CS-433 - ML4Science
Modelling Energetic Particles in Matter

Adam Dandachi, Salma Ed-Dahabi, Salim Najib
supervised by Prof. Sani Nassif
EPFL - Fall 2022

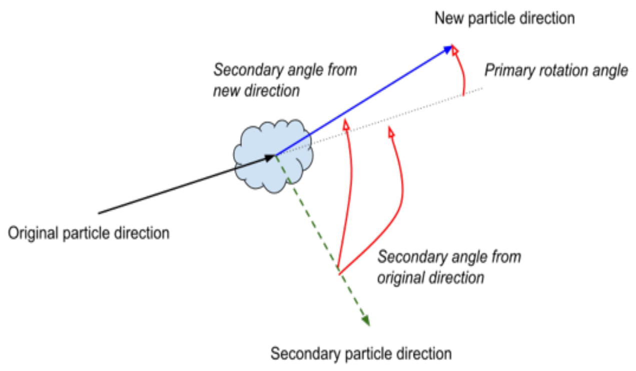
New particle direction
Secondary angle from new direction
Primary rotation angle
Original particle direction
Secondary angle from original direction
Secondary particle direction

Fig. 1. Compton scattering

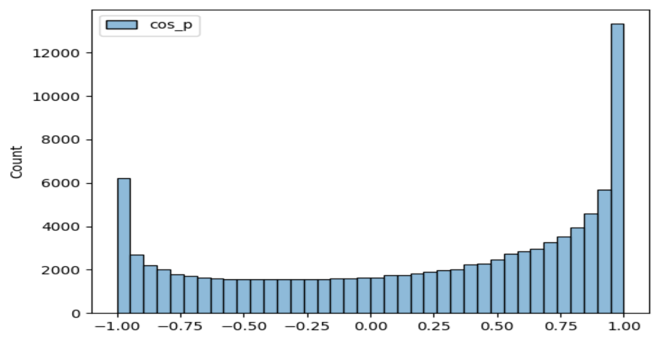

Fig. 2. Expected `cos-p`

***Abstract*—Nowadays, Machine Learning provides a lot to many fields, in particular physics research. Inference from data simplifies the process of finding relevant information when the scale of the data becomes humanly unmanageable. Deep learning - *in particular generative adversarial models* - enables the simulation of energetic particles' behaviour in matter, following the samples' statistics of particles produced by CERN's Geant4. This paper is a project report on the implementation of such a model.**

I. Introduction

A. Physical context

A *parent* particle (e.g a photon) in matter (e.g water) at a given energy (e.g 3 MeV) and time may have 1 of N events and will result in a change of *state*, possibly interacting with a new *child* particle. State encompasses particle position, direction and kinetic energy. Figure (1) depicts one of those events, the Compton scattering [1]. Another one of these events is the Rayleigh scattering [2].

Simulating such physical processes has critical applications including healthcare [3], nuclear physics and space science. These simulations are usually done using poorly parallelizable physics-based software [4]. The goal of this project is to achieve similar results using machine learning methods, which are usually GPU-friendlier.

B. Problem definition

We model the movement of a particle inside of material as a sequence of *steps*, grouped into *tracks*. During each step, the particle travels in a straight line, then a new step begins once an event happens to the particle, deviating it from its course, altering its kinetic energy and possibly interacting with another particle. The task is, *given a particle's position, energy and direction*, to predict the next step it will take.

II. Dataset

CERN's Geant4 toolkit for the simulation of the passage of particles through matter [5] was used to generate ground truth tracks for parent photons of initial energies ranging from 0.1 to 6 MeV. Steps were isolated from these tracks, keeping the following random values (all others were deterministic):

- `en-p`: Initial kinetic energy of the photon, between 0.1 and 6 MeV.
- `dist-p`: Distance covered in that step, exponentially distributed, see (8).
- `cos-p`: cos of the photon's direction at the next step w.r.t the current step, roughly $\beta(\frac{1}{2}, \frac{1}{2})$ distributed, see (2).
- `de-p`: Amount of energy deposited by photon in matter during step. Restricted to the case of emission, it is tightly Laplace distributed.

