Neural Networks for Cosmic Ray Simulations

Pranav Sampathkumar,[a,*] **Tanguy Pierog**[a] **and Antonio Augusto Alves Jr.**[a,b]

[a] *Institute for Astroparticle Physics, Karlsruhe Institute of Technology, Karlsruhe, Germany*

[b] *Brazilian Synchrotron Light Laboratory, Brazilian Center for Research in Energy and Materials, Campinas, Brazil*

E-mail: pranav.sampathkumar@kit.edu, tanguy.pierog@kit.edu

A neural network based model in order to learn the solution to the 1D cascade equation governing the evolution of Extensive Air Showers (EAS) is presented. The neural network is then used to generate the spectra of secondary particles at every height slice. The ability of the network to learn the function to generate the next iteration in shower development is showcased. Pitfalls in using the network in generating the entire shower is discussed. A sequential network model, which can iteratively generate the entire shower from an initial table is presented. We show that the network learns to generate a single step with approximately 5% error and how the network is accurate in the later parts of the shower and error prone in the early parts of the shower.

38th International Cosmic Ray Conference (ICRC2023)
26 July - 3 August, 2023
Nagoya, Japan

ICRC2023
38th International Cosmic Ray Conference
The Astroparticle Physics Conference

[*]Speaker

1. Introduction

Cosmic rays are messengers from distant sources and they provide us a way to study the high energy processes in the universe which are beyond the reach of current accelerators (beyond 10^{17}eV primary energy). These high energy particles when they interact with our atmosphere create showers of secondary particles known as Extensive Air Showers (EAS), these secondary particles and the associated electromagnetic emissions from the charged particles are detected by various detectors around the world [1–3]. Explicit Monte Carlo is the most common method to make theoretical predictions of these high energy processes using experimental data from these detectors. Despite the high energy of the primary particle, analyzing them is challenging because of their low intensities which make it hard to gain enough statistics for the theoretical predictions. This is additionally complicated by the fact that computational complexity of an EAS simulation increases as a power law with energy (Figure 1). Thus analysing high energy showers is computationally heavy and there is a need for new techniques in order to lower the computational load.

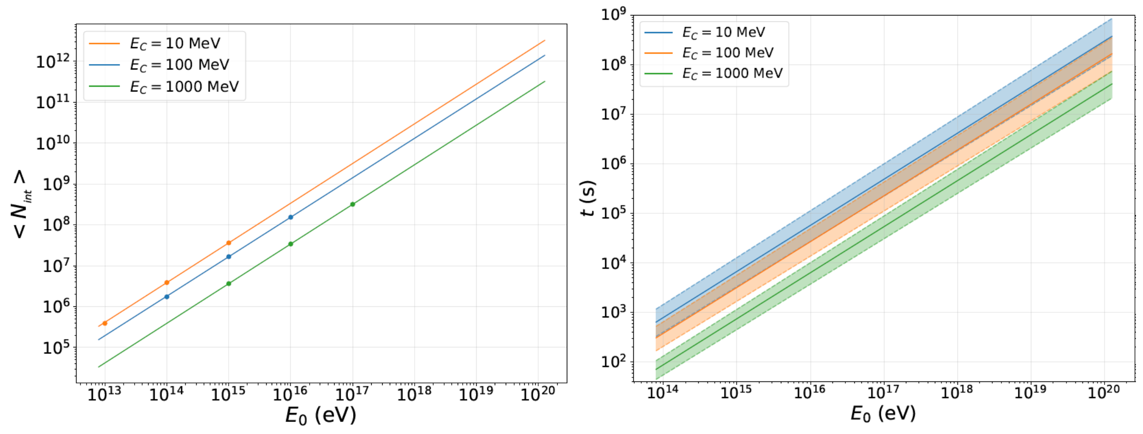

Figure 1: The left plot shows variation of the number of particles and the right plot shows the variation of the time of computation with respect to the energy of the primary particle [4]. E_c is the cutoff energy, beyond which the particles aren't tracked in a simulation. We see as E_c increases, the simulations become faster, but the power law remains the same. Hybrid methods change the power law, making it scale better at higher energies

