Int. J. Appl. Math. Comput. Sci., , Vol. , No. , –
DOI:

GENERATIVE MODELS FOR FAST CLUSTER SIMULATIONS IN THE TPC FOR THE ALICE EXPERIMENT

KAMIL DEJA [a], TOMASZ TRZCIŃSKI [a,*], ŁUKASZ GRACZYKOWSKI [b], FOR THE ALICE COLLABORATION

[a]Institute of Computer Science
Warsaw University of Technology, Pl. Politechniki 1 00-661 Warsaw
e-mail: kdeja@mion.elka.pw.edu.pl,t.trzcinski@ii.pw.edu.pl

[b]Faculty of Physics
Warsaw University of Technology, Pl. Politechniki 1 00-661 Warsaw
e-mail: lukasz.graczykowski@pw.edu.pl

Simulating the possible detector response is a key component of every high-energy physics experiment. The methods used currently for this purpose provide high-fidelity results. However, this precision comes at a price of a high computational cost. In this work, we present a novel solution for generating the possible responses of detector clusters to particle collisions, using the real-life example of the Time Projection Chamber (TPC) in the ALICE experiment at CERN. Its essential component is a generative model that allows to simulate synthetic data points that bear high similarity to the real data. Leveraging recent advancements in machine learning, we propose to use state-of-the-art generative models, namely Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). In this work we implement two separate approaches: one to simulate random data samples possible to record in the detector and one which conditions generated examples on the initial information about particles. For both ideas we propose and evaluate several models based on convolutional or recursive networks. The main advantage offered by the proposed methods is a significant speed-up in the execution time, reaching up to the factor of 10^3 with respect to the currently used simulation tool. Nevertheless, this computational speed-up comes at a price of a lower simulation quality. In this work we adapt currently available methods and show their quantitative and qualitative limitations.

Keywords: Generative Adversarial Networks, Variational Autoencoder, Fast Simulation, LHC, ALICE, TPC

1. Introduction

High-energy physics (HEP) experiments, including those conducted at the Large Hadron Collider(LHC) (Evans and Bryant, 2008), rely heavily on detailed simulations of the detector response in order to accurately compare recorded data with theoretical Monte Carlo models. This simulated data is used to understand the physics behind the real collisions registered in the experiment. The simulations are also used to adjust the experimental data for detector's inefficiencies and various conditions observed in the detector during the data taking procedure.

Traditional approaches to the simulation of the detector response use Monte Carlo methods to generate collisions. To properly simulate events, those methods need to synthesise propagation of every simulated particle through the detectors medium and model every interaction with the experimental apparatus using a transport package, usually GEANT3 (Brun *et al.*, 1994), GEANT4 (Agostinelli et al., 2003) or FLUKA (Ferrari *et al.*, 2005). This approach comes with disadvantages. First of all, those packages require specific implementations of the interactions of highly energetic particles with matter that occur inside the detector. In practice, the physics regulating those interactions is complex and often difficult to simulate. Secondly, the simulation methodology that relies on Monte Carlo methods is computationally expensive and scales linearly with the amount of simulated collisions.

To address the above shortcomings of a Monte Carlo-based simulation, we propose a fundamentally

*Corresponding author

different approach to simulations in high-energy physics experiments that relies on generative models. The proof-of-concept solution, that implements this approach is based on the recently proposed models, such as Variational Autoencoders (Kingma and Welling, 2013) and Generative Adversarial Networks (Goodfellow *et al.*, 2014). The main idea behind using those models for simulations is to learn how to synthesise the situation of the detector after a collision from the data, instead of modelling every interaction of the particles. Such approach enables much faster and easily parallelizable generation.

This work is the extension of the previous article published as a conference proceedings (Deja *et al.*, 2018). In (Deja *et al.*, 2018), we concentrated on the usage of generative models for simulating the situation possible to register in the detector, in this work we extend this idea by focusing on improving current simulation tool. Therefore we implement a solution leveraging from the part of the recent design based on conditional Generative Adversarial Networks (Mirza and Osindero, 2014). Such an approach extends the standard idea of modelling images similar to real examples by applying additional constrain on some initial parameters, in our case values describing simulated particle.

As a first step towards implementing our solution in production, we perform a set of experiments that aim at simulating clusters of the Time Projection Chamber (TPC) (Dellacasa et al., 2000; Abelev et al., 2014) detector of the ALICE (A Large Ion Collider Experiment) (Aamodt et al., 2008) experiment at the LHC. We present quantitative, qualitative and computational cost evaluation of our method for several generative models and their variations, including standard Variational Autoencoder, Deep Convolutional Generative Adversarial Networks (Radford *et al.*, 2015) and Long Short Term Memory (Hochreiter and Schmidhuber, 1997) based networks. We also show how to improve the results obtained from those models using a recently proposed training method called a Progressive GAN (Karras *et al.*, 2017) as well as introduced in this work combined loss function leveraging from both autoencoder and adversarial approaches.

To our knowledge, our works represent the first attempt to use machine-learning generative models for clusters simulation in any high-energy physics experiment currently operating at CERN. The main contribution of this paper is a prototype of a proof-of-concept method for event simulation in the TPC detector that relies on generative models. We provide here an extensive overview of the recently proposed generative models that can be used in this context, as well as a detailed evaluation of their current performances when applied to cluster simulation. Evaluation results show a promising speedup over currently used Monte Carlo based simulation methods reaching up to 10^3, however, at the expense of simulation quality. We address this problem in the future work and show how to improve the generative models in the context of cluster simulation.

