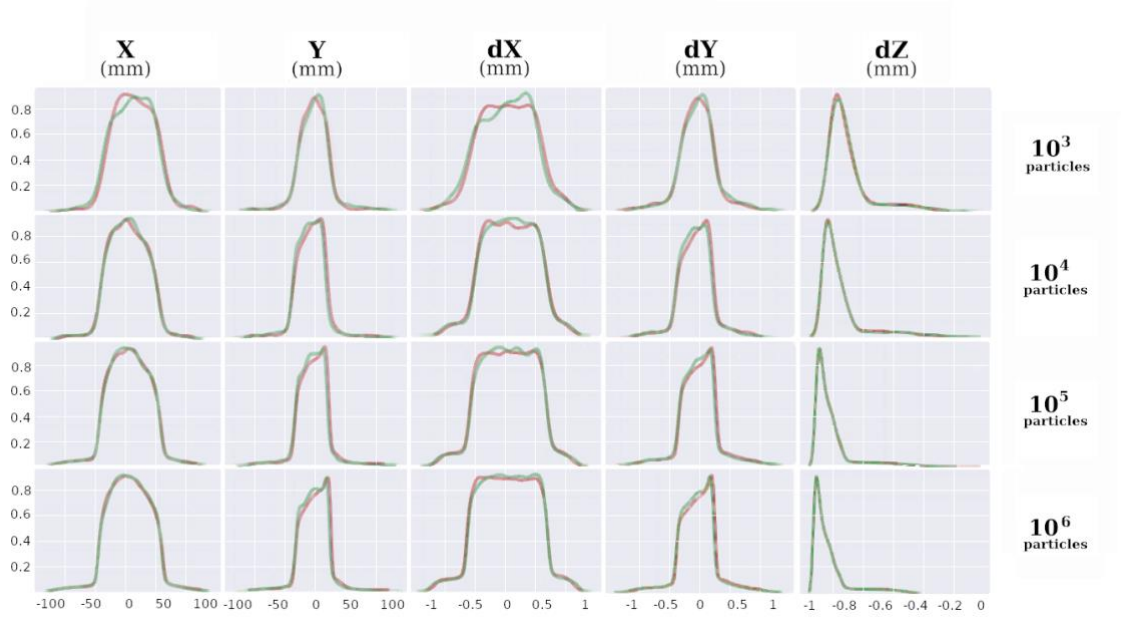


Figure 6.4: Marginal correlation plots for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using (10^3, 10^4, 10^5, 10^6) photons and the energy ranges between 40keV-50keV. Note the slight differences in terms of superimposition of real data and fake data for 10^3 photons for X and dX.

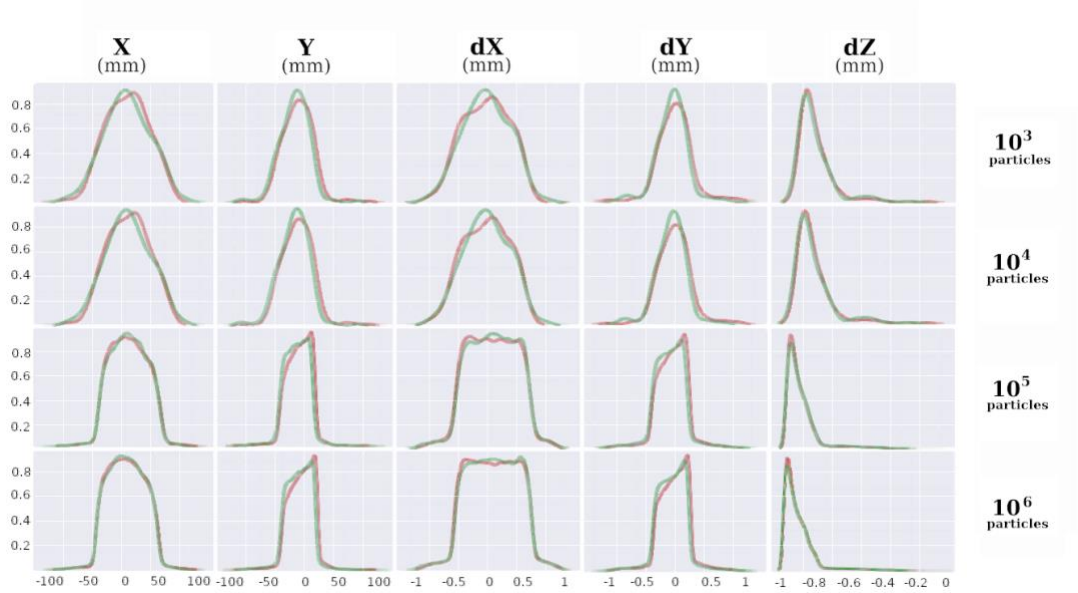


Figure 6.5: Marginal correlation plots for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using (10^3, 10^4, 10^5, 10^6) photons and the energy ranges between 50keV-60keV. Note the slight differences in terms of superimposition of real data and fake data for 10^3 and 10^4 photons.

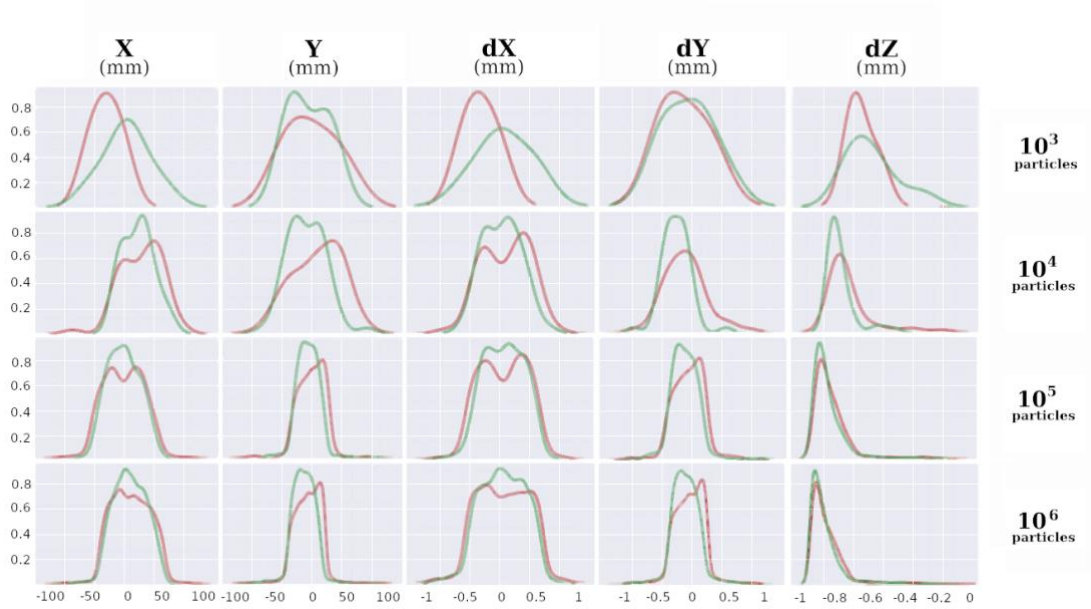


Figure 6.6: Marginal correlation plots for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using (10^3, 10^4, 10^5, 10^6) photons and the energy ranges between 60keV-70keV. Note the huge differences regarding superimposition of real data and fake data.

By increasing the number of data considering the whole given energy range, as it is shown in figure 6.7, better superimposition of fake data on real data has been observed. The best overlay belongs to 1 million samples of data.