One of the commonly used techniques to reduce the computational load is to perform weighted sampling on the sub-showers. These methods are in general called *thinning* methods [5], which vary based on the weighted sampling algorithm used. These algorithms don't simulate all the particles explicitly, but only a fraction of the shower. They then assign a weight to the simulated particles in order to account for the skipped particles. While this method reduces the computational cost of simulations, such a sampling method also introduces artificial fluctuations as we skip more and more particles. This can be controlled by imposing a maximum weight on the particles, but that too limits us in reducing the computational cost. Another method is to use tabulated sub-showers [6]. Though such pre-tabulation has a huge memory footprint and makes it less flexible to change in simulation conditions. An alternate method is to numerically solve equations describing shower development. In this method, high energy cascades are simulated explicitly, whereas low energy cascades are solved numerically using cascade equations. Cascade equations describe the mean behaviour of the shower, thus by combining explicit simulations and numerical solutions, we can capture both the mean behaviour of the shower and its fluctuations.

In this work, we explore the use of neural networks to perform these hybrid simulations. The advantage to using neural networks is, we can learn the solution to the cascade in a data driven manner. We show a neural network approach to learn the solution to the 1D-cascade equation. Sequential networks are used to generate the mean behavior of the shower in an iterative manner.

2. Cascade Equations for Hybrid Simulations

A simple Monte Carlo process can be described by a linear cascade equation. Current approaches to cascade equations [7] involve manually constructing the cascade equations using theoretical physics insights. The theoretical understanding enables us to derive the analytical form of the cascade equations incorporating various processes such as hadronic and electromagnetic interaction models. These analytical forms are then solved numerically using suitable boundary conditions. The problem in this case is an initial value problem where the initial conditions are provided by an explicit Monte Carlo simulation. The attempts towards cascade equations differ in their theoretical physics inputs, but the underlying linear structure remains the same. For the one dimensional evolution of the shower, the state of the shower in phase space is completely described by the energy spectrum at a particular height. So, we see the linear evolution of the energy spectrum with grammage as our evolution parameter. The general structure of a cascade equation can be given as follows.

$$\frac{\partial n_i(E,X)}{\partial X} = \overbrace{\sum_a \int_E^{E_0} \sigma_a n_a(E',X) P_{a\to i}(E,E')dE' - \int_{E_{min}}^{E} \sigma_i n_i(E',X) P_{i\to a}(E,E')dE'}^{\text{Production and Decay Terms}} \\ \underbrace{+S_i(E,X) - \sigma_i n_i(E,X) - \alpha \frac{\partial n_i(E,X)}{\partial E}}_{\text{Source, Interaction and Loss terms}} \tag{1}$$

where, $n_i(E,X)$ is the energy spectra at given depth X. Solution to the cascade equation provides us with the energy spectra $\forall X$. In this work, height and grammage [*] are used interchangeably, as both can be used as an evolution parameter for the cascade equation.

The *Production* terms in Equation (1) account for the creation of new particles and they are additive in nature. They increase the number of particles in a particular energy bin. The *Decay* terms account for the loss of particles. They reduce the number of particles in a particular energy bin and create particles in lower energy bins. *Source* terms account for sudden appearance and disappearance of particles, not accounted for by the other processes in the cascade equation. It can account for new particles entering the shower at a different height, or particles leaving the cascade equation to be handled differently in a hybrid manner. *Interaction* and *Loss* terms account for loss in energy of particles. They also account for the loss of particles by not being involved in the cascade after the threshold energy. This involves ionization losses in the atmosphere, etc. In more general

[*]grammage is the amount of mass passed through by a particle, $X = \int_0^H \rho(h)dh$, with units g/cm^2

terms, this equation could be written as

$$\frac{\partial n_i(X)}{\partial X} = \sum_j W_{ij} n_j(X) \tag{2}$$

where, W is the *transfer matrix*.

This system of linear coupled differential equations is then solved using integrators. This provides us with the energy spectrum at every height, giving us the projection of the shower along the longitudinal axis. The analytical form of such a solution could be written as,

$$n_i(X) = \int_0^X W_{ij} n_i(x) dx \tag{3}$$

where $n_i(0)$ is the initial condition for the solution.

CONEX is one such simulation program, which employs the hybrid method. The cascade equation is based on an updated algorithm from Ref. [8] and the above-threshold particle cascades are done explicitly.

One of the problems with extending this approach to higher dimensions is the lack of basis which describes the shower completely, and also the lack of the analytical form of the cascade equations in higher dimensions.