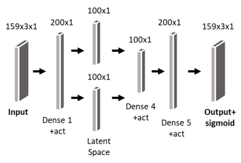
The remainder of this paper is organised as follows. In the next section, we describe related works. We then overview generative models that can be used in the proposed solutions and present the core idea behind its adaptation. Finally, we describe the dataset and present the results of our evaluation. In the last section, we conclude this work and outline the next steps.

2. Related Work

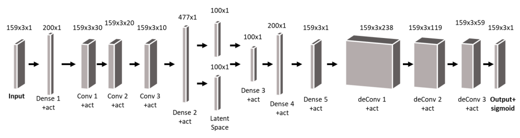
Currently used methods start from a collision simulation using a Monte Carlo generator, than different transport packages such as GEANT3 (Brun *et al.*, 1994), GEANT4 (Agostinelli et al., 2003) or FLUKA (Ferrari *et al.*, 2005) are used to synthesise propagation of particles. At first, they provide space points where a given energetic particle has deposited some or its energy, as well as simulate the secondary particle showers in the detectors which leave their signal. Those space points have to be further translated to a simulated electronic signal from the detector sensors, as it would appear in a real collision. In the tracking detectors, those electronic signals, called *digits*, are then used to calculate *clusters*, a set of space-time digits from the adjacent sensitive elements of the detector that were presumably generated by the same particle crossing these elements.

Machine learning generative models, which we intend to use in the context of high-energy physics, have already proven their potential in numerous research areas related to text-to-image generation (Reed *et al.*, 2016), conditional image generation for criminal identification purposes (Yan *et al.*, 2016) or even drug design (Kadurin *et al.*, 2017). Their main research directions, however, are still restricted to applications related to images and videos. The task we aim to address in this paper, differs significantly from those applications. Simulating precise responses of a detector require high-fidelity of the obtained results, measured both qualitatively and quantitatively. This is not the case for artificially generated images or videos where subjective assessments are often the only way to evaluate generated output.

Despite those problems, there are several attempts to use Generative Adversarial Networks for simulating parts of the physics processes, like the very first one in the field: calorimeters response simulation (Paganini *et al.*, 2017). In that work, the authors generate particle showers in electromagnetic calorimeters. Following the LAGAN processing (de Oliveira *et al.*, 2017), they adapt a DCGAN architecture to generate a number of 2D images, which they merge, to produce a final 3D Calorimeter response. Although our work draws inspiration from (Paganini



(a) VAE



(b) cVAE

Fig. 1. Variational Autoencoders architectures evaluated in the context of detector's response simulation. Each block represent a network layer with it's size above.

et al., 2017), we tackle a different problem related to detector's response simulation, while (Paganini *et al.*, 2017) focuses on generating particle showers.

3. Generative models

In this section we overview two generative machine learning models that we use in our solution and their architectures. First, we describe a Variational Autoencoder (VAE), and propose two different models of a VAE that fit our task. We use them as an evaluation baseline for the second method which is based on the Generative Adversarial Network (GAN) algorithm. We present those methods in the following sections. We also outline an adaptation of conditional Generative Adversarial Networks and recently proposed progressive GAN training algorithm (Karras *et al.*, 2017) for cluster simulation.

3.1. Variational Autoencoder. Variational Autoencoder is a machine learning method that relies on an autoencoder idea proposed in (Hinton and Salakhutdinov, 2006). It is implemented through an artificial neural network that can be used for dimensionality reduction and data compression. The simplest architecture of an autoencoder consists of an input layer with N neurons, where N is a dimensionality of the data, a hidden layer with $K < N$ neurons and the output layer with N neurons, exactly the same size as the input. The goal of an autoencoder is to re-generate the same output as received on the input, while internally reducing the latent representation to a lower number of dimensions.

The Variational Autoencoder (VAE) (Kingma and Welling, 2013) extends this idea by focusing on the generalisation part of the encoder architecture.VAE re-generates new samples straight from the latent space by modifying the distribution of noise populated in the hidden layer. To ensure that it generates meaningful results, additional normalization constraint is added to the training loss function. It forces a neural network to ensure a normal distribution of the values retrieved as the output of the hidden layer neurons.

The actual VAE loss function with reparametrisation trick is described in Eq. 1:

$$\begin{aligned}\mathcal{L}(\phi,\theta,x) =&\mathbb{E}_{z\sim q_\phi(z|x)}[\log(p_\theta(x|z))]-\\ &D_{KL}(q_\phi(z|x)||p_\theta(z))\end{aligned} \tag{1}$$

where ϕ and θ are the NN parameters for accordingly encoder and decoder part, q_ϕ is then the distribution of encoded values, and the p_θ the distribution of noise from which we sample. D_{KL} is a Kullback-Leibler divergence defined as relative entropy between two distributions.

In this work, we focus on two architectures of Variational Autoencoders that can be applied in the context of detector's response simulation. First of them, dubbed simply VAE, is a fully connected network with one hidden layer. Second one, which we call cVAE, is a convolutional model with three convolutional and deconvolutional layers in encoder and decoder (generator) layers. Fig. 1 shows both architectures.

3.2. Generative Adversarial Networks. A second family of generative machine learning methods discussed in this work is based on the Generative Adversarial Network (Goodfellow *et al.*, 2014). The main concept behind this unsupervised generative model is to train two neural networks to play a *min-max game* between each other.

The first one – a *generator* – is trained to generate new data points whose distribution resembles the distribution of real data. The second one – trained as a *discriminator* – aims at predicting whether a given example comes from a real or synthetic distribution. During the training process, both networks are updated iteratively, one at a time, which leads to a simultaneous optimization of loss functions of both generator and discriminator.

