# Modelisation of electron diffusion in water for radiation therapy using Generative Adversarial Networks
# CS-433 Machine learning

Devianne Paul (CSE), Dormoy Camille (NX), Jégard Clémence (SV)
*EPF Lausanne, Switzerland*

***Abstract*—The aim of this project is to predict the best way possible collision events of electrons in water given initial conditions of the particles. During its diffusion in the water, the electron undergoes many interactions before losing all its energy and dissipating. At each encounter, it may or may not create a new particle with its own new properties. Using Generative Adversarial Networks (GANs)[1], we will model these different types of interactions in order to best predict the behavior of electrons over time and space in the media. Simulating elementary particle physics is very costly computationally. By using a pre-trained machine learning algorithm, one can simulate the same events with many less computations.**

## I. Introduction

Certain cancerous tumors are very resistant and difficult to irradiate even by resorting to surgery. The latter contributes significantly to the increase in the mortality rate of cancers. However, in view of the many advances in research and understanding of particle therapy, it is henceforth possible to irradiate tumors using highly energetic particle beams to treat resistant tumors. Radyalis is a software using the Monte Carlo [2] calculator to produce fast and accurate models of photon, electron, and proton beam treatments for radiation therapies. Within this project, we will work specifically on electron therapy by modeling the behavior of electrons in water with energies ranging from 0.1 to 20 MeV.

In radiation therapy, focusing the total energy on tumors is primordial to minimize the effect on nearby living cells. By generating events with the same statistics as real particle interactions, we are aiming to determine how the kinetic energy of an electron beam propagates through a media. Data files used to train the algorithm were produced by Geant4, a particle physics event simulator created by CERN researchers. After visualization through the Pandas Python library, the data is handled to bring out a readable and optimal data frame ready for Machine Learning process. From this data frame, a subsequent table is created including the probability of gamma particle emission, electron emission, or no-emission for each kinetic energy range. Ultimately, this data will feed a Generative Adversarial Network (GAN), a recent competitive machine learning algorithm for statistical modeling of particle physics events. This algorithm has already proven its efficiency in fast computation in particle physics [3]. Some tests that we used in our `final_test.ipynb` file were provided to us in order to be able to visualize the results of our models.

## II. Data Analysis

### A. Exploratory data analysis

We initially have ten .csv files containing data of different electron behaviors in water with initial kinetic energy values of 0.1, 0.2, 0.3, 0.5, 1.0, 2.0, 3.0, 5.0, 10.0, and 20.0 in millions of electron volts (MeV). An electron loses parts during its journey leaving 'interaction steps' which categorize the data. Since an electron is not influenced by its past events, an initial particle with 20 MeV will step by step go through all ranges of energy from 20 to 0 MeV. We so decided to exclusively use the file with the highest energy of 20 MeV. An electron can generate a secondary particle during its journey, introducing a new particle in the system to simulate. The data files contain 13 different features evolving across time and space.

*1) Step:* In this feature, the steps of electrons are indexed with integers. A particle dies when its kinetic energy is null. The following step row will be indexed back to 0. It is useful to keep track of the electron journey through the media but has no predictive power on the possible interactions.

*2) Position:* The features X(mm), Y(mm), and Z(mm) are intended to give the position of the electron in the three-dimensional space. As the media is homogeneous, this should not bring any information on the possible interactions.

*3) Speed:* The features DX, DY, and DZ are intended to give the direction of the electron in the three-dimensional space. We use this data to calculate the angle with respect to its initial direction as well as the daughter particle direction if any were to be emitted. These angles should be predicted by the model as they suffice to describe the change of direction of a particle. We compute and observe the normal distributions of DY and DZ and see that they are centered around 0 as the electron's initial direction is $(1, 0, 0)$. This proves the isotropic state of the media in which the electron propagates. One can observe similar behavior for the feature DX but as it starts from 1 it can only reach lower values.

*4) Kinetic energy:* KinE(MeV) represents the kinetic energy at its actual step.

*5) Variation in kinetic energy:* dE(MeV) corresponds to the energy loss of the particle across steps. It should be predicted by our model to predict the next step of energy after an interaction.

*6) Step length:* Distance traveled by the particle from step i to step i+1. This should also be predicted by our model to give the next position of our electron after one interaction.

*7) Track Length:* The total length of the particle's propagation since step 0. Track length is the summation of all the step lengths. This data is not necessary for our model since we aim to predict every step. The Track length should just be the sum of all step lengths from electron creation to its endpoint.

*8) Next Volume:* The NextVolume feature takes two arguments: "phantom" and "world". We decided to remove steps where the latter argument is set to "world" as it indicates an electron out of the system which corresponds to a particle leaving the medium.