3. CONEX inspired sequential network approach

Integrating the cascade equation provides us with the energy spectrum at every height. Since the structure of the cascade equation is that of an initial value problem, the solution is equivalent to a *functional* which takes the energy spectrum at a particular height and provides us with the energy spectrum,

$$f : n(E, X) \longmapsto n(E, X + \Delta X) \tag{4}$$

$$f[n] := \int_X^{X+\Delta X} W_{ij} n_i(x) dx \tag{5}$$

this functional helps us iterate through the height and generate the entire shower from an initial condition.

3.1 Neural Networks

Neural networks are universal approximators, which take a functional form of $\vec{y} = \sigma(W \cdot \vec{x})$ for every layer of the neural network. The function is used repeatedly to add additional layers to the neural network, also known as the *depth* of the network. The size of the vectors, $\vec{x}$ and $\vec{y}$, are referred to as the *width* of the network. The parameters of the network are fine-tuned using the backpropagation algorithm, where the chain rule of differentiation is used to compute the differentials on the parameters and then optimized using some form of a gradient descent algorithm.

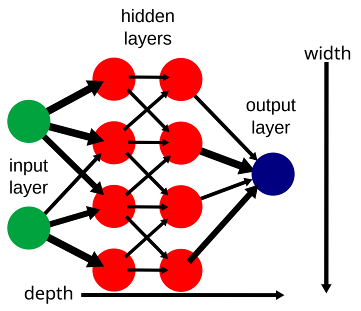

Figure 2: Basic schematic of a neural network showcasing the width and depth of a neural network. Each edge corresponds to a parameter which is multiplied and each node corresponds to a parameter which is added. Each node also has an *activation function* which adds non-linearity to the network making them universal approximators

Sequential networks are networks used on sequential datasets, in order to generate an output by repeatedly applying the network on the input. In our case, the output is also generated sequentially. The such a network is trained by using the backpropagation algorithm after every output. This is useful in situations where the network needs to be used repeatedly (such as generating an Extensive Air Shower from an initial condition), the network fine tunes its parameters from further repetitions in order to keep the errors in check. *Recurrent* neural networks (RNNs) [9] are a subclass of sequential networks, where the network is applied recursively and the subsequent networks in the sequence are connected, so that certain nodes can affected the output of the network down the sequence. These connections are called *hidden states*. Since cosmic ray shower generation is a simple Monte Carlo process, it is *memory-less*. ie, the evolution of a shower at a particular height slice is independent of its evolution until that point. So, we use a recurrent neural network, without a hidden state to mimic the memory-less behaviour of a cosmic ray shower.

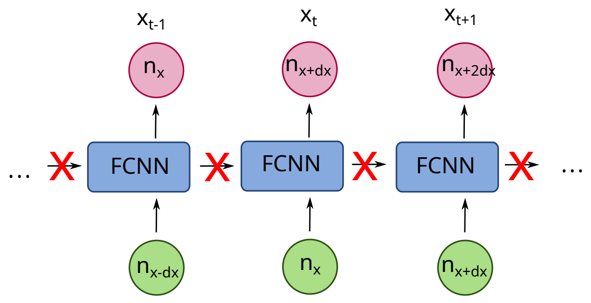

Figure 3: Basic schematic of a sequential neural network showcasing how the network is used iteratively. We see that hidden state which is passed between iterations is disabled to mimic the memory-less nature of simple Monte Carlo processes. We use the same network for all the iterations, and the parameters are tuned after a various number of steps.

3.2 The dataset

As an initial check of the idea, we take a purely electromagnetic cascade in air (US-Standard atmosphere). The network is tested on its ability to learn the functional as well as using the functional to generate the entire shower. As the cascade equation describes only the mean behaviour of the shower as opposed to the fluctuations in the explicit shower simulations, CONEX is used to generate the training dataset. This dataset conveys the mean behaviour at every height slice. We get the mean energy spectrum for the three particles, electrons (e^-), photons (γ) and positrons (e^+). As the number of particles in each energy bin varies across several orders of magnitude, we pre-process the table by taking the logarithm of the numbers. The numbers are added with a smoothing factor of 10^{-14} before taking the logarithm to avoid the pole at zero.