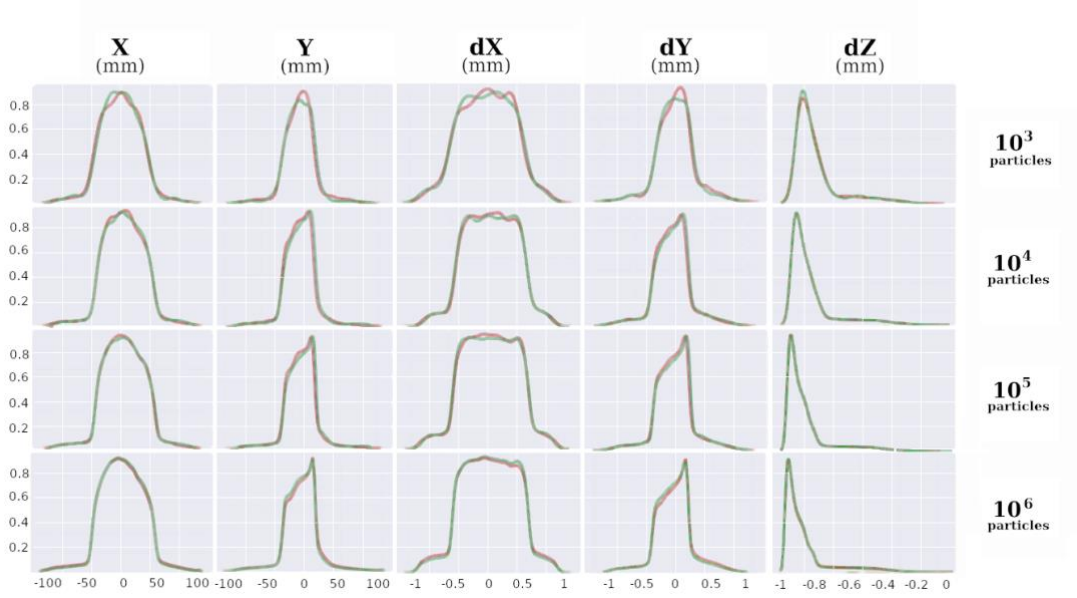
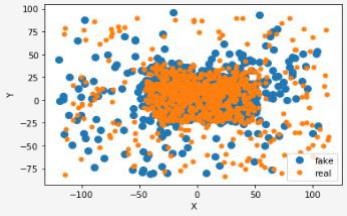
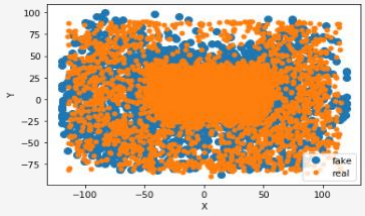
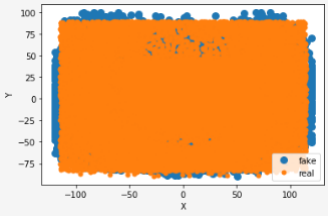
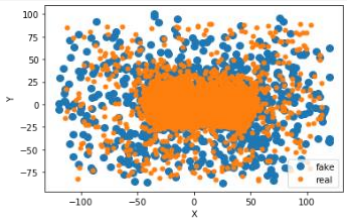
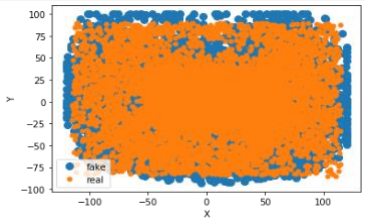
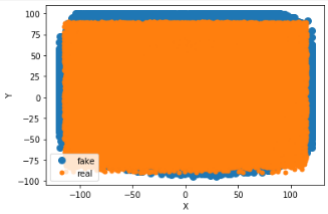
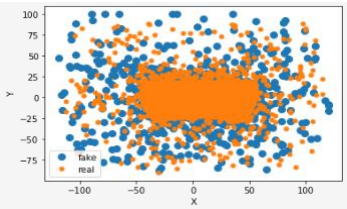
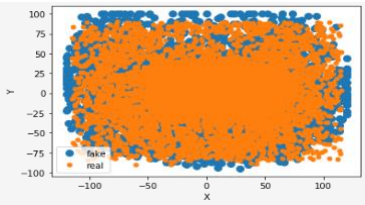
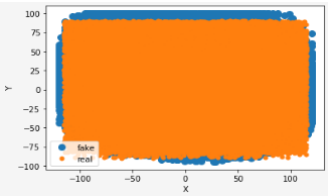
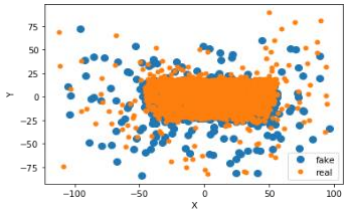
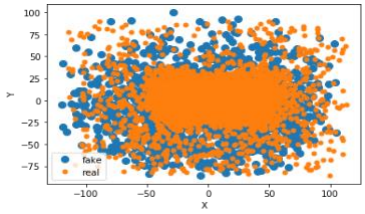
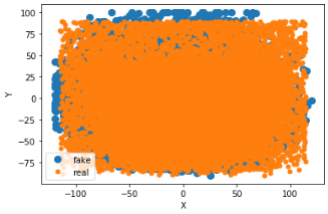
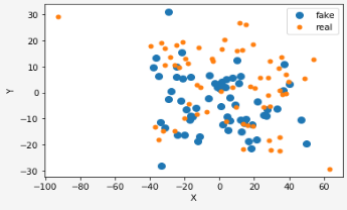
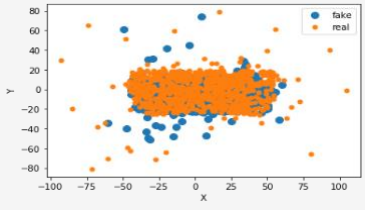
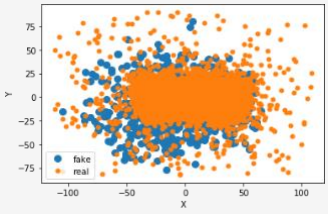


Figure 6.7: Marginal correlation plots for GAN generated fake data (red) and Monte Carlo simulated reference data (green) using (10^3, 10^4, 10^5, 10^6) photons and all the energy ranges between 20keV-70keV. Note the considerable overlap regarding superimposition of real and fake data.

As it is shown in tables 6.2, in the correlation X-Y plots, the superimposition of the real particles and fake ones are analyzed, based on different energy ranges, 3 different number of data (10^4,10^5,10^6) were selected. For energy range between 70keV-80keV, as it is shown in figure 6.8, no data was generated for X and Y parameters in spite of the fact, that real particles in this range do exist.

Table 6.2: X-Y plots for GAN generated fake data (blue) and Monte Carlo simulated reference data (orange) using (10^4, 10^5, 10^6) photons and the energy ranges between 20keV-70keV. These plots show how the particles distributed on the x-y coordinate. X-axis and y-axis show the intervals of the X and Y in millimeters, respectively. Note how the reference data is scattered for 10^6 photons and the energy range between 20keV to 60keV.

Energy range/ particles	**10^4**	**10^5**	**10^6**
20keV-30keV			
30keV-40keV			
40keV-50keV			
50keV-60keV			
60keV-70keV			

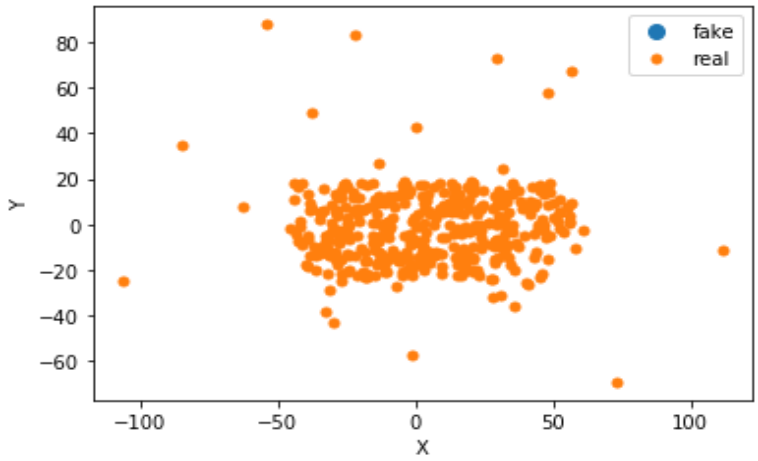


Figure 6.8: X-Y plots for GAN generated fake data (blue) and Monte Carlo simulated reference data (orange) for the energy range between 70keV-80keV. These plots show how the particles distributed on the x-y coordinate for the mentioned energy range. X-axis and y-axis show the intervals of the X and Y in millimeters, respectively. Note that no data was generated.