*9) Interaction type:* The type of interaction (scattering, ionic, ...) is given by the ProcName feature. This feature is not taken into account in our model.

### B. Feature processing

In order to create an accurate model, we now want to select the features we will need to predict the emission of our electrons. First of all, we watch the general correlation of all our features with a map, see Figure 1. From this heatmap, we can see that StepLenght and dE(MeV) are highly correlated (0.99). This will be taken into account in our GAN as we only predict the dE feature and obtain StepLength by proportionality.

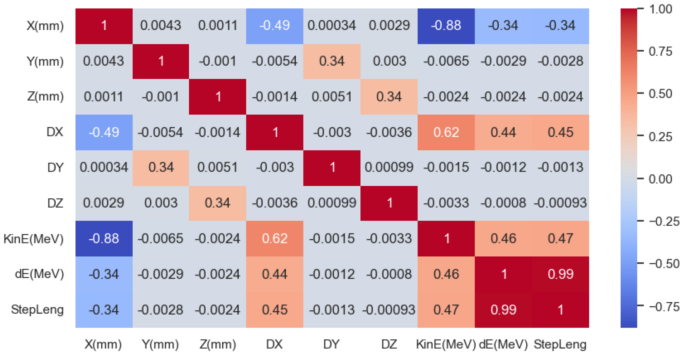


Fig. 1. Heatmap of our data

Moreover, as the media is isotropic, the information contained in DX, DY, and DZ can be summarized by one angle variable obtained by the scalar product from the initial and final directions for one interaction. This angle should be predicted by the GAN. When a daughter particle is generated from an interaction, the angles of its direction with the initial and final direction of the mother particle also need to be predicted. All these ways to process our feature will be done before the creation of our GAN.

## III. Methods Implementation

### A. Classification

The first thing our model must be able to do is a classification to determine if an electron with a given energy is going to result in an emission and if it's the case, if it emits an electron or a gamma ray (no other daughter particle types were observed).

From the previous insights we got with the feature processing of our data and physics-based restriction (such as isotropy), we decided to use as the only influence for the classification, the initial energy of our particle (KinE(MeV)).

With this information taken into account, we first tried to implement known multi-classification methods. To test and train them, we generated from our data a train and a testing set. Then, through the sklearn python library, we created a classification model using the following methods: Balanced decision Tree, Balanced random forest, Categorical Naïve Bayles (NB), Bernoulli NB, K-nearest-neighbors, and balanced logistic regression.

To rank our models, we computed the general accuracy. However, as shown in Figure (2), different emission-type of interactions can occur for the same kinetic energy. We so need to store the distribution of each interaction type for a given initial energy.

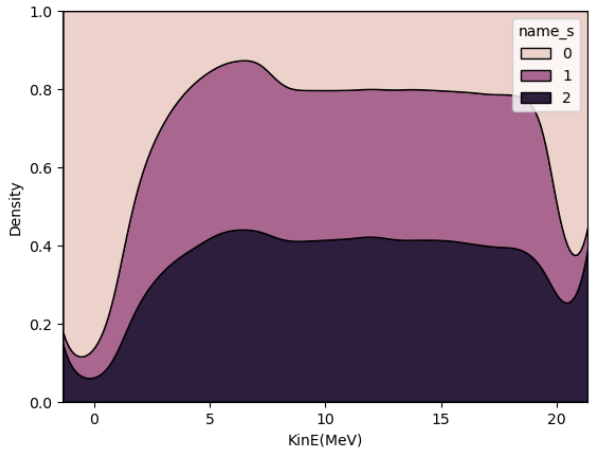


Fig. 2. Interaction type density for a given Kinetic energy of mother particle.{0: no daughter particle, 1: electron emission, 2: gamma-ray emission}

The most efficient way of storing such conditional probabilities is to generate a table that reports at each range of energy the probabilities of our three events.

### B. Generative Adversarial model

A Generative Adversarial Network is based on two functions: the discriminator and the generator. The discriminator network minimizes its prediction of whether a sample is real or fake. The generator network optimizes its weight in order to fool the discriminator as much as possible. It is a 2-player game. The more iterations, the more 'real' are the fake samples from the generators. This model addresses well to our

problem since we aim to mimic the statistics of the electron interactions.

GANs do not take continuous features as inputs. We should then create different GAN models for specific energy ranges. The energy ranges need to be chosen wisely. A kinetic energy would first be mapped onto an energy range. For that energy range, the corresponding GAN model will predict an output with the same statistics no matter the kinetic energy, if it is in the same energy range.