And if a child electron particle was emitted:

- `en-c`: Kinetic energy of the child, see (3).
- `cos-c`: cos of electron's direction after event with photon's initial direction.

The dataset has a few particularities. Most importantly, there are few ($\sim 0.18\%$) steps with no associated emission.

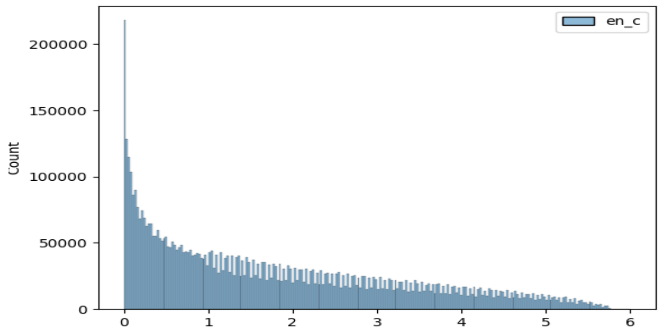

Fig. 3. Expected `en-c`

III. Models and Methods

A. Overall system architecture

Since the goal is data generation, Generative Adversarial Networks [6] are a natural choice of ML model, or more precisely Conditional GANs [7] since the data is endowed with labels. This choice is also motivated by previous uses of cGANs in physics research [8]. Performance tests revealed that the best strategy is to cascade multiple cGANs as shown in (4) rather than having one cGAN do all in one go. Putting everything in one model promoted undesirable Gaussian-like properties to the output signals, which are highly undesirable here given the distributions of the ground truth data features.

The label varies from a cGAN component to another. To generate a new step for a given photon, the distance `dist-p` it covers during that step is generated using its energy `en-p`. Then a simple classifier decides whether or not an electron is emitted. The case of no emission will be detailed later. In case of emission, three cGANs are run in cascade sequentially predicting `en-c`, `cos-p` and `cos-c`. At each step, the label is comprised of `dist-p`, `en-p`, plus the outputs of previous cGANs in the sequence.

B. Conditional GANs

See figure (5) for the general diagram of a cGAN.

1) Network architecture: Small networks suffice for all critic and generator networks as the dimensionality of the data is low and its inner interactions, although highly nonlinear, are relatively simple. Two hidden layers with output dimensionality 128 or 256 sufficed in all cases. Since computing performance is an initial motivation of the project, unnecessary computational complexity at runtime should be avoided. Figure (6) shows the generator network for `cos-c`.

2) Training: Each cGAN was trained individually using the L_2-regularized Wasserstein-1 loss:

$$\mathbb{D}_W(p_g, p_d) = \inf_{\gamma \in \prod(p_g, p_d)} \mathbb{E}_{(x_g, x_d) \sim \gamma} [||x_g - x_d||]$$

where p_d is the target distribution, p_g the generator's distribution and $\prod(p_g, p_d)$ is the set of all joint distributions with marginals p_g and p_d. The loss function is minimized using the