To obtain to which extent the generated particles are statistically significant, the tables 6.3-6.7 have been designed. The marked number of p-values in the following tables shows that for every data characteristic (number of data, parameters and energy range), considering the significance level, the null hypothesis is rejected in this area. The p-values in the mentioned tables have been obtained between the data generated by the GAN and the data from reference dataset (PHSP1).

Table 6.3: P-value between GAN generated fake data and Monte Carlo simulated reference data (PHSP1) for the energy range between 20keV-30keV using (10^3,10^4,10^5) photons. Note the p-values for 10^3 photons.

Photons/Parameters	**X**	**Y**	**dX**	**dY**	**dZ**
10^3	0.10	0.02	0.16	0.00	0.02
10^4	0.83	0.49	0.79	0.28	0.66
10^5	1	1	1	1	1

Table 6.4: P-value between GAN generated fake data and Monte Carlo simulated reference data (PHSP1) for the energy range between 30keV-40keV using (10^3,10^4,10^5) photons. Note the p-values for 10^5 photons

Photons/Parameters	X	Y	dX	dY	dZ
10^3	0.47	0.48	0.45	0.34	0.65
10^4	1	1	1	1	1
10^5	0.16	0	0.27	0	0

Table 6.5: P-value between GAN generated fake data and Monte Carlo simulated reference data (PHSP1) for the energy range between 40keV-50keV using (10^3,10^4,10^5) photons. Note the p-values for 10^5 photons.

Photons/Parameters	X	Y	dX	dY	dZ
10^3	0.57	0.54	0.43	0.59	0.27
10^4	1	1	1	1	1
10^5	0.15	0	0	0.14	0.00

Table 6.6: P-value between GAN generated fake data and Monte Carlo simulated reference data (PHSP1) for the energy range between 50keV-60keV using (10^3,10^4,10^5) photons.

Photons/Parameters	X	Y	dX	dY	dZ
10^3	0.77	0.20	0.96	0.14	0.30
10^4	0.77	0.20	0.96	0.14	0.30
10^5	0.77	0.20	0.96	0.14	0.30

Table 6.7: P-value between GAN generated fake data and Monte Carlo simulated reference data (PHSP1) for the energy range between 60keV-70keV using (10^3,10^4,10^5) photons. Note the p-values for 10^4 and 10^5 photons.

Photons/Parameters	X	Y	dX	dY	dZ
10^3	0.12	0.98	0.21	0.89	0.89
10^4	0.12	0.00	0.14	0.00	0.00
10^5	0.12	0.00	0.14	0.00	0.00

As shown in the figure 6.9 and 6.10, for five different characteristics of the particles, the x-axis shows the bins of energy and the y-axis denotes the p-value amount. Each colored point in the figures below demonstrates the p-value corresponding to each bin. The confidence level is marked by red horizontal line. In figure 6.9 for 10^3 particles, bin is set to 10 and the p-values between the real and fake data denotes, that the GAN was trained well due to the fact that respective p-values are above the significance level. In Figure 6.10 it is shown, that the p-values which are located beneath the confidence level reject our null-hypothesis for a higher number of particles.

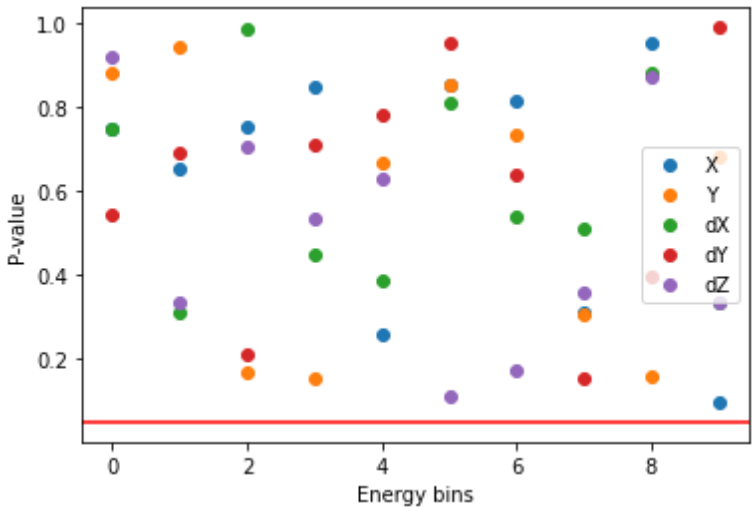


Figure 6.9: The visualized p-value plot between GAN generated data and Monte Carlo simulated reference data (PHSP1) using 10^3 photons and 10 bins of energy. Alpha and confidence level have been chosen 0.05 and 95%, respectively. Five different parameters for two datasets are indicated in the figure legend. Note that the whole p-values are above significance level.

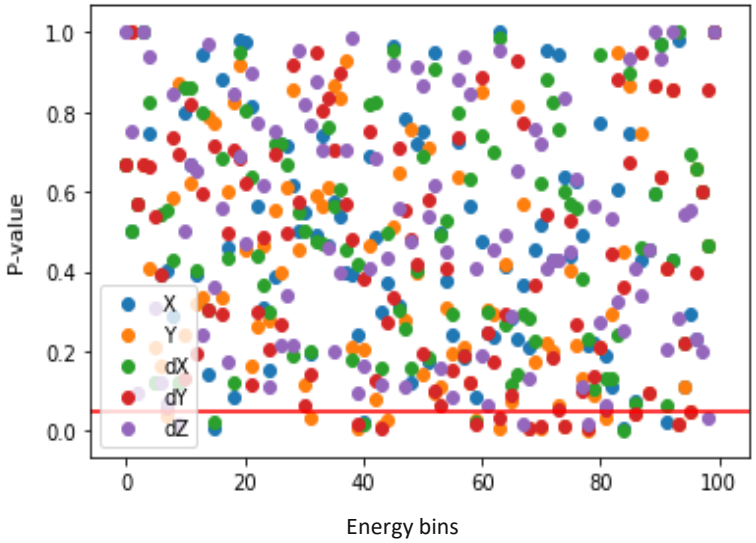


Figure 6.10: The visualized p-value plot between GAN generated data and Monte Carlo simulated reference data (PHSP1) using 10^6 photons and 100 bins of energy. Alpha and confidence level have been chosen 0.05 and 95%, respectively. Five different parameters for two datasets are indicated in the figure legend. Note the most of the p-values are above significance level and only a few of p-values rejects the null hypothesis.

The correlation matrices shown in figure 6.11 and figure 6.12 define the covariance normalized by the product of their standard deviations between the six parameters for the original phase space and the fake one. As an example, the value of 0.96 in figure 6.11 denotes that there is high correlation between two parameters (X,dX) in the phase space produced by MC simulations, which is also equal to the value in figure 6.12 produced by fake data. This is also valid for (Y-dY).

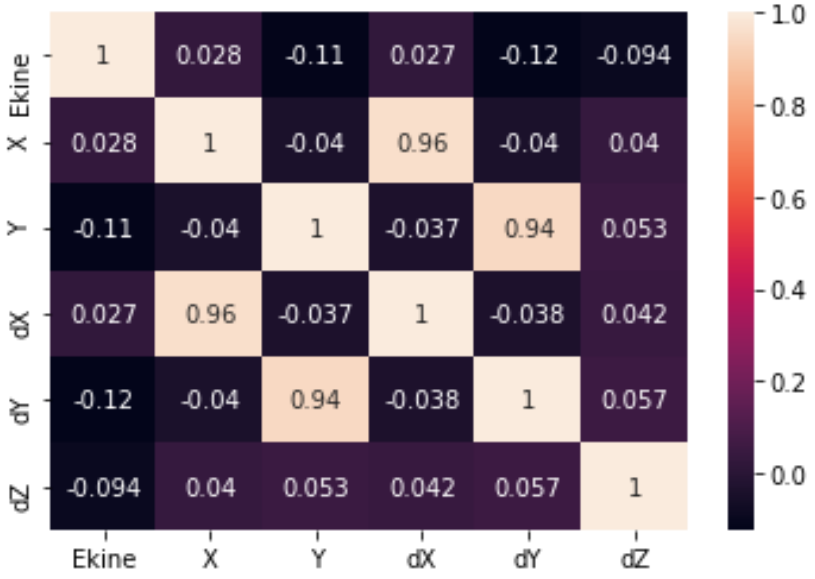


Figure 6.11: Correlation matrix for Monte Carlo simulated reference data (original data) using 10^6 photons. Note the correlation value between Y-dY and X-dX.