*1) Choosing energy ranges:* We have identified the variables we need to predict. By plotting their distribution, from initial data, for specific energy ranges, we were able to find energy ranges that give the same distribution for the outputs (see `Energy_ranges` notebook). The complete model is displayed on Figure 3.

*2) Building the GANs:* The GANs models are built using the PyTorch library. The generator and discriminator networks are built with similar architecture (see `GAN_function`). The networks are composed of three hidden layers with 128 neurons by default for each layer. The activation function between layers is the Leaky ReLU activation function with default negative slopes. The generator network receives as input a noise variable $z$ of length 100 which follows a Gaussian distribution. Depending on the energy range and the type of particle emitted (or none), the generator outputs two or five variables. Once trained, the GAN generators are saved. They can then be used to generate new events fitting the statistics shown from the initial data. For each iteration, the network takes 64 random data samples for training. Learning rates for the generator and the discriminator are set to common values $l_g \approx 1 \cdot 10^{-4}$ and $l_d \approx 5 \cdot 10^{-4}$.

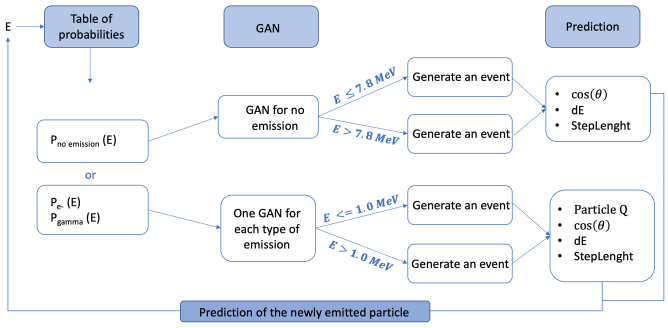


Fig. 3. Model for the prediction of the parameter of a given particle from its initial energy E. ( We first map the energy we are looking at on the table of probabilities. According to the corresponding probabilities distribution, we generate a GAN model (In total 6 GAN will be generated). From these models, we predict the next step of the particle and reenter the scheme with the new computed energy.

## IV. Results

Once the GAN models are trained, we can evaluate their efficiency on predicting the variable with the same statistics. The models are very sensitive to the number of variables they need to predict. They are also dependent on the distribution of the variable to predict. For instance, for the variable `cos_theta` when there is no emission of daughter particle and energy is below 7.8 MeV, the distribution is normal. Since the generator starts with an input noise from a Gaussian distribution, it matches after only 1 iteration the statistic of the variable.

When the number of variables to predict is bigger, and the distribution more complex than a Gaussian, the GAN needs more iteration to match with the statistics. An example is displayed in Figure (4).

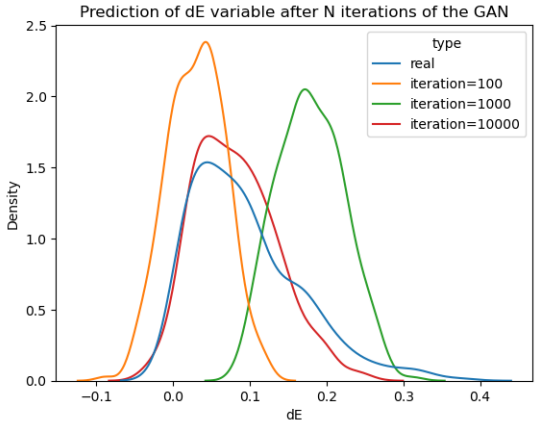


Fig. 4. Kernel density estimate of $\Delta E$ for real data samples and the third GAN model fake samples: emission of e- particle and $E_{kin} < 1.0$ MeV.

One can observe that when we have a few iterations the generator produces a normal distribution while for many iterations, the curve starts matching the complexity of the statistics. [1].

Once the GAN models are trained, we can use them to generate events for electrons in a system with given kinetic energy that will follow the same statistics as the one observed in the data. The events will predict the next step of the particle using all the predicted variables generated. The daughter particles created by the electron during its journey will also be simulated.[2]

However, we encountered one issue when the initial kinetic energy of the electron is too high. As seen in Figure (2), the electron produces many daughter particles for energies above 5 MeV. Therefore, for one electron simulated, many daughter particles will also have to be simulated. Ultimately, the number of particles in the system grows exponentially making the simulation infinitely long. To tackle this issue, we simulated two systems: in the first one, the initial energy of the electron is restricted below 0.3 MeV ensuring that $\mathbf{E}[N_{daughter}] < 1$ from the creation of the electron to its end. The second system does not simulate daughter particles generated by the mother particle but accounts for every initial energy from 0 to 20 MeV. Both systems are represented in Figure (5). What is displayed

[1]A more explicit comparison of the real data samples and the fake data samples is presented in the appendix on Figure (6)

[2]For gamma rays we use a random generator that does not use the GAN models who are only fitted for electron particles diffusion.

on this figure is actually the $\log(\Delta E)$ of all the electrons that went through each pixel of the displayed space.