The *min-max* loss function introduced in (Goodfellow *et al.*, 2014), encompassing both networks can be written in the following manner presented in the Eq. 2:

$$\begin{aligned}\underset{G}{min}\,\underset{D}{max}\,V(D,G) =&\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]+\\ &\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]\end{aligned} \tag{2}$$

where $p_{data}(x)$ is the distribution of real data, x is a sample from p_{data}, $p(z)$ is a distribution of a noise vector z input into a generator and G and D represent a discriminator.

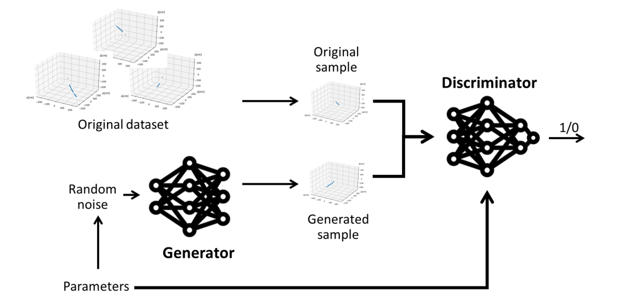
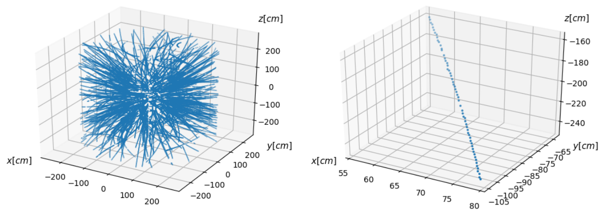


Fig. 2. Conditional Generative Adversarial Network, is the extension to standard approach with additional parameters applied to both *generator* and *discriminator*



(a) (b)

Fig. 3. Input data examples: **(a)** a full simulation of space points occurring after particles collision in the TPC detector of the ALICE Experiment; **(b)** clusters generated by a single particle.

3.3. Conditional Generative Adversarial Networks. Sometimes, as for the task of particle clusters generation, the data samples we want to simulate depend on some initial parameters. In such cases, the wise idea would be to condition the output images distribution for Generative Adversarial Network on the initial parameters. This was proposed as the Conditional Generative Adversarial Network (Mirza and Osindero, 2014).

In this approach authors incorporate additional parameters to the standard GAN architecture. This is achieved in two steps, presented in Fig. 2. Firstly, the noise applied as a generator input is generated with the respect of those parameters. Secondly, the same values are applied to the discriminator so it can distinguish not only if the generated sample is similar to the real data, but also if it fits the provided parameters.

4. Dataset

To evaluate the performance of generative models, we use a dataset of 3D trajectories of particles after collision generated with a Monte Carlo simulation method of PYTHIA 6.4 (Sjostrand *et al.*, 2006) Perugia-0 (Skands, 2010) model. The dataset therefore contains a sample of proton–proton collisions at the centre of mass energy of $\sqrt{s} = 7$ TeV.

After the simulated collision, the generated particles were transported, using GEANT3 (Brun *et al.*, 1994) package, through the detector's medium so that the full simulation of the detector response as well as the track reconstruction were performed. The trajectories correspond to the real traces observed in the TPC detector of the ALICE Experiment at LHC, CERN and the experimental conditions corresponded to 2010 data-taking period of the experiment. Fig. 3 shows sample visualisation of the particle trajectories from the collected dataset.

Since the size of the dataset generated this way results in petabytes of data, we randomly sample it to contain around 300 events corresponding to approximately 1.000.000 data samples. Though, from further experiments we excluded all of the particles which were not produced exactly during the collision – all products of further decays and all not charged particles. We also removed those which produced less then 50 clusters, because of their little importance for physical analysis. Finally after this preprocessing we ended up with around 50 000 data examples.

These contain mostly 3 types of particles. The vast majority are **Pions** (π^+ and π^-) – 48523 particles, with the mass of 0.139e. Then, dataset includes 3893 **Kaons** (K^+ and K^-) – centre of mass equals to 0.4936e, and 2015 **Protnons** (P) – 0.938e. A few additional particles are present with marginal number of occurrences – 430 in total.

Each data sample representing particle clusters is stored together with the initial particle's momenta (p_x,p_y,p_z) its charge, and mass, so the initial values generated by the monte carlo simulation. Those input is associated to a series of 3D coordinates (x, y, z) corresponding to the trajectory of a given particle after the collision. The minimal and maximum values of the coordinates depend on the detector size, which is a cylinder-shaped structure of approximately $5m \times 5m \times 5m$. Since the collision does not necessarily occurs in the central point of the detector, the minimal distance from the $[0, 0, 0]$ point, where the trajectories are collected, is 848 mm. The precision of the recorded coordinates is limited to the resolution of the TPC read-out pads, which is between 0.8 to 1.2 mm, depending on the size of the pad (Dellacasa et al., 2000). For normalisation purposes, the input data coordinates are scaled to fit the $[0, 1]$ interval in each dimension. We also apply other normalisation procedures, such as zero-padding the samples for particle trajectories consisting less than 159 points (maximum number which can be registered by readout pads). Although additional characteristics of the particles can be observed, *e.g.* its energy and speed, for the purpose of the evaluation presented in this work we restrict our data samples to 3D coordinate values. Therefore we work on the data with final resolution of

$\mathbf{159 \times 3}$ pixels. To produce meaningful output we input to the generative models random noise of **100** values.