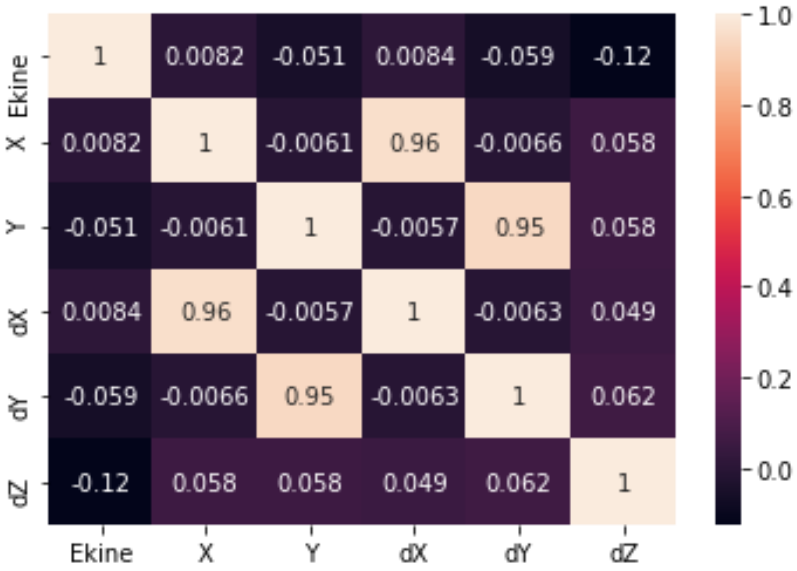


Figure 6.12: Correlation matrix for GAN generated data (fake) using 10^6 photons. Note the correlation value between Y-dY and X-dX.

The following plots obtained from the PCA codes for (10^3,10^4,10^5) photons by increasing transparency of the colored dots for all the energy ranges. As it is shown in figures 6.13-6.15, the majority of particles has been positioned at the center of PCA plot.

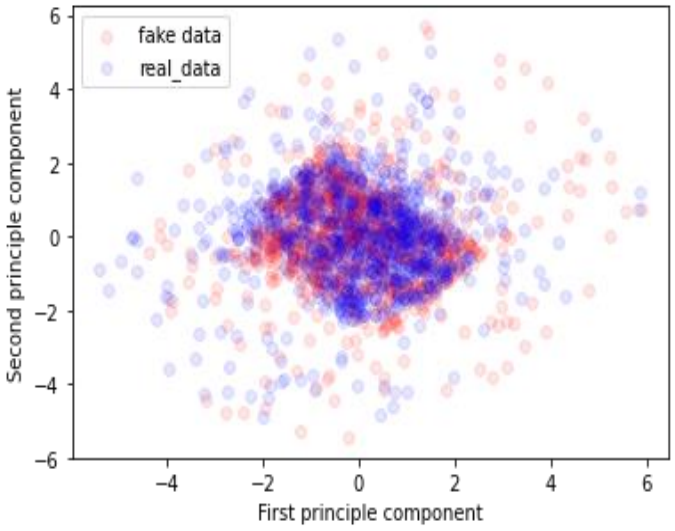


Figure 6.13: PCA plot for GAN generated fake data (red) and Monte Carlo simulated reference data (blue) using 10^3 photons. Note the superimposition of photons with an increase in transparency.

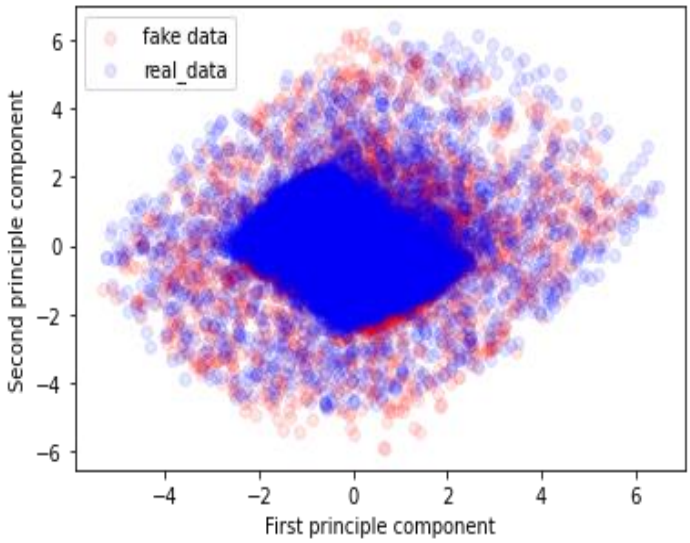


Figure 6.14: PCA plot for GAN generated fake data (red) and Monte Carlo simulated reference data (blue) using 10^4 photons. Note the superimposition of photons with an increase in transparency.

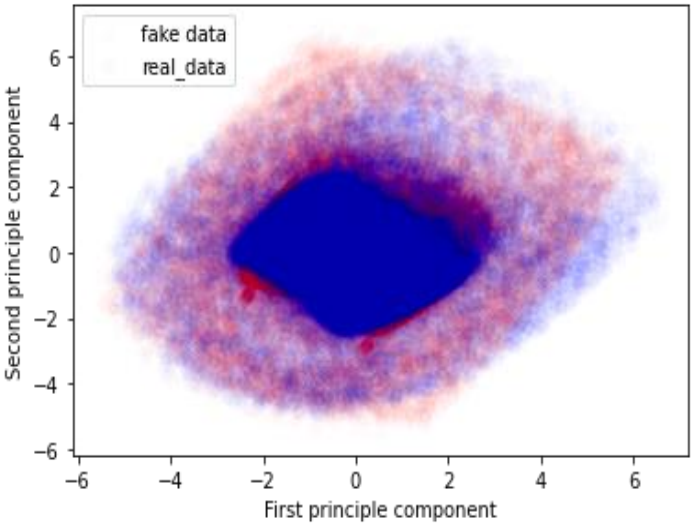


Figure 6.15: PCA plot for GAN generated fake data (red) and Monte Carlo simulated reference data (blue) using 10^5 photons. Note the superimposition of photons with an increase in transparency.

Table 6.8: P-value between Monte Carlo simulated reference data for training (PHSP1) and Monte Carlo simulated data for the evaluation (PHSP2) using (10^3,10^6) photons. Note that all the p-values are above significance level.

Data/Parameters	Ekine	X	Y	dX	dY	dZ
10^3	0.97	0.99	0.11	0.96	0.09	0.96
10^6	0.97	0.99	0.11	0.97	0.09	0.96

As it was mentioned in section 5.3, two phase space files, produced using a MC model, were provided. The file used for training is called PHSP1 and the other one used for evaluating is called PHSP2. The p-values in table 6.8 were gained in order to compare both phase space produced by MC model. The equal p-values for each parameter (10^3 and 10^6 photons) have been obtained. The maximum of P-value belongs to the X.

Table 6.9: The values of final D-Loss and G-Loss obtained at the end of training after 10^4 iterations.

Final D-Loss	**Final G-Loss**
-0.0002	0.0083

7 Discussion and Conclusion

In this chapter, at first, the results of the study are discussed and a conclusion is given. The suggestions for further research are presented to the readers. In the end, the limitation of the presented study is reported.

7.1 Discussion

The purpose of the present study was to implement a Generative Adversarial Network in order to produce a fake phase space and to evaluate the degree of similarity between the generated particles and the real ones. As mentioned before, training of GAN is a challenging process and due to the fact that GAN is always accompanied with the instability and mode collapse, the proposed method, called WGAN [66] was used in this research.

I verified that the phase space produced by GAN requires less than 10 MB of storage and the generation of particles from GAN is a fast process (within seconds), similar to the one produced in this related work [68]. The fake data was generated after around 10 hours, by using Google Colab. The process of training took 5 times longer than in the previous work by Sarrut [68]. Also, the learning rate was set to 2×10^{-5}, which is two times more than the value used in the related work [68] and the batch size was selected to 10^3, which is 10 times less [68]. The rest of the hyperparameters, like the number of neurons and the number of layers were already checked and altered. No significant improvement was achieved by that. However, with 10^3 batch size and 10^4 iterations, my final D-Loss was perfectly converged to zero (-0.002), which is significantly higher than the final D-Loss (-0.005) reported by Sarrut, with a batch size of 10^4 [68]. My work therefore shows, that the accurate particles generated by GAN were achieved by a smaller training dataset.

