PROCEEDINGS OF SCIENCE

Modeling images of proton events for the TAIGA project using a generative adversarial network: features of the network architecture and the learning process

Julia Dubenskaya[*], **Alexander Kryukov, Andrey Demichev**

Lomonosov Moscow State University, SINP,
Moscow 119991, Russia

E-mail: jdubenskaya@gmail.com

High-energy particles interacting with the Earth atmosphere give rise to extensive air showers emitting Cherenkov light. This light can be detected on the ground by imaging atmospheric Cherenkov telescopes (IACTs). One of the main problems solved during primary processing of experimental data is the separation of signal events (gamma quanta) against the hadronic background, the bulk of which is made up of proton events. To ensure correct gamma event/proton event separation under real conditions, a large amount of experimental data, including model data, is required. Thus, although proton events are considered as background, their images are also necessary for accurate registration of gamma quanta. We applied a machine learning method, namely the generative adversarial networks (GANs) to generate images of proton events for the TAIGA project. This approach allowed us to significantly increase the speed of image generation. At the same time testing the results using third-party software showed that over 95% of the generated images are correct and can be used in the experiment. In this article we provide a detailed GAN architecture suitable for generating images of proton events similar to those obtained from IACTs of the TAIGA project. The features of the training process are also discussed, including the number of learning epochs and selecting appropriate network parameters.

*The 5th International Workshop on Deep Learning in Computational Physics (*DLCP2021*)*
28-29 June, 2021
Moscow, Russia

[*]Speaker

PoS(DLCP2021)011

1. Introduction

High-energy particles interacting with the Earth atmosphere give rise to extensive air showers (EAS) emitting Cherenkov light. This light can be detected on the ground by imaging atmospheric Cherenkov telescopes (IACTs) [1]. During operation, each IACT records a set of images of the air shower against the background light of the night sky.

The TAIGA Cherenkov telescope array [2] is a ground-based observatory for gamma-ray astronomy that records air showers produced by charged cosmic rays or high energy gamma rays. The bulk of the registered events are proton events, their number exceeds the number of registered gamma events by several orders of magnitude. At the same time, the events of interest are gamma events. Therefore, one of the main problems solved during primary processing of experimental data is the separation of gamma events against the hadronic background. To ensure correct separation, simulation data containing both types of events are required. Thus, although proton events are considered the background, their images are also necessary for accurate identification of the registered gamma quanta.

The standard method for obtaining images similar to those recorded by the TAIGA Cherenkov telescope array is direct modeling of physical processes occurring in an EAS. This method gives very accurate results, but requires significant resources and time. For some applications, this accuracy is redundant, so we can apply machine learning methods such as generative adversarial networks (GANs) [3] that are much lighter and faster. In our previous work [4], we generally described the use of GANs to quickly generate images similar to those recorded by the TAIGA Cherenkov telescope array. In this work, we will focus on the features of the generation of proton events using a GAN.

2. Generative adversarial network for proton event images

From a technical point of view a GAN is a system of two neural networks that are trained simultaneously in an adversarial game: a generative network (generator) that captures the data distribution, and a discriminative network (discriminator) that estimates the probability that a sample came from the training data rather than generator. The training procedure for generator is to maximize the probability of discriminator making a mistake. The system as a whole corresponds to a minimax two-player game.

Recently, GANs are increasingly used to generate, improve or process images. When creating a network, one needs to develop a general architecture (choose the number of layers, filter size, activation functions, etc.), as well as rules for preparing a training set, and then perform training. After that, the generator can be used to create images. It is important that although these images are similar to the training set, they are completely new ones.

The following is a description of the features of the network for generating images of protons for the TAIGA project.

2.1 Network architecture

Despite the fact that selection of neural network parameters is not formalized and is usually done intuitively, there are more or less standard recommendations [5, 6] for creating neural networks for generating images of numbers, letters, clothes, etc.