As shown in Fig. 3(a), after the collision occurs, up to few hundreds particle trajectories can be observed inside the detector. Although we could attempt to simulate all of those trajectories at once using generative models, we postulate to generate separate trajectories for individual particles first and then merge them together to achieve the final goal. This approach implies better control of the studies and evaluate the results iteratively, as more trajectories are added. Furthermore, the highest reported resolution of the output generated by the state-of-the-art generative models, such as VAEs and GANs, is 1024×1024 (Karras *et al.*, 2017). This resolution is relatively low, when compared to the one observed in the TPC detector – $5000 \times 5000 \times 5000$ possible clusters locations. To circumvent this limitation of the generative models, we transform our simulation problem into a so-called transactional form and simulate individual points with their (x, y, z) coordinates, which we can then link together to form a full trajectory.

5. Architectures

The general framework of Generative Adversarial Networks is rather flexible in terms of the type and number of layers used as a generator and discriminator. Hence, we propose several architectures that fit to the context of simulating detector's response in a high-energy physics experiment, with respect to the dataset described above.

More precisely for the first approach of simulating possible detector output, we evaluate the following models:

- **Multi-layered Perceptron (MLP)**: a baseline Generative Adversarial Network that consists of 4 fully connected layers used as a discriminator network with a generator built exactly in the reverse symmetrical way,

- **Recurrent GAN based on Long-Short Term Memory (LSTM)**: a recurrent network that comprises LSTM units proposed in (Hochreiter and Schmidhuber, 1997),

- **Deep Convolutional Generative Adversarial Network (DCGAN)**: inspired by (Reed *et al.*, 2016) multi layered network, with two dimensional convolutional/de-convolutional layers, as shown in Fig. 4.

5.1. Progressive GAN. A typical approach to increase the performance of neural networks is to add more layers to the model, hence increasing the depth of a resulting neural network. This, however, comes at a price of more complex training procedure. To evaluate how much performance gain we can obtain by growing our network, we rely on a recently proposed progressive GANs training method introduced in (Karras *et al.*, 2017).

As presented in Fig. 4, the DCGAN model we evaluate is rather shallow, *i.e.* it consists of only 3 convolutional/deconvolutional layers. We therefore employ the progressive training method for our DCGAN and grow our network by gradually increasing the number of stacked layers. This way, we are able to enhance the resolution of the resulting simulation by gradually increasing the number of possible locations of simulated cluster responses. This is of our particular interest, since in the context of high-energy physics the resolution of the simulated cluster's response has to be relatively high. Our DCGAN is therefore grown in 5 progressive steps, ending with 6 convolutional/deconvolutional layers. Fig. 5 shows the final architecture obtained by the training method.

5.2. Conditional GAN architectures. For the generation of particle clusters from initial particle parameteres, we propose three models based on the conditional Generative Adversarial Networks. Those conditional architectures derive from similar techniques:

- **Conditional Deep Convolutional Generative Adversarial Network (condDCGAN)**: an extension of the DCGAN model with respect to the conditional Generative Adversarial Network approach.

- **Conditional LSTM based Generative Adversarial Network (condLSTMGAN)**: conditional network based on the LSTM network with similar architecture to the *LSTM GAN*.

- **Conditional Progressive DCGAN (condproGAN)**: application of progressive training for the conditional DCGAN model.

5.3. Conditional Generative Adversarial Network with additional loss. To enhance the quality of the samples generated with conditional GAN, as well as to prevent the model from collapsing we propose the new training procedure, which derives from both Generative Adversarial Networks and Variational Autoencoders. The idea is to train the generator to produce the new samples

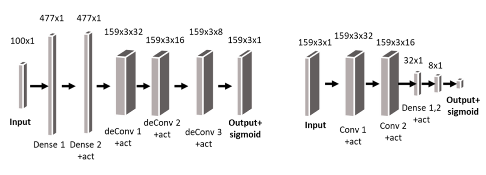


Fig. 4. Architecture of the DCGAN model. Each block represent a network layer with it's size above.

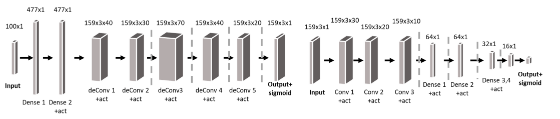


Fig. 5. Architecture for the Progressive GAN model. Dotted lines represent consecutive grow of the network with progressive steps. Each block represent a network layer with it's size above.

only for the parameters selected from the training data. It allows to compare the generated samples with the original ones in the same manner as in autoencoders. However, to ensure the non-deterministic behaviour of the model, we propose to combine the distance measured from original sample with the standard loss produced by the GAN's discriminator. The exact generator's loss function for such an approach is described in 3.

$$\mathcal{L}_G(m,X) = \mathbb{E}_{z\sim p_z(z|m)}[\alpha \log(1 - D(G(z))) + \beta \frac{1}{n}\sum_{i=1}^{n}(X_i - G(\hat{z})_i)^2] \tag{3}$$

where m is the initial parameters(particle momenta), X an original value corresponding to m, $p(z|m)$ is a distribution of a noise vector under initial parameters m, z input into a generator, and G and D represent generator and discriminator. In second part of the loss function n is the number of produced clusters. Additional parameters α and β are used to weight the share of individual losses. Test showed that the best performing values are $\alpha = 0.6$ and $\beta = 0.8$.

We conduct tests on this training procedure using three architectures. In order to distinguish them during the evaluation process, we refer to them using additional pluses at the end of architecture's name:

- condLSTM GAN+
- condDCGAN+
- condproGAN+

In addition, to speed up the training, and to make sure that the discriminator learns to distinguish whether the given output is in line with initial parameters, we conduct additional discriminator training. Every 2000 batches we stop the standard procedure and focus on the discriminator. For a few thousand of samples we train it only on the real data examples. For that purpose, we prepare a special batches containing real examples with their initial parameters and some for which we overwrite those momenta using random values. The goal of the discriminator's training for this part is to predict whether the samples correspond to the given initial parameters.