The particles generated by GAN were compared to the original phase space by MC model. For the assessment, two different techniques were employed to compare both the generated data from the Generator and the real data for a

specified amount of energy. To evaluate the generated data respectively, the numerical method and the visualized method were used. In this work, a new method to visualize the p-value was proposed in order to attain a better comprehension in the evaluation process. This method was implemented as a function in python as it was mentioned in 5.3.4.1.

The implementation was part of my work. In the following, all the evaluation techniques used in this thesis are discussed.

As shown in table 6.1, **PCA plots** for different energy ranges from 20kV to 60kV and for a particular number of 10^4 denote the well-matching of real and fake particles. In the mentioned table, the PCA plots for a larger number of particles were not matched well. Once the value of transparency was increased, the claim was rejected, since the majority of particles existed at the center and only a small fraction of them was scattered in the surrounding, as it shown in figure 6.14 and figure 6.15. This small fraction could not cause the non-matching of the particles.

The **marginal correlation** plots of the parameters in the energy range of 20 keV-60 keV in the figures 6.2-6.7 show that the fake data and real data were superimposed on each other, neglecting the slight differences in some areas. One limitation of such method is that it could not be suitable for larger energies (60 keV - 70 keV), meaning that such energy range lacks superimposition, as shown in figure 6.6. By comparing all the PCA obtained plots of the energy range (60 keV - 70 keV) in table 6.1 with figure 6.6, it denotes that the non-superimposition of plots in the mentioned energy range could be due to a lack of data for training. In addition, figure 6.7 denotes that the particles generated by the GAN were completely matched with the particles generated by MC simulations if no energy range considered.

In **X-Y correlation technique**, for the highest number particles, as well as the energy range between 20keV-60keV, the particles were superimposed thoroughly. However, below this range, real particles and fake ones were not overlaid perfectly. As shown in figure 6.8, no particle was generated, in the energy range of 70keV-80keV, due to fact that real data in this range was also limited. I realized that this technique was not powerful like PCA, since only two parameters were used. Nevertheless, it shows how particles were scattered on the

x-y coordinate. If a low number of particles were chosen, the majority of them would distribute at the center.
As a result of the **correlation matrix**, as it shown in figure 6.11 and figure 6.12, the high order of the correlations for X-dX and Y-dY were also modeled by the GAN. The correlation between X-dX and Y-dY was the same for generated data and original ones. The equivalent correlation shows satisfactory results, which denotes that the phase space was trained well. The reason for this issue is that the X-dX and Y-dY correlations are initiated from the cone geometry of the X-ray [68] which were obtained equivalent for the original and fake data.

The major difference as shown among figure 6.9 and figure 6.10 is, that by increasing the number of samples (particles) for 100 bins, more particles were needed to be compared with each other in each bin in the **visualized p-value technique.** This causes the p-value to easily go down to zero. If 10^6 particles are used for 100 bins, the calculated p-value may face a limitation. Once the number of the particles in each bin was more than 10^4, then the approximate values were gained by the ks_2samp in Kolmogorov-Smirnov in python [75]. This shows that the test could not provide exact values when the number of comparative particles was more 10^4. However, in case of having fewer than 10^4 particles in each bin, such limitation no longer occurs.

The **p-values** obtained in the tables 6.3-6.7 showed contradictory results in comparison with the slight differences existed in the marginal correlation plots. For better comparison, the values in table 7.1 were calculated from tables 6.3-6.7. This technique was proposed as a comparative method, to check the total p-values of each mentioned table against each other. As an example, 80% means that in the energy range between 20keV-30KeV, 80% of the p-values in table 6.3 were above significance level. The lowest significance score (60%), obtained from table 6.7, was apparently caused by the lack of data, seen in PCA and X-Y correlation methods for the given energy range (60keV-70keV).

Table 7.1: The significance score obtained from tables 6.10-6.14 for different energy ranges.

Energy range	**20-30 keV**	**30-40 keV**	**40-50 keV**	**50-60 keV**	**60-70 keV**
Significance score	80%	80%	80%	100 %	60%

The p-value in the energy range between 50keV-60keV was completely statistically significant, that is to say, the p-values in all areas were above significance level (100%). As a result of the **marginal correlation plots**, the real and fake data were overlaid on each other in the range of 20keV-60keV and the value existed in the given range in table 7.1 concluded, that the amount of significance score (80%-100%) for the mentioned energy range was precise. Besides, the p-value in the energy range of 60keV-70keV had the lowest significance score as shown in the mentioned table. The lack of superimposition existed in figure 6.6, which is initiated from the lack of reference particles for the given energy range, results in the lowest significance score.

The superimposition of real and fake data on each other, for the energy range of 20keV-60keV, is seen for the position of the parameters and the direction of the particles. The significance score in the given range is 80%-100%, means that only about 20% of p-values reject the null hypothesis. However, for the energy range of 60keV-70keV, the significance score results 60% which indicates 40% of p-values reject the null hypothesis, which is considerable high.

Most of the particles in the reference dataset (PHSP1) were distributed in the energy range of 20keV-60keV, as it shown in figure 5.3. In the evaluation techniques, the energy range of 60keV-70keV was also considered to assess the importance of insufficient data for training.

After reviewing all the evaluation techniques, I conclude that, p-value and PCA techniques were not suitable methods to analyze millions of photons generated by GAN. Although the mentioned techniques were very effective for the low number of photons.

Overall, the results of all used evaluation methods for generated and original phase space denote that the generated particles were approximative but not exactly matching. However, two phase spaces produced by MC had the equivalent p-values for a different number of particles (10^3, 10^6). As a conclusion of used evaluation techniques, the used methods in this thesis like PCA, correlation plot and correlation matrix show that the best superimposition of real and fake data are seen in the energy range of 20keV-60keV.

A considerable advantage of using GAN to generate the characteristic of particles is that an arbitrary number of particles can be produced within seconds by modeling a large phase space file [68]. In addition, the phase space file produced by GAN which requires less storage, was produced based on a smaller training dataset.

7.2 Conclusion

In this thesis, it was researched whether it is possible to use ANN to generate the phase space which contains the characteristic of the particles. In order to do so, a method called Generative Adversarial Network proposed by Sarrut was used. This approach concludes that the generated phase space requires less than 10 MB storage which is considerably less in comparison to the 50 GB reference dataset. In addition, the generation of the 10^6 particles takes less than few seconds. Besides, all the evaluation techniques, marginal distribution, PCA, P-value technique, etc., denote that our generated data has the feasibility to be replaced by original one. The results demonstrates that the purpose of this thesis has come true, if there is sufficient data provided for training.

7.3 Suggestions for Further Research

GAN has been proposed, for the first time, to generate data with a higher number of dimensions and less smooth distributions like images [68]. The use of a special type of GAN, Wasserstein GAN with gradient penalty, to generate the characteristic of particles, was a novel method proposed by Sarrut. Nevertheless, GAN has different types that are considering the state-of-the-art techniques in the artificial intelligence. For this purpose, working on different neural network methods like cycle GAN or conditional GAN to generate the phase space characteristics would be a good suggestion.

7.4 Limitations of the Study

The limitation of this work is initiated from a lack of reference particles in the energy range between 60keV-70keV as it is obviously shown in the table 6.1 and

in table 6.2 as well. If the original phase space contains all the given energy ranges, the generated one would be 80% or more statistically significant.