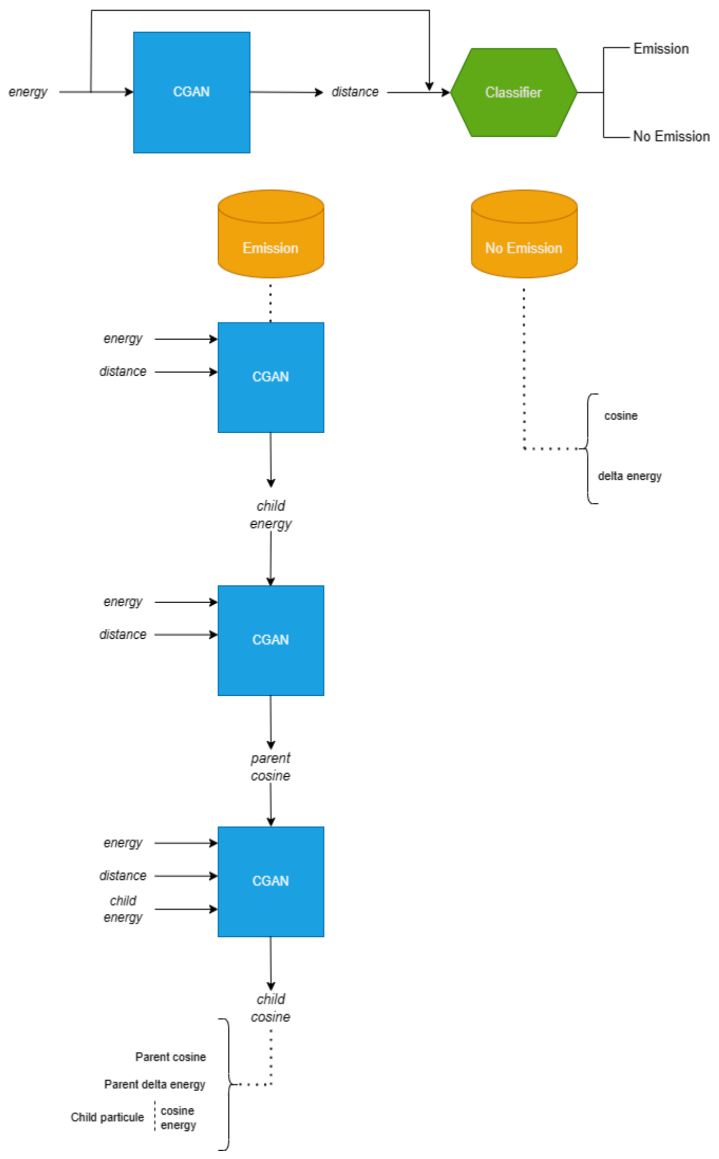

Fig. 4. System structure

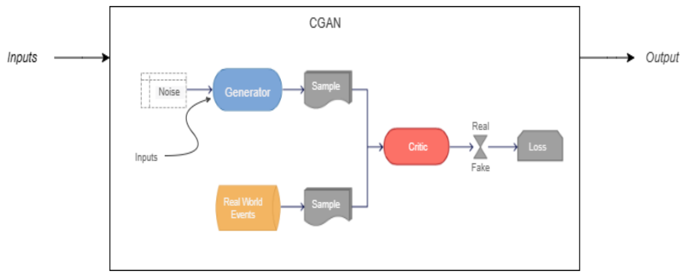

Fig. 5. cGAN structure

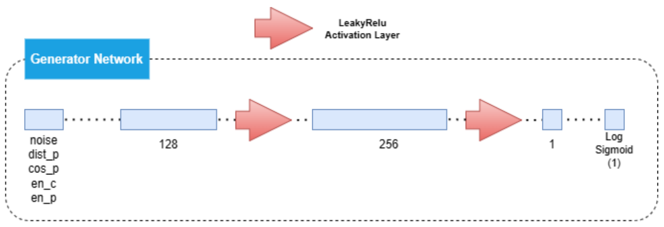

Fig. 6. Generator network for `cos-c`

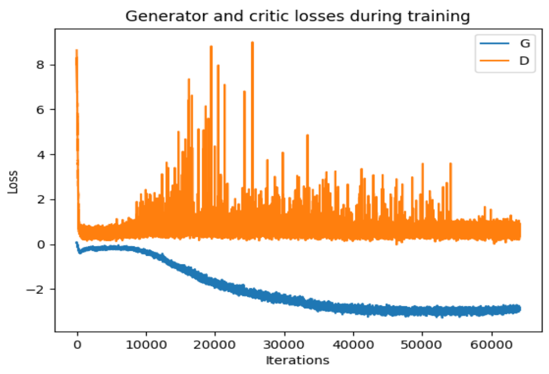

Fig. 7. Training losses

Adam algorithm with weight decay. A typical plot of generator and critic training losses is shown in figure (7), where the Nash equilibrium is reached at iteration ∼53,000.