5.4. Implementation details. To train all networks, we use dropout (Srivastava *et al.*, 2014), Leaky ReLu activation (Glorot *et al.*, 2011), and batch normalization (Ioffe and Szegedy, 2015). As our loss function, we take a binary cross-entropy loss. We optimize our loss function with ADAM, a stochastic optimization method (Kingma and Ba, 2014). We initialize network weights with the glorot algorithm (Glorot and Bengio, 2010). The neural networks are implemented in Python using Keras library (Chollet et al., 2015) with TensorFlow backend (Abadi et al., 2016). Our implementations are based on those enlisted in the large scale study (Lucic *et al.*, 2017).

6. Results

In this section, we evaluate methods based on generative models in terms of both: the quality of the generated samples and their computational cost. In the first section we focus on the basic approach of generating any possible particle clusters directly from the random noise. Then in following sections we present the results of quality and computational cost tests of generating particle clusters conditioned on initial particle data. As the baseline model, we use the Monte Carlo-based methods currently used in the ALICE Experiment to generate the full TPC response simulation of the generated particle showers, propagated through the detectors medium by the GEANT3 package. We use the most recent implementation provided as part of the O2 software (Abelev et al., 2014; Suaide *et al.*, 2014).

6.1. Evaluation for detector response simulation. To evaluate the quality of the trajectories generated by the first method, we refer to the physical property of the particle tracks observed after the collision. When particles move through the detector medium, they form a track which shape can be defined as a helix. We leverage this phenomenon and calculate for each method an evaluation metric that measures the distance of generated trajectory from a theoretically ideal shape of helix, approximated with an arc. We report this metric as a mean squared error (MSE) between the ideal helix shape and the simulated points to average it across all simulated particles. We give the result in the final metric system, re-scaled using detector's dimensions. Although this approach does not take into account the physical properties of the particles resulting in different parameters of the helix created by each particle, it allows us to compare several unsupervised simulation methods that are not conditioned on particle characteristics. The proposed metric validates the quality of the shape generated by the evaluated simulation methods and serves as a approximation of a method performance. More complex and more accurate

Table 1. Quality of the Generative models, and their performance comparing to the GEANT3 based simulation solution.

Method	**MSE (mm)**	**speed-up**
GEANT3 (*current simulation*)	1.2	1
Random (*estimated*)	166.155	N/A
GAN-MLP	55.385	10^4
GAN-LSTM	54.395	10^4
VAE	33.40	10^4
DCGAN	19.19	10^2
cVAE	17.70	10
proGAN	**4.31**	**30**

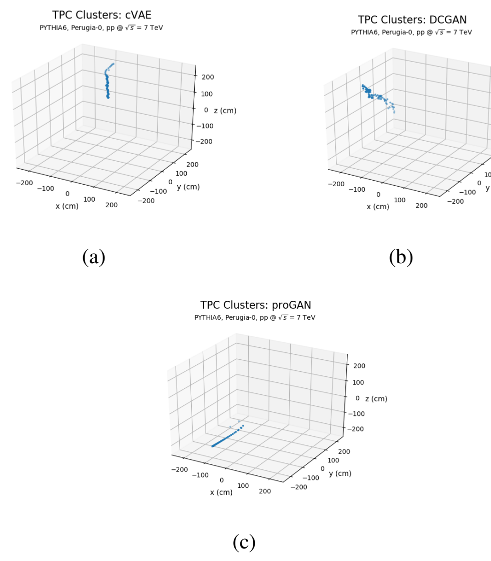


Fig. 6. Exemplar results generated by the **(a)** convolutional Variational Autoencoder (cVAE), **(b)** the Deep Convolutional GAN (DCGAN) and **(c)** the progressive DCGAN (proGAN) .

evaluation procedures are introduced for the conditional approach in the next section.

We also evaluate the computation cost of generating the results for various methods. To that end, we execute the code 10 times on a standalone machine with an Intel Core i7-6850K (3.60GHz) CPU (using single core, no GPU acceleration) and record the average execution time. We then normalise it with respect to the baseline method of simulation, namely GEANT3, which is currently used to simulate the detector's response.

The results of our evaluation are presented in Tab. 1. For the baseline simulation method GEANT3, the average mean squared error (MSE) is equal to 1.2mm. When using a random generator to generate a set of 3D points, the estimated expected MSE is 166mm. Compared to those baselines, the best performing progressive DCGAN model (proGAN) achieves a satisfactory MSE of 4.31mm. This result corresponds to a sub-centimetre accuracy of our method, while providing over 30-fold speed-up with respect to the baseline method of simulation, a Monte Carlo-based GEANT3.

Regarding the other evaluated models, the Variational Autoencoder (VAE) achieves the MSE value of 33.40mm, while providing a speed-up of over four orders of magnitude. In fact, this is the least complex model of all evaluated in this work and therefore both qualitative and quantitative results it achieves are relatively good.

Furthermore, we are able to improve the performance of the VAE by substituting the fully connected layers with the convolutional ones, as it is done in the convolutional Variational Autoencoder (cVAE). With this modification, we can reduce the MSE error to approximately 17.70mm. However, because of a significant increase in model complexity, it is also considerably slower than the VAE. The observed speed-up with respect to the GEANT3 solution reaches only one order of magnitude. As shown in Fig. 6(a), the qualitative results generated by the convolutional VAE appear to be more random, although their resolution is higher than in the case of VAE.