Bibliography

1. Aderibigbe, A. (2014). A Term Paper on Monte Carlo Analysis/Simulation. University of Ibadan. www.academia.edu/8748422/Monte_Carlo_Simulation.
2. Harrison, R. L. (2010). Introduction To Monte Carlo Simulation. AIP Conference Proceedings, 1204, 17–21. https://doi.org/10.1063/1.3295638
3. Haghighat, A. (2020). Monte Carlo methods for particle transport. Crc Press.
4. Behling, R. (2015). Modern Diagnostic X-Ray Sources: Technology, Manufacturing, Reliability. CRC Press.
5. Bre, Facundo & Gimenez, Juan & Fachinotti, Víctor. (2017). Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. Energy and Buildings. 158. 10.1016/j.enbuild.2017.11.045.
6. Hoskin, P. (2019). External Beam Therapy (3rd ed.). Oxford University Press. https://doi.org/10.1093/med/9780198786757.001.0001
7. National Cancer Institute (NCI). (2019, December 27). Getting External Beam Radiation Therapy. Cancer.org. https://www.cancer.org/content/dam/CRC/PDF/Public/9245.00.pdf
8. Baskar, R., Lee, K. A., Yeo, R., & Yeoh, K. W. (2012). Cancer and radiation therapy: current advances and future directions. International journal of medical sciences, 9(3), 193–199. https://doi.org/10.7150/ijms.3635
9. Jackson, S. P., & Bartek, J. (2009). The DNA-damage response in human biology and disease. *Nature*, *461*(7267), 1071-1078.
10. Ferrari, A. (2011, November). Monte Carlo Sampling. 18th Fluka Course, Shanghai, China. https://indico.cern.ch/event/540415/contributions/2194806/attachments/1285749/1912258/09_Statistics_and_sampling_2015.pdf
11. Zechner, A., Stock, M., Kellner, D., Ziegler, I., Keuschnigg, P., Huber, P., Mayer, U., Sedlmayer, F., Deutschmann, H., & Steininger, P. (2016). Development and first use of a novel cylindrical ball bearing phantom for 9-DOF geometric calibrations of flat panel imaging devices used in image-guided ion beam therapy. Physics in medicine and biology, 61(22), N592–N605. https://doi.org/10.1088/0031-9155/61/22/N592
12. Keuschnigg, P., Kellner, D., Fritscher, K., Zechner, A., Mayer, U., Huber, P., Sedlmayer, F., Deutschmann, H., & Steininger, P. (2017). Nine-degrees-of-freedom flexmap for a cone-beam computed tomography imaging device with independently movable source and detector. Medical physics, 44(1), 132–142. https://doi.org/10.1002/mp.12033
13. EBG MedAustron GmbH. (n.d.). Medical Technology. medaustron.at. Retrieved October 3, 2021, from https://www.medaustron.at/en/medical-technology
14. medPhoton GmbH. (2015) ImagingRing Geometry Specifications Interface Control Document. Salzburg, Austria.

15. van der Heyden, B. (2020). Advanced Computed Tomography imaging in radiotherapy. Maastricht University.
16. MedPhoton GmbH. (n.d.) ImagingRing Benutzerhandbuch IR (Version 1.1). Salzburg, Austria
17. Rumelhart, David E and Hinton, Geoffrey E and Williams, Ronald J. Learning Representations by Back-Propagating Errors. nature, 323(6088):533, 1986.
18. Geant4 Collaboration. (2016). Geant4 User's Guide for Application Developers (Version 10.3 ed.). https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/BackupVersions/V10.3/fo/BookForAppliDev.pdf
19. Wilson, R. R. (1946). Radiological use of fast protons. Radiology, 47(5), 487-491.
20. Bragg, W. H., and Kleeman, R. (1905). XXXIX. On the α particles of radium, and their loss of range in passing through various atoms and molecules. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 10(57), 318-340.
21. K. P. Nesteruk, C. Calzolaio, D. Meer, V. Rizzoglio, M. Seidel, and J. M. Schippers (2018) Large energy acceptance gantry for proton therapy utilizing superconducting technology
22. Delaney TF, Kooy HM editors. (2008). Proton and charged particle radiotherapy [M] Philadelphia: Lippincott Williams and Wilkins;
23. Liu, H. and Chang, J.Y. (2011). Proton therapy in clinical practice. Chinese journal of cancer, 30(5), p.315.
24. M H Seegenschmiedt, O Micke, R Muecke (2015) Radiotherapy for non-malignant disorders: state of the art and update of the evidence-based practice guidelines
25. Mücke, R., Seegenschmiedt, M. H., Heyd, R., Schäfer, U., Prott, F. J., Glatzel, M., Micke, O., & German Cooperative Group on Radiotherapy for Benign Diseases (GCG-BD) (2010). Strahlentherapie bei schmerzhafter Kniegelenkarthrose (Gonarthrose): Ergebnisse einer deutschen Patterns-of-Care-Studie [Radiotherapy in painful gonarthrosis. Results of a national patterns-of-care study]. Strahlentherapie und Onkologie : Organ der Deutschen Rontgengesellschaft ... [et al], 186(1), 7–17. https://doi.org/10.1007/s00066-009-1995-7
26. Kamada T. (2015). Twenty years of carbon ion radiation therapy at the national institute of radiological sciences: accomplishments and prospects. Int J Particle Ther. 2:459–63. doi: 10.14338/IJPT-15-00030.1
27. Weaver, K., Rowland, J., Bellizzi, K., & Aziz, N. (2010). Forgoing medical care because of cost. Cancer, 116(14), 3493-3504.
28. Rackwitz, T., & Debus, J. (2019). Clinical applications of proton and carbon ion therapy. Seminars in oncology, 46(3), 226–232. https://doi.org/10.1053/j.seminoncol.2019.07.005
29. S. Jan et al.(2004). Gate: a simulation toolkit for PET and SPECT. Physics in Medicine and Biology, 49(19):4543.
30. S. A. et al. (2003) Geant4 - a simulation toolkit. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 506(3):250–303. doi: 10.1016/s0168-9002(03)01368-8.

31. Landberg T, Chavaudra J, Dobbs J, et al. (1993). ICRU Report 50: Prescribing, Recording, and Reporting Photon Beam Therapy. International Commission on Radiation Units and Measurements.
32. Landberg T, Chavaudra J, Dobbs J, et al. (1999). ICRU Report 62. Prescribing, Recording, and Reporting Photon Beam Therapy (Supplement to ICRU Report 50). International Commission on Radiation Units and Measurements.
33. ICRU. (2007). Estimation and Presentation of Uncertainty in the Delivered Dose. Journal of the ICRU, 7(2), 131–134. https://doi.org/10.1093/jicru_ndm042
34. N. Hodapp. (2012) The ICRU Report No. 83: Prescribing, recording and reporting photon-beam intensity-modulated radiation therapy (IMRT), International Commission on Radiation Units and Measurements.
35. Andreo, P. (2018) The physics of small megavoltage photon beam dosimetry. Radiotherapy and Oncology, 126(2):205–213.
36. Knipe, H., Jones, J. (n.d). 3D conformal radiation therapy. Reference article, Radiopaedia.org. (accessed on 01 Oct 2021) https://radiopaedia.org/articles/67493
37. Cancer Research UK. (2020, November 6). Intensity Modulated Radiotherapy (IMRT). Cancerresearchuk. https://www.cancerresearchuk.org/about-cancer/cancer-in-general/treatment/radiotherapy/external/types/intensity-modulated-radiotherapy-imrt
38. Mayo Foundation for Medical Education and Research. (2012). Image-guided radiation therapy (IGRT). mayoclinic.org. https://www.mayoclinic.org/tests-procedures/image-guided-radiation-therapy/about/pac-20385267
39. Di Yan. (2006). Image-Guided/Adaptive Radiotherapy, Springer Berlin Heidelberg, 321–336
40. PTCOG. (2021) Particle Therapy Facilities in Clinical Operation. Ptcog.ch, Particle Therapy Co-Operative Group (PTCOG), www.ptcog.ch/index.php/facilities-in-operation.
41. Reiz, N. (2018) Head scatter modeling of the ImagingRing System
42. INTERNATIONAL ATOMIC ENERGY AGENCY. (2014) Diagnostic Radiology Physics. IAEA. Vienna
43. WHO. (9 Dec. 2020). The Top 10 Causes of Death. who.int. WHO. www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death.
44. SEER Training Modules, Module Name. U. S. National Institutes of Health, National Cancer Institute, https://training.seer.cancer.gov
45. Demtröder, W. (2015). Experimentalphysik 3 (5th Edition). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49094-5
46. Kaiser, Adeel & Eley, John & Onyeuku, Nasarachi & Rice, Stephanie & Wright, Carleen & McGovern, Nathan & Sank, Megan & Zhu, Mingyao & Vujaskovic, Zeljko & Simone, Charles & Hussain, Arif. (2019). Proton Therapy Delivery and Its Clinical Application in Select Solid Tumor Malignancies. Journal of Visualized Experiments. 10.3791/58372
47. Alpaydin, Ethem. (2014) Introduction to Machine Learning. MIT press
48. Lundervold, A.S. and Lundervold, A. (2019). An overview of deep learning in medical imaging focusing on MRI. Zeitschrift für Medizinische Physik, 29(2), pp.102-127.