The architecture of our generator is more or less standard. The generator takes as input a point in the latent space – a random vector of 8192 (128x8x8) entries, and outputs a single 32x32 grayscale image. This is achieved by using a fully connected layer to interpret the point in the latent space and provide sufficient activations that can be reshaped into many copies (in our case 128) of a low-resolution version of the output image (e.g. 8x8). Then the generator performs image resizing procedure thus increasing the size of its output relative to the input. The peculiarity of our generator is that the image resizing is performed only at the first and second layers, while the third and fourth layers keep the image size unchanged. It should be noted that at the first layer, upsampling is used to resize the image, and at the second layer, inverse convolution with a 2×2 strides is applied. This combination, despite looking odd, ensures stable network operation and provides the best quality generated images without artifacts (such as a checkerboard-like pattern [7]).

Thus, our generator has 4 layers of convolution. All layers except the output layer use 4x4 filters and a leaky ReLU function with alpha=0.2 as the activation function. The output layer has one 6x6 filter and uses a sigmoid for its activation. It is worth noting that although the best practices for GAN design recommend using the hyperbolic tangent as the activation function of the output layer of the generator, in our case the sigmoid provides better results.

We also apply batch normalization (BN) [8] in the generator. The main advantage of this technique is that it greatly speeds up the learning process. In our case, BN makes the generator and, as a result, the entire GAN more stable. Without using this technique, our generator created blurry images and poorly reproduced the black background of the proton images. We adopted BN between convolutional layers before each activation function.

The architecture of the generator is shown on Figure 1.

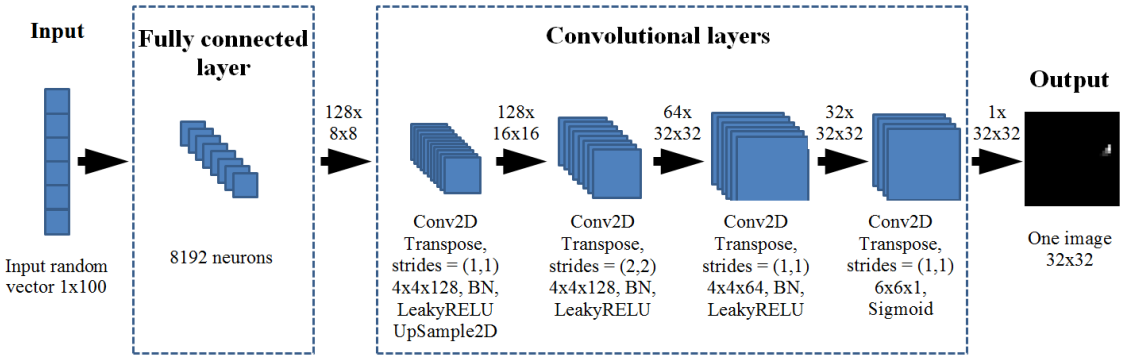

Figure 1. Architecture of the generator

A discriminator is a classic convolutional neural network that performs image classification. The architecture of our discriminator differs from the standard ones. Unlike most examples in which large enough images are generated, the images of proton events are small. That is, usually we have only a few light pixels on a black background. We found that having a standard architecture with several convolutional layers, our discriminator becomes unstable and

from time to time the GAN generates a completely black square. This is an extreme case of one of the GAN failures - a mode collapse [9], when generator is only capable of generating one or a small subset of different outcomes. To avoid this failure we had to reduce the number of convolution layers to two and add two fully connected layers.

Our discriminator takes as input one 32x32 grayscale image and outputs a binary prediction as to whether the image is real or fake. It is implemented using best practices for GAN design such as using a 2x2 stride to downsample, and the Adam version of stochastic gradient descent with a learning rate of 0.0002 and a momentum of 0.5. In the convolutional layers, the convolution filter size is 4x4; a leaky ReLU function with alpha=0.2 is used for the activation. In the fully connected layers the activation function is a leaky ReLU function with alpha=0.2, and the dropout with a probability of 0.4 is also used. The output layer uses a sigmoid function for its activation.