As far as the models based on the Generative Adversarial Networks are concerned, the results obtained by the Multilayer Perceptron (MLP) method indicate that this approach, as well as the recurrent network method based on the LSTM units, do not achieve sufficient accuracy. Although their speed-up over the baseline simulation method reaches four orders of magnitude, the quantitative and qualitative results are only slightly better than those obtained by a random generator.

However, when analysing the results obtained by the Deep Convolutional GAN (DCGAN) model, we can see that extending neural network architectures with convolutional layers significantly reduces the MSE value, while still providing a reasonable speed-up over the baseline that reaches two orders of magnitude. We explain that performance gain by the fact that convolutional layers can better simulate the working conditions of 3D simulation, as well as allow for deeper and more powerful models. The MSE value that is achieved by the evaluated DCGAN architecture is approximately 19.19mm and is comparable to the results produced with VAE and cVAE architectures. The qualitative results can be seen in Fig. 6(b).

Extending the DCGAN model using a progressive training method (proGAN) allows us to improve the quality of the results and reach an unprecedented MSE value 4.31mm. Although the computational complexity of the proGAN model is higher than the competing approaches, it still offers a massive 30-fold speed-up over the baseline GEANT3 simulation method.

6.2. Evaluation for conditional particle trajectories generation. Conditioning the generator's output on the parameters enables to use better means of evaluation. Particle clusters are not fully deterministic, however they are still subjected to several laws of physics. In this section we will evaluate if examples generated with conditional generative models follow those regulations by comparison to the original samples from full simulations.

For proper evaluation of conditional models' quality we randomly split our initial dataset to two separate fragments: training set with 40 000 and test one with remaining 14 872 data samples. Those 14 872 particles intended for testing are: 13163 Pions, 1035 Kaons, 547 Protons and 127 others, what matches the distribution in the training set.

To measure the quality of conditional models, once again we refer to the physical phenomena of particle tracks shape. It is known that for the same environment, theoretical particle track's helix definition depends only on the initial particle parameters. Therefore we approximate the closest arc to the original clusters simulated with monte carlo method and using it as a reference we compute the distance between it and clusters produced with given generative model. We refer to this measure as a Mean Squared Error (MSE) distance from clusters to the helix.

We calculate the MSE on each particle from test dataset. This time however, we refer not only to the mean of the MSEs of all particles but also to their median value. This is because of the nature of this comparison. A single outlier – for example a rare particle flying in the different direction than the predicted one may significantly bias the computed mean.

Table 2 presents the results of this evaluation for different models based on conditional GAN approach. Each model was trained for 200 000 batches of data randomly sampled from trainset which is equal to around 200 epochs on the whole training data.

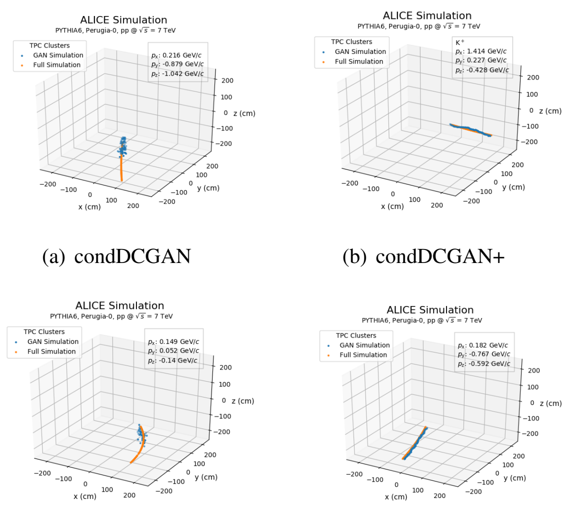
We compare the results obtained with standard conditional GAN approach with those generated with additional loss function (additional + sign by the name). There is no difference in architectures of enhanced and standard models therefore the observed speed-up for generating new samples is equal.

This time we also encountered problems with condLSTM GAN training. Without additional loss, generator could not learn well the distribution of real data. Hence it produced almost random clusters. As listed in table 2, the mean MSE is only slightly lower than for the randomly generated samples. However, application of additional loss in condLSTM GAN+ model resulted in its better convergence and made it possible to properly train the generator. The MSE error was therefore reduced to around 200mm. As presented in Fig. 7(c) generator well resolved the direction in which the particle flew however, the simulated track is fuzzy. This behaviour explains the high MSE value of the model. Nevertheless the condLSTM GAN model is rather shallow, hence the observed speed-up is significant and reaches 2 orders of magnitude.

When analysing the results obtained with convolutional conditional models, the standard condDCGAN converges better then the LSTM based one. In evaluation conducted on the testset we observed the median MSE value of 738.71 millimetres. As visible in

Table 2. Quality of conditional generative models, comparing to the GEANT3 based simulation solution.

Method	Mean MSE (mm)	Median MSE (mm)	speed-up
GEANT3 (*current simulation*)	1.20	1.12	1
Random (*estimated*)	2500	2500	N/A
condLSTM GAN	2093.69	2070.32	10^2
condLSTM GAN+	221.78	190.17	
condDCGAN	795.08	738.71	25
condDCGAN+	136.84	82.72	
condproGAN+	183.52	115.97	20



(a) condDCGAN (b) condDCGAN+

(c) condLSTM+ (d) condproGAN+

Fig. 7. Exemplar results generated by conditional models **(a)** conditional DCGAN model without additional loss, **(b)** conditional DCGAN with additional loss,**(c)** conditional LSTM GAN with additional loss and **(d)** conditional progressive DCGAN with additional loss.

Fig. 7(a) this time there are also evident problems with track's cohesion.