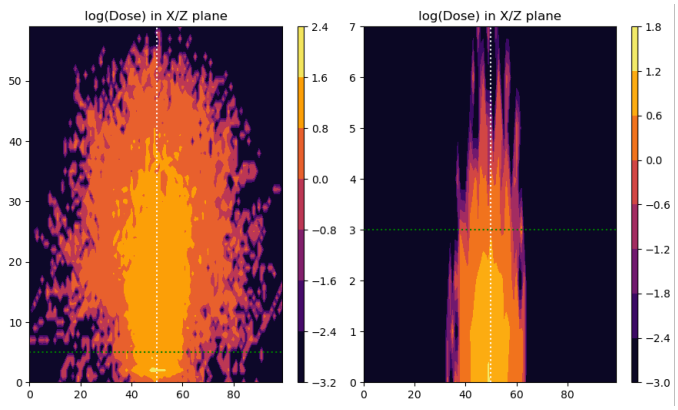


Fig. 5. Heatmap energy loss $\Delta E$ of the particles for the 1st system (right) and the 2nd system (left).

## V. Discussion

*1) On the efficiency of GANs:* Figure (4) showed how the generator is able to match the complexity of the statistics when the iteration is high enough. It illustrates the convergence of the generator networks toward real statistics.

We also observed the effect of the initial noise distribution on the predicting power. An interesting prospect would be to change the distribution of the initial noise and see if the networks converge faster or slower. Some predicted variables have complex distributions compared to the initial Gaussian distribution. We expect to have better convergence for this variable when the initial noise distribution presents a similar curvature.

*2) On the issue of the models:* For a very high number of iterations even complex statistical distributions are mimicked well by the generator (see Figure (6) in Appendix). However, in that case, one can notice that the network tends to cluster too much the data, sometimes losing the sparsity of the distribution. For instance, on the `cos_theta` vs `cos_psi` plot, the fake samples are clustered on one single point while the real data shows some sparsity for these variables.

To counter this effect, a few solutions can be studied. The batch size at each iteration could be reduced, making the sparsity of the real distribution more impactful on the weights. Augmenting the data set is also an option. Ultimately, the networks might be too complex for the data, so one could try to remove one or two layers, and even reduce the number of neurons per layer.

Also, by splitting depending on the energy ranges, the models lose accuracy for special energies. This is not observable here since the real data is also grouped the same way as the model, but in real conditions in the system, this might produce inconsistencies such as an electron losing more energy than what it initially had. This might be solved by refining the energy ranges even more.

## VI. Summary

The aim of our project was to predict in the best possible manner the collision event of an electron in water from its initial characteristics. To find the best way of doing this, we first analyzed in detail the data sets provided and decided to select only the one with energies going from 0 to 20 MeV. After a precise analysis of our features and the creation of a clean data set taking into account all our feature processing, we create a prediction model. This prediction model first maps the energy of the particle under study to its energy range corresponding. It then collects the corresponding probabilities of events from the table of probabilities and finally applies the corresponding GAN model to the particle.

The use of GANs for simulating particle physics problem proved to be very efficient to tackle our problem. The generator networks converges towards the statistical distribution of real interactions. Some parameters still need to be adjusted in readiness for optimization. Over-training is also something to account for when choosing the networks complexity. Therefore, the use of GAN models for fast computation in radiation therapy simulation displays interesting prospects. Additionally, an interesting view of the problem would be to use GANs for image generation of figures like Figure (5). We aim to learn the structure of the particle diffusion in the media. It then means using a convolutional recognition of the different energy dose released in the media. Such approach has proven to deliver satisfactory results. [4]

## VII. Acknowledgments

We would like to thank professor Sani Nassif for providing us with this interesting project and for his valuable time and feedback all along our progress in the project.

## References

[1] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[2] "Radyalis website," https://radyalis.com.