49. Goodfellow, Ian and Bengio, Yoshua and Courville, Aaron. (2016) Deep learning, volume 1. MIT press Cambridge,
50. Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org.
51. Maier, A., Syben, C., Lasser, T. and Riess, C. (2019). A gentle introduction to deep learning in medical image processing. Zeitschrift für Medizinische Physik, 29(2), pp.86-101
52. Nair, Vinod and Hinton, Geoffrey E. (2010). Rectified Linear Units Improve Restricted Boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML- 10), pages 807–814
53. Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, Advances in Neural Information Processing Systems 25 (NIPS), pages 1097–1105. Curran Associates, Inc
54. Li, F.-F., Johnson, J., and Yeung, S. (2017). CS231n: Convolutional neural networks for visual recognition.
55. Maas, Andrew L and Hannun, Awni Y and Ng, Andrew Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. In Proc. icml, volume 30, page 3
56. Xu, Bing and Wang, Naiyan and Chen, Tianqi and Li, Mu. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv preprint arXiv:1505.00853.
57. Hinton, Geoffrey. (n.d.).Neural Networks for Machine Learning. YouTube, uploaded by Blitz Kim, 8. Dezember 2016, www.youtube.com/watch?v=SJ48OZ_qlrc.
58. Ruder, S. (2016). An overview of gradient descent optimization algorithms. CoRR, abs/1609.04747
59. Tieleman, Tijmen and Hinton, Geoffrey. (2012). Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude. COURSERA: Neural networks for machine learning, 4(2):26–31
60. Hinton, G., Srivastava, N., and Swersky, K. (2012). Lecture 6a: Overview of mini- batch gradient descent. COURSERA: Neural Networks for Machine Learning
61. Riedmiller, Martin and Rprop, I. (1994) Rprop-Description and Implementation Details
62. Kingma, Diederik P and Ba, Jimmy Adam. (2014). A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980
63. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, & Yoshua Bengio. (2014). Generative Adversarial Networks.
64. Goodfellow, Ian. (2017) NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv preprint arXiv:1701.00160,
65. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved Techniques for Training GANs. Pages 2234–2242,
66. Arjovsky, Martin and Chintala, Soumith and Bottou, L´eon. (2017) Wasserstein GAN. arXiv preprint arXiv:1701.07875
67. Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein Generative Adversarial Networks. In Precup, D. and Teh, Y. W., editors, Proceedings of the 34th International

Conference on Machine Learning (ICML), volume 70 of Proceedings of Machine Learning Research, pages 214–223. PMLR

68. Sarrut, D., Krah, N. and Létang, J.M. (2019). Generative adversarial networks (GAN) for compact beam source modelling in Monte Carlo simulations. Physics in Medicine & Biology, 64(21), p.215004.
69. Villani, C. (2008). Optimal Transport. Springer.
70. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of Wasserstein GANs. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in Neural Information Processing Systems 30 (NIPS), pages 5769–5779. Curran Associates, Inc
71. Gulrajani, Ishaan and Ahmed, Faruk and Arjovsky, Martin and Dumoulin, Vincent and Courville, Aaron C. (2017) Improved Training of Wasserstein GANs. In Advances in Neural Information Processing Systems, pages 5769–5779.
72. Jolliffe, I.T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), p.20150202
73. Sarrut, David. (2021)GAGA = GAN for GATE. python. commit 84163e8. GitHub. https://github.com/dsarrut/gaga
74. Li, Fei-Fei, Krishna, R. & Xu, D. (2021, 8. April). Lecture 4: Neural Networks and Backpropagation [lecture slides]. Stanford University.
75. The SciPy community. (2021, August 2). scipy.stats.ks_2samp. SciPy Documentation. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ks_2samp.html
76. Pearson, K. (1900). X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *50*(302), 157-175.
77. Lin, M., Lucas Jr, H. C., & Shmueli, G. (2013). Research commentary—too big to fail: large samples and the p-value problem. *Information Systems Research*, *24*(4), 906-917.
78. Piacentino, E. (2019) Generative Adversarial Network based Machine for Fake Data Generation.
79. Brownlee, J. (2019, June 12). Best Resources for Getting Started With GANs. Machinelearningmastery. Retrieved October 15, 2021, from https://machinelearningmastery.com/resources-for-getting-started-with-generative-adversarial-networks/
80. Fedus W., Rosca M., Lakshminarayanan B., Dai A.M., Mohamed S. and Goodfellow I. (2018). “Many Paths to Equilibrium: GANs Do Not Need to Decrease aDivergence At Every Step”. arXiv preprint arXiv:1710.08446.
81. Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training Generative Adversarial Networks. In Proceedings of the Fifth International Conference on Learning Representations (ICLR).
82. Neff, T. (2018) Data Augmentation in Deep Learning using Generative Adversarial Networks.
83. Paszke A., Gross S., Chintala S., Chanan G., Yang E., DeVito Z., Lin Z., Desmaison A., Antiga L. and Lerer A. (2017). “Automatic differentiation in PyTorch”.

Appendix

The p-value for each parameter was obtained for different energy intervals. Although the p-value for the whole distribution was unexpectedly small, the null hypothesis was rejected and two distributions were not the same. The reason is that the p-value in the wo-sample Kolmogorov-Smirnov test could not be an accurate factor for comparing the large amount of data.
In order to compare two proposed methods (numerical p-value method and the correlation plot method), each corresponding table and figure for these two techniques were considered. It is clear that in some related areas, the p-value was below the significance level, meaning that the result was not statistically significant but the corresponding plot was perfectly overlaid. For this issue, the table 7.1 was designed in order to denote that to what extent the p-value was precise in this case. As the result of the correlation of plots, the real and fake data were overlaid on each other in the range of 20keV-60keV and the p-value percentage in the given range concluded that the amount of significance (80-100%) for the mentioned energy range was precise . Surprisingly, the p-value in the energy range between 50keV-60 keV was completely statistically significant, that is to say, the p-values in all areas were above significance level (100%). Besides, the p-value in the energy range of 60keV-70keV was not statistically significant, as table 6.7 shown the lack of superimposition for such range of energy.
The small amount of the p-values below significance level rejects the null hypothesis; however, it can be concluded that the both distributions are not the same but the reason for this issue could be due to the size of the comparative data. The superimposition of real and fake data on each other, for the energy range of 20keV-60 keV, was seen for the position of the parameters and the direction of the particles. The significance score in the given range 80%-100% means that about 20% of p-values rejects the null hypothesis and concludes two distributions (fake and real data) are not same. Without considering this percentage and by observing the visualized methods (PCA, correlation plot and X-Y plot), the best superimposition of real and fake data was seen in the energy range of 20keV-60keV. For the energy range of 60keV-70 keV, no superimposition was observed and the numerical p-value method defined the 40% of the p-values rejects the null hypothesis.

Code description

Below, the different sections of the GAN code model will be explained, which is the base for the rest of GAN systems shown in this project. It is important to highlight that the base code and structure of this script has been extracted from GitHub [73].