The architecture of the discriminator is shown on Figure 2.

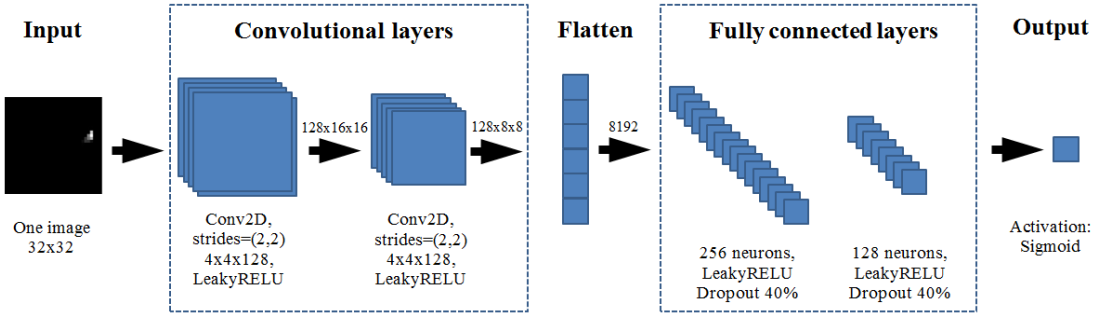

Figure 2. Architecture of the discriminator

2.2 Training set preparation

For training we used a sample of two-dimensional images obtained using TAIGA Monte Carlo simulation software [10, 11], containing about twenty five thousand proton events. This software performs direct simulation of series of interactions of the primary cosmic ray and its progeny with the atmospheric nuclei. Thus, we train the network on the model images. The images from our training set are cleaned, i.e. they do not contain noise from fluctuations in the background of the night sky and electronic components. When preparing the training sample images, we applied coordinate transformation, image resizing and pixel values recalculation.

The original images recorded by the TAIGA telescope cameras have a hexagonal shape, as the photomultipliers that directly record signals are arranged in a hexagonal grid. We generate square images from hexagonal ones by switching to an oblique coordinate system and adding zeroes to the right and bottom, since current high-performance neural network frameworks are implemented for square grids. As a result, we get images of thirty-two by thirty-two pixels.

Our training set contains images with different pixel values (in photoelectrons), so to reduce this difference, we had to switch to a logarithmic scale by applying the logarithm function to each pixel value of each image: ln(1+x).

Finally, the pixel values must be scaled to the range [0, 1] to match the output of the generator model. Unlike standard images, the pixel value of which lies in a fixed range (for example, from 0 to 256), the pixel value in photoelectrons is theoretically unlimited from above. Therefore, when moving to the range [0, 1], we calculate the maximum pixel value for our

training set, which we keep in mind and use later to reverse convert the generated images. This is how we get a square grayscale image which we feed to the discriminator input. An example of an original image and the corresponding image after preprocessing is shown in Figure 3.

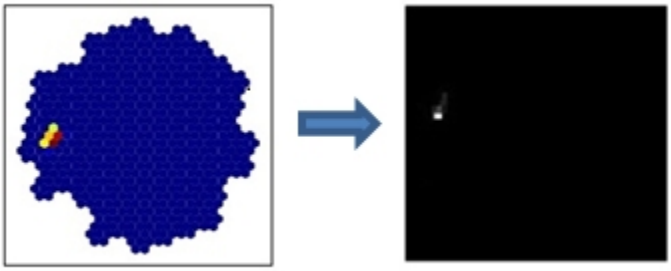

Figure 3. An example of converting an input image

2.3 Training details

In addition to the architecture and parameters of the network itself, two more hyperparameters of the learning process should be mentioned that are important when training a GAN: a batch size and a number of epochs.

The batch size is a number of training images to work through before updating the weights of the network. The batch size influences the dynamics of the learning algorithm and can be used to control the stability of training. For generation of proton images we tried batch sizes of 32, 64, 128 and 256 images. We can confidently say that the batch size of 32 images is too small for our task and for our data, which makes the learning process unstable and greatly increases the probability of a mode collapse (up to values of more than 50%). The batch sizes of 64 and 256 images are acceptable, but the most stable training results were achieved with the batch size of 128.

The number of epochs controls the number of complete passes through the training dataset. Depending on the task, the recommended number of epochs varies from 10 to 1000 and more.

The feature of our training dataset is that the events in it are not uniformly distributed over energies (energy and pixel values are related: the more energy, the greater the maximum pixel value and vice versa). So, most of the events have low energy. There are few events with very low energies, as well as with high energies. Events with very high energies are extremely rare. We found that in our case, when the number of learning epochs is less than 50, the network completely loses information about rare and very rare events.

Experimentally, we found that when the number of epochs is about three hundreds, the network becomes capable of generating rare events (but not extremely rare ones). A further increase in the number of epochs does not expand the range of generated energies. For super rare events, we will either have to create a separate network or modify the training set in some way. A detailed description of the statistical differences between the generated sample and the training set is described in Section 3.

3. Implementation and results

We implemented our network using the Keras high-level API of the TensorFlow [12] software package. A server with a Tesla P100 GPU was used to train our network. With a training set of 25000 events, with batch normalization, a batch size of 128 images and 300 epochs, the overall training took 6 hours. After training, the network is able to generate 400 proton images in 1 second, or 1,440,000 images per hour. The standard TAIGA Monte Carlo simulation software generates an average of 1000 images per hour. Thus, our generative network creates images almost 1500 times faster.

The output of the neural network is a set of square grayscale images, with pixel values ranging from 0 to 1. To get each image in its original format, the inverse transformations are required. First, we reverse-scale the images from the range [0, 1] using the previously stored maximum pixel value of the training set. We then drop the logarithmic scale by applying an exponential function (exp(x) - 1) to each pixel value of each image. Then we discard the extra pixels on the right and bottom of each image (if the network is well trained, these pixels are always zero). Finally, the generated images are converted back to a hexagonal form.

An example of a generated image and its back conversion is shown in Figure 4.

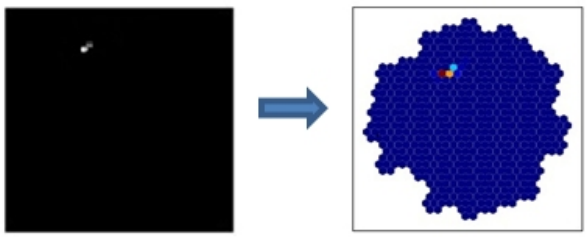

Figure 4. A generated image and its back conversion

We checked the quality of the generated images in our previous work [3]. This check showed that over 95% of the generated proton images differ with confidence from the gamma event images, with less than 1% being misinterpreted as the gamma event images. In this paper, we further investigate the brightness distribution of the generated images compared to the brightness of the training set images. By the brightness of the image, we mean the sum of the values of the photoelectrons of all the pixels in the image. For this test, we generated a sample of 10,000 images of proton events and also selected 10,000 images from the training set. For each image from each sample, we calculated the brightness, and then determined the distribution of the number of events by brightness. The distributions for the training set (shown in red) and for the generated sample (shown in blue) are plotted in Figure 5.

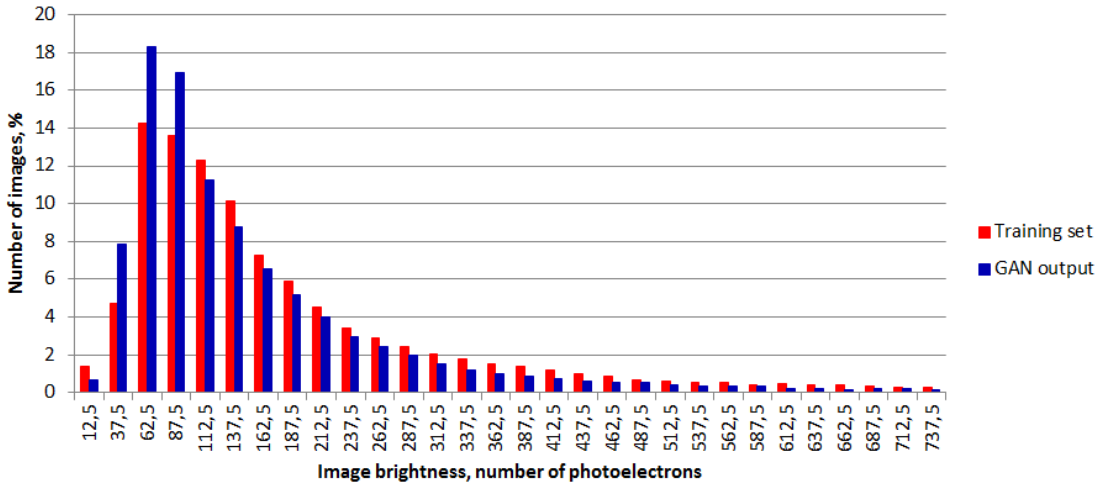

Figure 5. Comparison of distributions for the training set and for the generated sample

As we can see from the plot, the shape of the distributions is generally similar. However, the frequency of occurrence of rare events (with very low or very high brightness) is less for the GAN output. At the same time, this plot confirms that we managed to avoid the mode collapse, and the generator successfully creates images with different brightness.

4. Conclusions

Summarizing the above, we can conclude that generative adversarial networks simulate proton events for the TAIGA project with a high degree of accuracy and reliability. Most of the generated events are statistically indistinguishable from the events of the training set. At the same time, the rate of generation of the events using GANs is much higher than the rate of generation using the standard Monte Carlo method. Notice that our GAN provides a brightness distribution similar in shape to the original distribution without any extra efforts. In our future work, we hope to achieve an even better fit of these distributions.

5. Acknowledgements

This work was carried out in the framework of R&D State Assignment No.115041410196.

References

[1] T.C. Weekes et al., *Observation of TeV gamma rays from the Crab Nebula using the atmospheric Cerenkov imaging technique*, Astroph. J. **342** (1989) 379

[2] N. Budnev et al., *The TAIGA experiment: From cosmic-ray to gamma-ray astronomy in the Tunka valley*, Nucl.Instrum.Meth. **A845** (2017) 330-333

[3] I.J. Goodfellow et al., *Generative Adversarial Networks*, ArXiv **1406.2661** (2014)

[4] J. Dubenskaya et al., *Fast Simulation of Gamma/Proton Event Images for the TAIGA-IACT Experiment using Generative Adversarial Networks*, in proceedings of *the 37th International Cosmic Ray Conference* PoS(ICRC2021)874

[5] J. Langr, V. Bok, *GANs in Action: Deep learning with Generative Adversarial Networks*, Manning, New York, 2019

[6] J. Brownlee, *How to Develop a GAN for Generating MNIST Handwritten Digits*, https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras. Last accessed 01 December 2021

[7] A. Odena et al., *Deconvolution and Checkerboard Artifacts*, Distill, 2016

[8] S. Ioffe et al., *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, in proceedings of *the 32nd Int. Conf. on Machine Learning*, **ICML'15 v.37** (2015) 448–456

[9] H. Thanh-Tung et al., *Catastrophic forgetting and mode collapse in GANs*, in proceedings of *the Int. Joint Conf. on Neural Networks*, **IJCNN'2020** (2020) 1-10

[10] D. Heck et al., *CORSIKA: A Monte Carlo code to simulate extensive air showers*, Forschungszentrum Karlsruhe **FZKA 6019** (1998)

[11] E.B. Postnikov et al., *Hybrid method for identifying mass groups of primary cosmic rays in the joint operation of IACTs and wide angle Cherenkov timing arrays*. J.Phys.: Conf. Series **798**, 012030 (2017)

[12] TensorFlow Homepage, http://www.tensorflow.org. Last accessed 01 December 2021