To improve the accuracy of results produced with condDCGAN model we applied additional cost to the generator, which penalised it for creating clusters in some distance from the original ones. This idea proved to be very accurate, since its application in condDCGAN+ model reduced the median MSE value to around 82.72 millimetres. Considering that the detector's resolution reaches around 1 mm, the 8.3cm discrepancy is rather big. However, as presented in Fig. 7(b), for certain type of data the two tracks – generated and original ones are almost identical, even though, the generated clusters are not equal. We debate over this phenomenon later however, it may suggest that the solution proposed in this work can be applied at least for some part of processed data. The main reason for such an application is that condDCGAN is able to generate results around 25 times faster than standard GEANT method already without GPU acceleration.

Finally to smooth out the curvatures of generated particle clusters, we adapted the progressive training method to the conditional GAN approach. The resulting condproGAN+ model is one of the deepest ones, therefore the observed speed-up comparing to the GEANT simulation tool is lower and equal to 20. Also the quality of generated results does not exceed this obtained with the condDCGAN+ model. As listed in Tab. 2, the mean MSE for condproGAN+ equals to 183.52mm which is around 5cm more than for the condDCGAN+. However, as presented in Fig. 7(d) the generated tracks are indeed smoother then for the standard approach.

Using the best method – **condDCGAN+** model, we carried out additional tests to investigate how our solution performs in terms of different additional measures.

Analysing the results. we discovered that the greater the particle initial momentum is, the higher is the accuracy of cluster's position generated with generative model. It is especially visible when concerning momenta perpendicular to the detectors axis dubbed *pT*. The results of the analysis are presented in Fig. 8. The greater the momentum, the straighter the particle track, hence probably generative models evaluated in this work tend to produce straighter tracks with not sufficient curvatures for particles with lower momenta.

Initial momenta and thus track curvature, depends on particle mass. This is why we also observed the lower MSE value for particles with greater mass eg. protons. Table 3 presents the mean MSE and median MSE comparison for different particles generated with condDCGAN+ model. Enlisted results supports the thesis that evaluated models could not well match the curvatures of generated tracks.

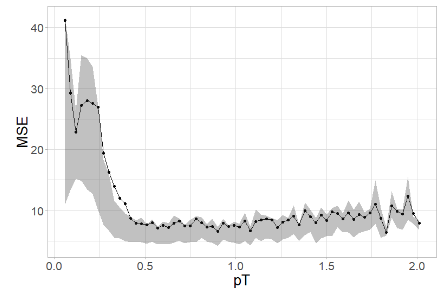


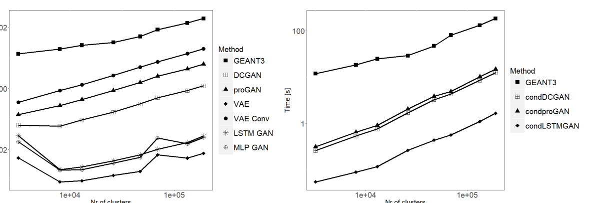
Fig. 8. Mean MSE per initial particle momenta in perpendicular direction pT. Grey area represents the first and third quartile for particles with given initial momenta.

Table 3. Quality of the Generative models, with respect to the type and mass of initial particle

Particle	Mass (GeV/c^2)	Mean MSE (mm)	Median MSE (mm)
Electron	0.0005	231.57	171.46
Muon	0.1056	188.58	125.42
Pion	0.1395	141.29	82.72
Kaon	0.4936	93.96	74.47
Proton	0.9383	98.23	77.37

6.3. Proposed solution's limitations. Despite yet not perfect quality of generative models presented in this work, we understand other limitations which come with our method. Fist of all it is not possible to simulate the whole event at the same time. Generating each particle track independently results in certain omission of dependencies between them. Although the probability of the interaction between two particles in the detector is negligible, it is common for certain types of them to decay and result in two new ones starting their tracks in the same spot. For the proof of concept solution presented in this work we excluded such a particles from our dataset. The second biggest problem not yet solved in the presented solution is constant change in the detector's environment. When standard approach can be easily conditioned on the new detector's parameters, to do so on generative models they should be trained taking into consideration wide spectrum of examples for different possible combinations.

6.4. Summary of computational costs evaluation. Despite the precision of our methods, to emphasise their true potential we present the execution times for different methods both conditional and not, while increasing the number of simulated clusters, i.e. model outputs. Fig. 9 shows the results of this experiment for the first approach. Although the computational cost of all the methods increases linearly with the number of simulated detector



(a) (b)

Fig. 9. Performance evaluation for standard GAN models **(a)** and conditional ones **(b)**

responses, the improvement achieved by the generative models is massive and grows with increasing number of clusters. The interesting issue is noticeable for shallow models when generating limited amount of outputted clusters. In such a case the constant time needed to set up the environment's session is higher that the actual time needed for generation, hence the pick in computational time is visible. As for the deeper models this problem is negligible.

We can observe similar speed-up when analysing the computational costs of conditional models. Here however, we take into account that both GEANT and conditional GANs require the simulation of initial particle momenta with a monte carlo method. Therefore the observed speed-up for conditional approach is by assumption slightly lower then for the previous task as presented in Fig. 9(b). Although the process of particle simulation takes only around 5% of the whole time, the computational cost for conditional models is also affected by the required extensions in the networks depth. Nevertheless the observed speed-up for conditional approach still reaches the factor of *20-25* for convolutional architecture or up to *100* for the LSTM based one.

It is worth mentioning that the presented results refers to the execution on a single-core CPU, while additional hardware-based speed-up of another order of magnitude was observed when using the GPU-based implementation for the neural network methods. Although Monte Carlo simulations can also benefit from such an acceleration, its iterative character does not allow the same improvement as in the case of deep neural networks.