Gaga_train:

In the first few lines, the necessary packages like json, click, numpy and etc, have been imported. Then the three pre-specified directories for the input file (training data), json file (file which contains the all GAN's parameters) and the output file with pth format have been defined. In the following lines of the code, the parameters existed in the json file and in the input file were read. The training part of GAN has been done after normalization of considered parameters or keys in input file. In the last step, the trained data have been saved in the mentioned directory as the output file.

```
import click
import json
import time
import socket
import gatetools.phsp as phsp
import gaga
import copy
import numpy as np
from colorama import init
from colorama import Fore, Style
import torch
def gaga_train(phsp_filename, json_filename, output_filename,
epoch,progress_bar, plot,  plot_every_epoch, w_e, w_n, w_l, w_p,  keys,
validation_dataset, validation_every_epoch, start_pth):
    '''
    \b
    Train GAN to learn a PHSP (Phase Space File)

    \b
    <PHSP_FILENAME>    : input PHSP file (.npy)
    <JSON_FILENAME>    : input json file with all GAN parameters
    <OUTPUT_FILENAME> : output GAN as pth file
    '''


    # term color
    init()
    pkeys = keys
```

```
    # read parameters
    param_file = open(json_filename).read()
    params = json.loads(param_file)
    params['progress_bar'] = progress_bar
    params['plot'] = plot
    params['plot_every_epoch'] = plot_every_epoch
    params['training_filename'] = phsp_filename
    params['validation_filename'] = validation_dataset
    params['validation_every_epoch'] = validation_every_epoch
    start = datetime.datetime.now()
    params['start date'] = start.strftime(gaga.date_format)
    params['hostname'] = socket.gethostname()
    params['dump_wasserstein_every'] = int(w_e)
    params['w_n'] = int(w_n)
    params['w_l'] = int(w_l)
    params['w_p'] = int(w_p)
    params['start_pth'] = start_pth

    # the epoch parma in the json file may be overwritten by the option
    if epoch:
        params['epoch'] = epoch

    # read input training dataset
    print(Fore.CYAN +"Loading training dataset ...
"+phsp_filename+Style.RESET_ALL)
    x, read_keys, m = phsp.load(phsp_filename)

    # consider only some keys
    if 'keys' in params:
        keys = params['keys']
        if pkeys != '':
            keys = pkeys
        keys = phsp.str_keys_to_array_keys(keys)
        if 'angleXY' in keys:
            x, read_keys = phsp.add_angle(x, read_keys, 'X', 'Y')
        x = phsp.select_keys(x, read_keys, keys)
    else:
        keys = read_keys

    params['training_size'] = len(x)
    params['keys'] = keys
    params['x_dim'] = len(keys)

    # normalisation
    x_mean = np.mean(x, 0, keepdims=True)
    x_std = np.std(x, 0, keepdims=True)
    params['x_mean'] = x_mean
    params['x_std'] = x_std
    x = (x-x_mean)/x_std

    # print parameters
    for e in params:
        if (e[0] != '#'):
            print('   {:22s} {}'.format(e, str(params[e])))

    # train
    print(Fore.CYAN +'Building the GAN model ...'+Style.RESET_ALL)
    gan = gaga.Gan(params,x)
    print(Fore.CYAN +'Start training ...'+Style.RESET_ALL)
    optim = gan.train()
```

```
    # save
    stop = datetime.datetime.now()
    params['end date'] = stop.strftime(gaga.date_format)
    output = dict()
    output['params'] = params
    output['optim'] = optim
    state = copy.deepcopy(gan.G.state_dict())
    output['g_model_state'] = state
    state = copy.deepcopy(gan.D.state_dict())
    output['d_model_state'] = state

    torch.save(output, output_filename)
```

Gaga_generate

As it was mentioned earlier, Gaga_generate provides the arbitrary number of generated particles. In addition, the generated particles requires less than few megabytes storage. Its function needs two inputs, first the directory of pth file coming from Gaga_train must be defined and then the number of arbitrary particles has to be specified.

```
import click
import gaga
import gatetools.phsp as phsp
import torch
import os
import numpy as np
from torch.autograd import Variable

def gaga_plot(pth_filename, n, output, toggle, radius):
    '''
    \b
    Generate a PHSP from a GAN

    \b
    <PTH_FILENAME>    : input GAN PTH file (.pth)
    '''

    n = int(n)

    # load pth
    params, G, optim, dtypef= gaga.load(pth_filename)
    f_keys = list(params['keys'])

    # generate samples
    b = 1e5
    fake = gaga.generate_samples2(params, G, n, b, False, True)
```

```
    # Keep X,Y or convert to toggle
    if toggle:
        keys = phsp.keys_toggle_angle(f_keys)
        fake, f_keys = phsp.add_missing_angle(fake, f_keys, keys, radius)
        fake = phsp.select_keys(fake, f_keys, keys)
    else:
        keys = f_keys

    # special case (for retro-compatibility)
    try:
        i = keys.index('E')
        fake[:,i][fake[:,i] <0] = 0.00000000001 # E should not be zero !
    except:
        i = keys.index('Ekine')
        fake[:,i][fake[:,i] <0] = 0.00000000001 # E should not be zero !

    # write
    if output == 'AUTO':
        b, extension = os.path.splitext(pth_filename)
        output = b+'.npy'
phsp.save_npy(output, fake, keys)
```

Gaga_plot

The gaga_plot code provides the marginal distribution plots for six parameters (Ekine, X, Y, dX, dY, dZ) obtained by the GAN and the reference dataset. Each plot also shows the mean and standard deviation in order to compare the generated data and real data. It requires two input file: the training data in NumPy format and also the path file in pth format generated by gaga_train code.

```
import click
import gaga
import gatetools.phsp as phsp
import torch
import numpy as np
from torch.autograd import Variable


def gaga_plot(phsp_filename, pth_filename, n, nb_bins,
              toggle, radius, quantile, plot2d):

    # nb of values
    n = int(n)

    keys_2d = plot2d;
    if keys_2d == None:
        keys_2d = []
```

```
# load phsp
real, r_keys, m = phsp.load(phsp_filename, n)

# load pth
params, G, D, optim, dtypef= gaga.load(pth_filename)
f_keys = params['keys']
keys = f_keys.copy()

# generate samples
fake = gaga.generate_samples2(params, G, n, int(1e5), False, True)

# Keep X,Y or convert to toggle
if toggle:
    keys = phsp.keys_toggle_angle(keys)

real, r_keys = phsp.add_missing_angle(real, r_keys, keys, radius)
fake, f_keys = phsp.add_missing_angle(fake, f_keys, keys, radius)

real = phsp.select_keys(real, r_keys, keys)
fake = phsp.select_keys(fake, f_keys, keys)

# curate keys_2d
k2 = []
for k in keys_2d:
    if (k[1] in keys) and (k[0] in keys):
        k2.append(k)
keys_2d = k2

# fig panel
nb_fig = len(keys)+len(keys_2d)
nrow, ncol = phsp.fig_get_nb_row_col(nb_fig)
fig, ax = plt.subplots(nrow, ncol, figsize=(25,10))

# plot all keys for real data
i = 0

q = {}
for k in keys:
    index = keys.index(k)
    d = real[:,index]
    q1 = quantile
    q2 = 1.0-quantile
    q[k] = (np.quantile(d, q1), np.quantile(d, q2))
    gaga.fig_plot_marginal(real, k, keys, ax, i, nb_bins, 'g', q[k])
    i = i+1

# plot all keys for fake data
i = 0
for k in keys:
    index = keys.index(k)
    d = real[:,index]
    #q1 = quantile
    #q2 = 1.0-quantile
    #q = (np.quantile(d, q1), np.quantile(d, q2))
    #print(q)
    gaga.fig_plot_marginal(fake, k, keys, ax, i, nb_bins, 'r', q[k])
    i = i+1


# plot 2D distribution
```

```
    if len(keys) > 1:
        starti = i
        for kk in keys_2d:
            gaga.fig_plot_marginal_2d(real, kk[0], kk[1], keys, ax, i,
nb_bins, 'g')
            i = i+1

        # plot 2D distribution
        i = starti
        for kk in keys_2d:
            gaga.fig_plot_marginal_2d(fake, kk[0], kk[1], keys, ax, i,
nb_bins, 'r')
            i = i+1

        if False:
            for kk in keys_2d:
                a = phsp.fig_get_sub_fig(ax,i)
                gaga.fig_plot_diff_2d(real, fake, keys, kk, a, fig,
nb_bins)
                i = i+1


    # remove empty plot
    phsp.fig_rm_empty_plot(nb_fig, ax)

    plt.suptitle(pth_filename)
    plt.tight_layout()
    plt.subplots_adjust(top=0.9)
    plt.show()

    #output_filename = 'aa.png'
    #plt.savefig(output_filename)
    #plt.close()
```