[3] F. Carminati, A. Gheata, G. Khattak, P. M. Lorenzo, S. Sharan, and S. Vallecorsa, "Three dimensional generative adversarial networks for fast simulation," *Journal of Physics: Conference Series*, vol. 1085, p. 032016, 9 2018. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1085/3/032016

[4] F. Rehm, S. Vallecorsa, K. Borras, and D. Krücker, "arXiv : Validation of Deep Convolutional Generative Adversarial Networks for High Energy Physics Calorimeter Simulations," *CEUR Workshop Proceedings*, vol. 2964, mar 2021. [Online]. Available: https://cds.cern.ch/record/2759294

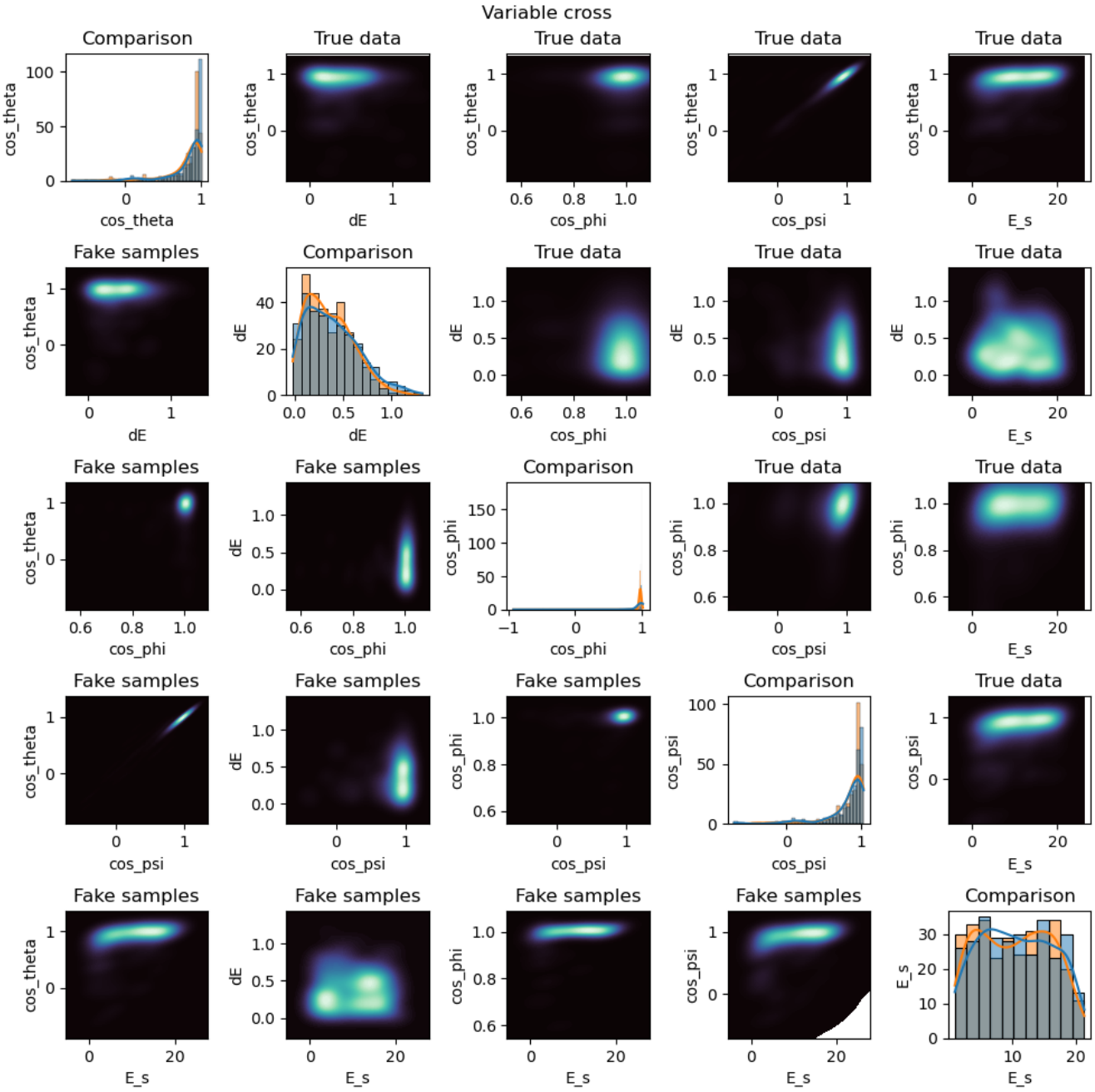


Fig. 6. Variable cross: Kernel density estimate of all the interaction variables for real data samples and the sixth GAN model fake samples: emission of gamma ray particle and $E_{kin} > 1.0$ MeV. The fake samples are displayed on the below part of the matrix and the corresponding variable cross for real samples is the element symmetric with respect to the diagonal of the matrix. The number of iteration is $N = 2 \cdot 10^4$. Keys for Comparison plots {Blue: Real samples, Orange: Fake samples}.