3.3 Network Details

We check if a network learns the functional useful in stepping through the height and also, the utility of the functional in generating the entire shower. The network is implemented using `PyTorch` [10]. The network has 5 hidden layers with 512, 256, 128, 256 and 512 nodes each. `LeakyReLU` [11] is used as activation function in all the layers including the final one because the number of particles in the spectrum can never be negative. Mean Squared Error (MSE) is the loss we try to minimize and the minimization is done using the ADAM algorithm [12]. We use a "memory-less" recurrent neural network, where the network is applied iteratively in sequence for multiple steps. Such *sequential chains* are useful in fine-tuning the network and helping it learn the right parameters to be able to generate the entire shower. These sequential chains are very important in ensuring that a network training on a single step doesn't lead to unphysical situations when applied iteratively to generate the entire shower. The length of the sequential chains is a hyper-parameter which was optimized to balance between computational cost of training and the need to fine tune the final network. The results in this paper are done using sequential chains of ten steps.

4. Results and Conclusions

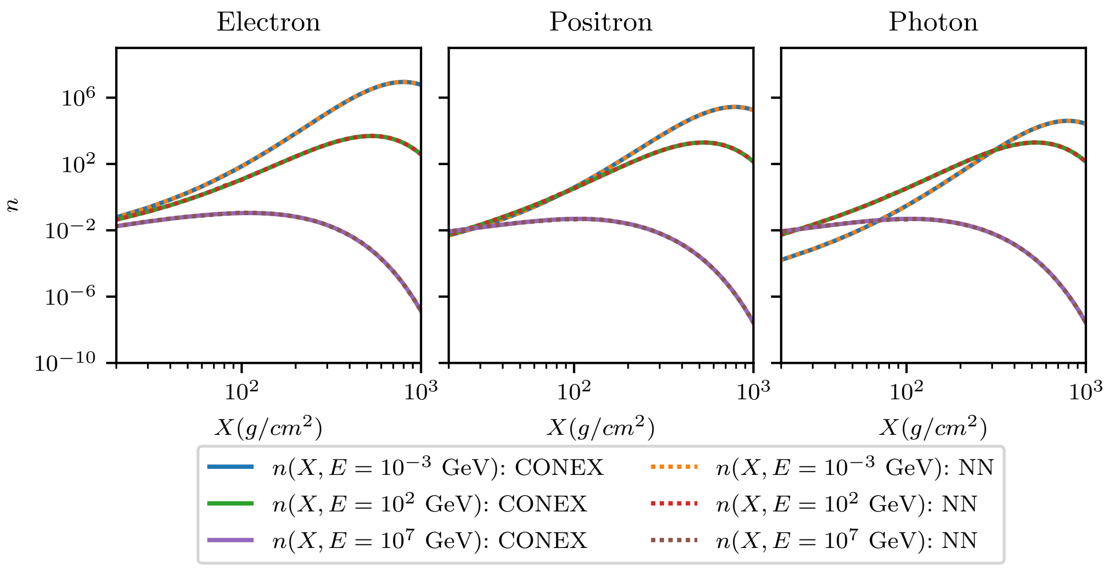

Figure 4: Comparison of CONEX generated showers and Neural Network generated showers after a single step of shower generation. Thus in this case, various height bins need to be compared independently since the network wasn't used sequentially.

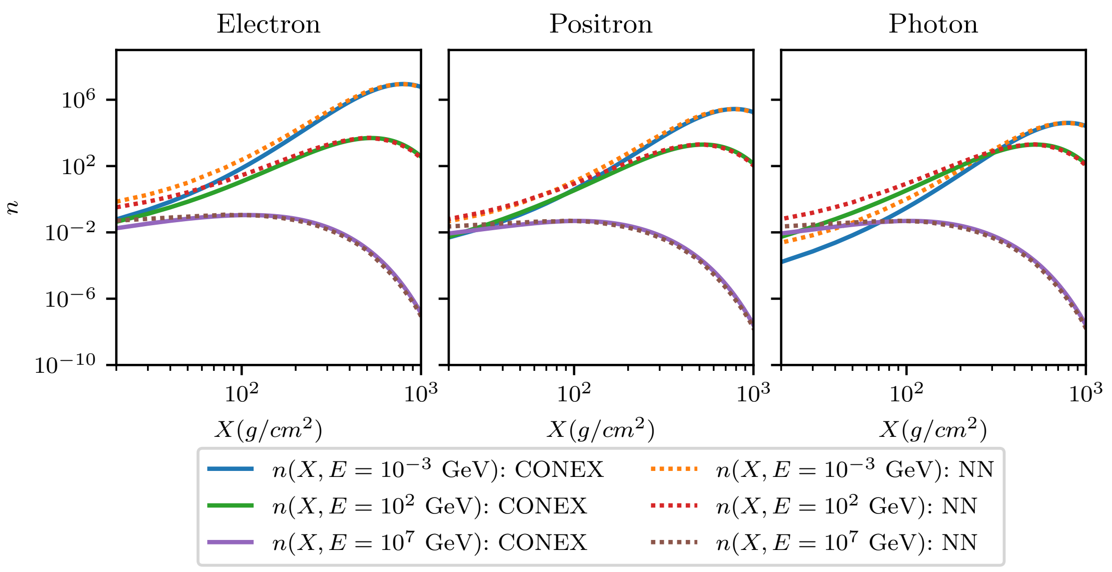

Figure 5: Comparison of CONEX generated showers and Neural Network generated showers after the entire shower is generated by from an initial table. We see that the network isn't accurate in the early parts of the shower.

The initial check is to see if the neural network is capable of learning the stepping functional (5), The results can be seen in Figure 4. The figure shows the ability of the neural network to learn the stepping functional across various energy bins. The network learns the stepping functional with approximately 5% error. When we tried to use the learned stepping functional to generate the entire shower, we found that the shower quickly leads to unphysical situations which aren't seen in the training dataset. This completely breaks the network and makes it not possible to use it to generate the entire shower. The network is then trained sequentially, to fine-tune the network for iterative generation. One can see the results of that in Figure 5. We note that the accuracy of the network is small in the early parts of the shower. One possible reason for this is the sparcity of the tables in the early parts of the shower making it hard for the network to understand the physical correlations in the cascade. Nevertheless the shower generation using neural networks indeed works for the later parts of the shower. Further work is needed in order to check the underlying physics of the generated showers. The network also struggles with learning the linearity of simple Monte Carlo processes.

References

[1] W. Apel, J. Arteaga-Velázquez, K. Bekk, M. Bertaina, J. Blümer, H. Bozdog et al., *KASCADE-Grande measurements of energy spectra for elemental groups of cosmic rays*, Astroparticle Physics **47** (2013) 54.

[2] T.I. Collaboration, Fermi-LAT, MAGIC, AGILE, ASAS-SN, HAWC et al., *Multimessenger observations of a flaring blazar coincident with high-energy neutrino icecube-170922a*,

Science **361** (2018) eaat1378
[https://www.science.org/doi/pdf/10.1126/science.aat1378].

[3] J. Abraham, P. Abreu, M. Aglietta, C. Aguirre, D. Allard, I. Allekotte et al., *Correlation of the highest-energy cosmic rays with nearby extragalactic objects*, *Science* **318** (2007) 938 [https://www.science.org/doi/pdf/10.1126/science.1151124].

[4] Jannik Augscheller, "Runtimes with different energies-corsika 8 general meeting." URL: https://indico.scc.kit.edu/event/2748/contributions/10553/attachments/5096/7746/CPU-Runtime_EM-showers.pptx.

[5] M. Kobal, *A thinning method using weight limitation for air-shower simulations*, *Astroparticle Physics* **15** (2001) 259.

[6] J. Alvarez-Muñiz, R. Engel, T.K. Gaisser, J.A. Ortiz and T. Stanev, *Hybrid simulations of extensive air showers*, *Phys. Rev. D* **66** (2002) 033011.

[7] T. Bergmann, R. Engel, D. Heck, N. Kalmykov, S. Ostapchenko, T. Pierog et al., *One-dimensional hybrid approach to extensive air shower simulation*, *Astroparticle Physics* **26** (2007) 420.

[8] G. Bossard, H.J. Drescher, N.N. Kalmykov, S. Ostapchenko, A.I. Pavlov, T. Pierog et al., *Cosmic ray air shower characteristics in the framework of the parton-based gribov-regge model nexus*, *Phys. Rev. D* **63** (2001) 054030.

[9] I. Sutskever, *Training Recurrent Neural Networks*, Ph.D. thesis, University of Toronto, CAN, 2013.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan et al., *Pytorch: An imperative style, high-performance deep learning library*, in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds., pp. 8024–8035, Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[11] A.L. Maas, A.Y. Hannun, A.Y. Ng et al., *Rectifier nonlinearities improve neural network acoustic models*, in *Proc. icml*, vol. 30, p. 3, Atlanta, GA, 2013.

[12] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds., 2015, http://arxiv.org/abs/1412.6980.