3) Hyperparameter selection: The single most important hyperparameter turned out to be the *noise distribution* that's input to the generator. It systematically turned out that matching the noise distribution roughly with the target marginal distribution worked best. Thus exponential noise is used for the `dist-p` cGAN; a distribution similar to the β distribution is used for `cos-p`: the Kumaraswamy distribution [9] whose density is cheaper to compute, etc.

C. Other components

1) Emission classifier: The classifier with inputs `dist-p` and `en-p` deciding if an emission occurred during that step is a simple L_2-regularized logistic regression model. The input is feature expanded with polynomial expansion as well as applications of $f(x) = e^{-x}$, as these features are common in physics formulae. Classes are also weight-balanced, the no-emission class having $\sim 0.18\%$ of the data. The measured F1-score is 0.993028.

2) Randomly generated signals: Two outputs are randomly generated because they are both almost constant in the given data, making cGANs perform poorly. The first one is `de-p` in the case of emission, which is generated as a tight Laplace sample around the dataset's mean. Indeed, the correlation between the parent energy and the deposited energy `de-p` is almost 0 in the dataset, thus it is fine to generate `de-p` independently. The second one is `cos-p` in the case of

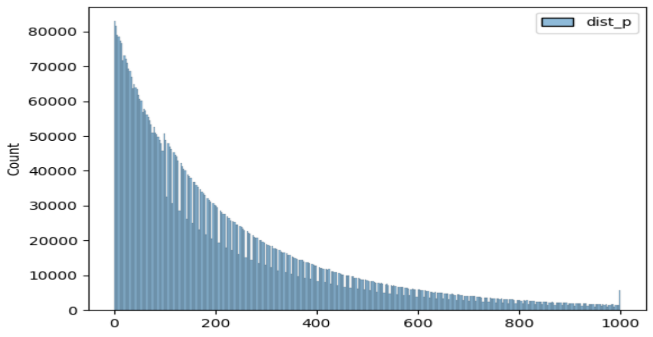

Fig. 8. Expected `dist-p`

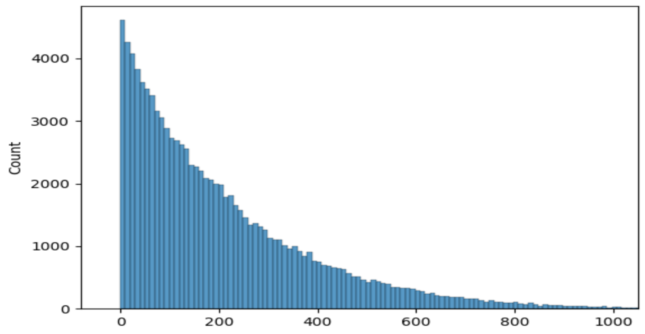

Fig. 9. Obtained `dist-p`

no emission, which is $1 - |\text{Laplacian}(0, \sigma)|$. In the given data, when conditioning on no-emission, `cos-p` has a high correlation with `dist-p` and `en-p`. To account for this in the generated samples: $\sigma = \frac{c}{1+\texttt{dist-p} \times \texttt{en-p}}$, the idea is to tight the Laplacian sample around 0 when `dist-p` and `en-p` increase, hence making `cos-p` increase as well.

IV. Results and Discussion

A. Individual components' performances

To visualize the individual cGANs' performances, we plot histograms of their outputs when fed the data's labels and compare them with the expected histograms. Figures (8) and (9) show the obtained and expected `dist-p`. Similarly, figures (10) and (11) show the expected and obtained `cos-c`.

B. Overall system results

To visualize the overall joint distribution of the system's output, the file `test.py` was provided. It outputs the contour plot of the energy `de-p` deposited in a box of water of size $2 \times 2 \times 2\ mm^3$ by $300,000$ photons, starting at approximately the same point with a uniformly random initial direction and energy between 0 and 6 MeV. The steps are simulated for a given particle until it has no more energy or it leaves the box. See figure (12). Simulating the tracks of $100,000$ particles on

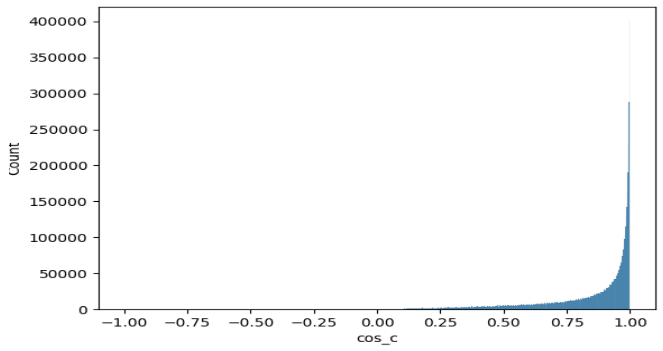

Fig. 10. Expected `cos-c`

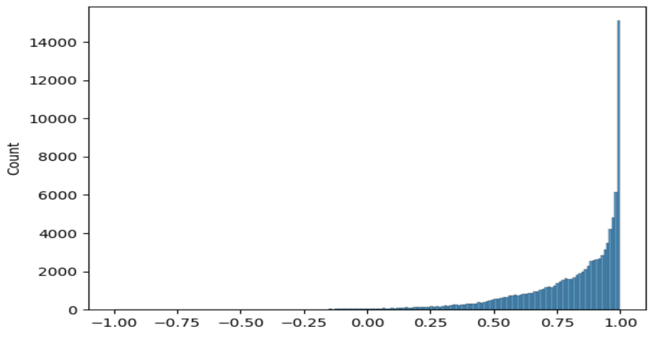

Fig. 11. Obtained `cos-c`

a laptop with an AMD Ryzen 7 CPU and an RTX 3060 GPU takes about 12 minutes.

V. Conclusion

With some basic knowledge of the physics behind this project, machine learning has again shown the extent of its effectiveness. Indeed, thanks to the performance of cGANs, there is no need to run costly physical Markov Chain Monte Carlo models to achieve our initial goal without being extremely time-consuming for the trained physicist. With more time at disposal, another strategy worth exploring is the use of stable diffusion instead of cGANs, i.e: generating groups of tracks at once instead of individual steps. This would probably improve the overall bulk results.

References

[1] Wikipedia contributors, "Compton scattering — Wikipedia, the free encyclopedia," 2022, [Online; accessed 20-December-2022]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Compton_scattering&oldid=1127017625

[2] ——, "Rayleigh scattering — Wikipedia, the free encyclopedia," 2022, [Online; accessed 20-December-2022]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Rayleigh_scattering&oldid=1128442417

[3] P. Mayles, A. Nahum, and J. Rosenwald, *Handbook of Radiotherapy Physics: Theory and Practice*. CRC Press, 2007. [Online]. Available: https://books.google.ch/books?id=v68J1dgCEn8C

[4] J. A. et al., "Recent developments in geant4," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 835, pp. 186–225, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900216306957

[5] "Cern's geant4," https://geant4.org/, accessed: 2022-12-20.

[6] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017. [Online]. Available: http://arxiv.org/abs/1701.00160

[7] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[8] A. B. Farimani, J. Gomes, and V. S. Pande, "Deep learning the physics of transport phenomena," *CoRR*, vol. abs/1709.02432, 2017. [Online]. Available: http://arxiv.org/abs/1709.02432

[9] S. Dey, J. Mazucheli, and S. Nadarajah, "Kumaraswamy distribution: Different methods of estimation," *Journal of Computational and Applied Mathematics*, vol. 37, 03 2017.

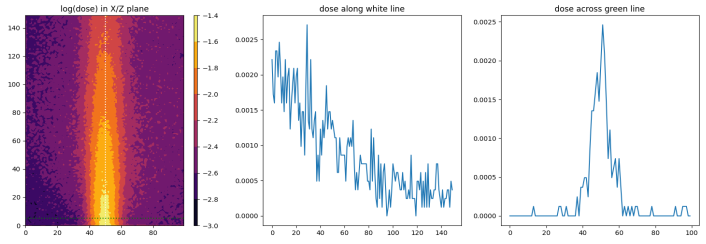

Fig. 12. Running `test.py`