7. Summary

In this work, we demonstrated the potential of machine learning generative methods namely Generative Adversarial Networks and Variational Autoencoders for cluster simulation in the high-energy physics experiments. We proved the possible application of those methods using the example dataset generated for the TPC detector in the ALICE Experiment at LHC. We implemented and evaluated several architectures that are based on generative models, and compared their results in terms of quality and computational cost. We adapted a progressive training technique to enhance the quality of generated samples. We also introduced the new training procedure which proved to provide more precise results in faster way than the standard approaches.

We also extended our previous works by applying conditional Generative Models, which may be used as a faster solution then currently used transportation packages.

The quality of the best performing method is not yet equal to the quality of currently used simulation methods. However, the results analysis revealed that proposed solutions can well reproduce the whole events while maintaining the laws of physics and almost ideally reproducing the restrictions observable in the original dataset. The most important advantage is that described methods are much faster. The observed computational speed-up is unprecedented and reaches up to 10^2 when compared to the currently employed GEANT 3 simulation technique.

Understanding the limitations of proposed methods, we plan to extend our works to reduce the inaccuracy of generated examples. With increased accuracy we also plan to use presented models as a main component of the semi-real time anomaly detection tool.

8. Acknowledgments

The authors acknowledge the support from the Polish National Science Centre grant no. UMO-2016/21/D/ST6/01946. The GPUs used in this work were funded by the grant of the Dean of the Faculty of Electronics and Information Technology at Warsaw University of Technology (project II/2017/GD/1).

References

Aamodt, K. et al. (2008). The ALICE experiment at the CERN LHC, *JINST* **3**: S08002.

Abadi, M. et al. (2016). Tensorflow: A system for large-scale machine learning, *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283.
URL:`https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf`

Abelev, B. et al. (2014). Upgrade of the ALICE Experiment: Letter Of Intent, *J. Phys.* **G41**: 087001.

Agostinelli, S. et al. (2003). GEANT4: A Simulation toolkit, *Nucl. Instrum. Meth.* **A506**: 250–303.

Brun, R., Bruyant, F., Carminati, F., Giani, S., Maire, M., McPherson, A., Patrick, G. and Urban, L. (1994). GEANT Detector Description and Simulation Tool, *CERN-W5013, CERN-W-5013, W5013, W-5013* .

Chollet, F. et al. (2015). Keras, `https://github.com/fchollet/keras`.

de Oliveira, L., Paganini, M. and Nachman, B. (2017). Learning particle physics by example: location-aware generative adversarial networks for physics synthesis, *Computing and Software for Big Science* **1**: 4.

Deja, K., Trzcinski, T. and Graczykowski, L. f. t. A. C. (2018). Generative models for fast cluster simulations in the tpc for the alice experiment, *3rd Conference on Information Technology, Systems Research and Computational Physics (ITSRCP)*.

Dellacasa, G. et al. (2000). ALICE: Technical Design Report of the Time Projection Chamber, *CERN-OPEN-2000-183, CERN-LHCC-2000-001* .

Evans, L. and Bryant, P. (2008). LHC Machine, *JINST* **3**: S08001.

Ferrari, A., Sala, P. R., Fasso, A. and Ranft, J. (2005). FLUKA: A multi-particle transport code, *CERN-2005-010, SLAC-R-773, INFN-TC-05-11* .

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks, *International Conference on Artificial Intelligence and Statistics*, pp. 249–256.

Glorot, X., Bordes, A. and Bengio, Y. (2011). Deep sparse rectifier neural networks, *International Conference on Artificial Intelligence and Statistics*, pp. 315–323.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative adversarial nets, *NIPS*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks, *Science* **313**(5786): 504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural computation* **9**(8): 1735–1780.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, *ICML*.

Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vanhaelen, Q., Khrabrov, K. and Zhavoronkov, A. (2017). The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology, *Oncotarget* **8**(7): 10883.

Karras, T., Aila, T., Laine, S. and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation, *CoRR* **abs/1710.10196**.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, *CoRR* **abs/1412.6980**.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes, *CoRR* **abs/1312.6114**.

Lucic, M., Kurach, K., Michalski, M., Gelly, S. and Bousquet, O. (2017). Are gans created equal? a large-scale study, *CoRR* **abs/1711.10337**.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets, *arXiv preprint arXiv:1411.1784* .

Paganini, M., de Oliveira, L. and Nachman, B. (2017). Calogan: Simulating 3d high energy particle showers in multi-layer electromagnetic calorimeters with generative adversarial networks, *CoRR* **abs/1705.02355**.

Radford, A., Metz, L. and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks, *CoRR* **abs/1511.06434**.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. and Lee, H. (2016). Generative adversarial text to image synthesis, *CoRR* **abs/1605.05396**.

Sjostrand, T., Mrenna, S. and Skands, P. Z. (2006). PYTHIA 6.4 Physics and Manual, *JHEP* **05**: 026.

Skands, P. Z. (2010). Tuning Monte Carlo Generators: The Perugia Tunes, *Phys. Rev.* **D82**: 074018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting, *JMLR* **15**(1): 1929–1958.

Suaide, A. A. D. P., Prado, C. A. G., Alt, T., Aphecetche, L., Agrawal, N., Avasthi, A., Bach, M., Bala, R., Barnafoldi, G., Bhasin, A. et al. (2014). O2: A novel combined online and offline computing system for the alice experiment after 2018, *Journal of Physics: Conference Series*, Vol. 513, IOP Publishing, p. 012037.

Yan, X., Yang, J., Sohn, K. and Lee, H. (2016). Attribute2image: Conditional image generation from visual attributes, *ECCV*.

Received:
Accepted: