Fast Simulation of Muon Backgrounds at SHiP using Generative Networks

A. Marshall[1], K. Petridis[1], and N. Serra[2]

[1]HH Wills Physics Laboratory, University of Bristol
[2]Physics Institute, University of Zurich

September 27, 2018

Abstract

We present a novel and fast approach to simulating muon backgrounds at SHiP. Through the use of generative networks we show we can accurately avoid a large bottleneck in the full SHiP simulation. We investigate and provide comparison between two popular modern generative network approaches from the machine learning community, Generative Adversarial Networks (GANs) and Varational Auto-encoders (VAEs). We conclude that the adversarial approach to training, GANs, can produce a more realistic output. For the simulation requirements at SHiP we show generative networks are capable of accurately approximating the full simulation (Pythia8 and GEANT4) of the dense fixed target with a speed up of $\mathcal{O}(10^6)$. The model produced here has been implemented in the SHiP simulation software. Comparison of hits and track reconstruction in detectors of the bottleneck between generated and fully simulated data shows good alignment. Methods presented in this paper can be easily generalized and applied to modeling any non-discrete multi-dimensional distribution.

1 Introduction

Generative networks have been studied in the machine learning community primarily for image generation applications. Each image in a training set has hundreds or thousands of pixels representing a high dimensional space. Within this space underlying features of the image set are encoded through dependencies between pixels. Generative networks attempt to model these underlying distributions that define a specific set of training images. These models can then be used to generate fake images that one would struggle to separate from the training set. Generative networks have shown some astonishing results: generating high quality images that obey fundamental features of training set images [19] [9], images from descriptive text [28], modeling image captions [18], photo realistic super resolution images [13] and high resolution images from semantic mapping [26] [11].

Current trends in high energy physics are seeing higher intensities across a range of experiments (SHiP and HL-LHC). This combined with a future increase in complexity of simulation software, will see simulation of high energy physics becoming exponentially more computationally expensive [3]. Generative networks of some form will be a useful tool in helping to provide a fast approach to simulation that can keep up with demands.

Elements of high energy physics simulation can be seen as highly complex functions transforming an input to an output. The reach and shape of an output distribution is dependent on the input and details of the simulation which can include stochastic elements. Generative networks can provide accurate representations of these intractable functions, without any prior inference [14].

Generative models have been applied to high energy physics. Current physical applications are in learning 2D energy distributions, with successful generation of 2D jet images [5], modeling showers in calorimeters [7] [17], modeling detector response and reconstruction algorithms [16] and learning features of jets [15].

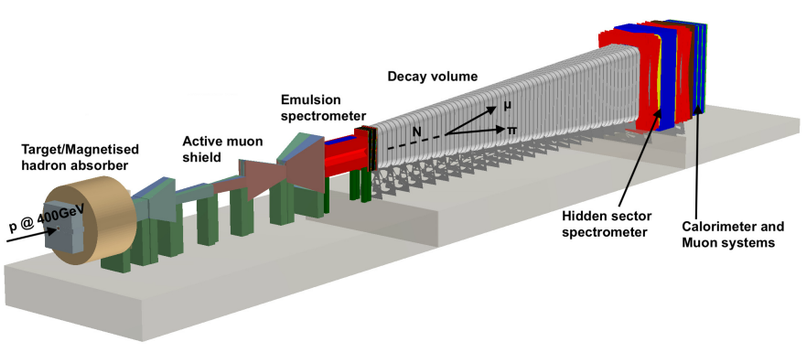


Figure 1: SHiP geometry (2018)

The SHiP experiment is a general purpose beam dump facility at the SPS at CERN searching for very weakly interacting long lived particles [4]. Searching for new physics at the intensity frontier, SHiP plans to collide 4×10^{13} 400 GeV protons on a dense W/Mo target (POT) over each 1s spill. This amounts to 2×10^{20} POT integrated over the lifetime of the experiment. This extreme beam intensity induces a large muon background of ($\sim 4 \times 10^{11}$ muons/s) leaving the hadron absorber (see figure 1). Background

reduction is paramount at SHiP, the aim is to have < 0.1 background events over life of experiment. A complex optimized arrangement of magnets downstream of the hadron absorber acts as a sweeper, reducing the flux of muons at downstream detectors by $\mathcal{O}(10^{-6})$ [2]. Any muons breaking this shield must be well understood to optimize the design of and verify vetoing detectors and cuts.

The SHiP simulation contains a large bottleneck. Simulating 400 GeV protons on the dense W/Mo target is computationally expensive. A muon reaches the end of the hadron absorber every $\mathcal{O}(10^3)$ POT. This takes $\mathcal{O}(5)$ minutes of CPU time, even after cutting any particle of energy < 10 GeV. We teach generative networks the properties of muons that reach the end of the hadron absorber. When trained these networks can accurately approximate these slow simulation steps. Models quickly producing large samples of muons which can be fed into the full simulation after the bottleneck.

Add paragraph justifying the approach, why this is statistically valid to do. (shutting thomas up paragraph) - adapting training sample can allow us to understand dangerous muons. if we have a generated distribution in which we have over or underesstimated the tails - run loads and loads of this. The muons that break the shield we can assign a weight to: if they are from a p-pt bin we have overesstimated (understand this by comparing full sim to generated distributions) we can weight this muon down. CAN EVEN GET AN ERROR ON THAT WEIGHT

2 Generative Models

There are two popular generative network architectures explored in this paper, both of which are built of two neural networks and a latent space of random noise z for generation.

2.1 Generative Adversarial Networks (GANs)

GANs employ two neural networks, the generator and the discriminator [8]. The generator $G(z;\theta_g)$ maps a randomly generated latent space z to a generated image x_{gen} and has trainable parameters θ_g. The discriminator $D(x;\theta_d)$ maps an input image x to an output prediction y (between 0 and 1) representing the probability x was either from training sample x_{real} or was a synthetic image from G, x_{fake}. See figure 2. In this configuration real data is labelled 1 and generated data 0. D is trained in isolation to correctly assign predictions to x_{gen} and x_{real} labelled data via a binary crossentropy loss function,

$$L_d = -(log(y_{true}) + log(1 - y_{pred})), \tag{1}$$

where y_{true} is the true label of x and y is output of $D(x)$. G is trained in a stacked model which directly connects output x_{gen} of G to D, in this stacked model all training parameters of D, θ_d, are frozen. The trainable parameters of G, θ_g are updated based on the loss function tied to the output of D,

$$L_g = -log(y_{pred}), \tag{2}$$

where y_{pred} is $D(x_{gen})$. The goal is for G to produce x_{gen} that is mislabeled by D as real x_{real} images from training sample ($y \simeq 1$). D and G are iteratively trained as to improve together. Training is complete when generated samples $G(z)$ are indistinguishable from x_{real}.

2.2 Varational Auto-Encoders (VAEs)

maybe using μ is confusing with muons?

VAEs are also formed of two neural networks, the encoder $E(x;\theta_e)$ mapping sample images x to a compressed latent space z and the decoder $D(z;\theta_d)$ which maps z back into images x_{out}, see figure 3. The distribution in z is an encoded representation of the full sample.

Importantly the VAE approach allows moulding of latent space z, the latent space representation is forced to be an uncorrelated multi-dimensional normal distribution. To achieve this the encoder $E(x;\theta_e)$ doesn't directly output z. Instead E outputs two vectors z_μ and $z_{log(\sigma^2)}$ holding values representing the mean and log variance of a normal distribution. The ith component of z, z^i is then sampled as (am i allowed to super script like this? i think maybe it looks way more messy than it needs to),

$$z^i = z^i_\mu + \epsilon \odot e^{(z^i_{log(\sigma^2)}/2)}, \tag{3}$$

where $\epsilon = \mathcal{N}(0,1)$. Simplistically the z_μ encoding defines where in z space a specific sample x should sit and the second term in equation 3 defines an possible region around that point.

The decoder then translates the encoded representation z back to real space x_{out}. Output samples x_{out} are then compared to corresponding inputs x using a binary cross-entropy term H in the loss function,

$$L_v = H + D_{KL}, \tag{4}$$

where the second term D_{KL} is the Kullback–Leibler divergence.

$$D^i_{KL} = -\frac{1}{2}\sum(1 + z^i_{log(\sigma^2)} - (z^i_\mu)^2 - e^{z^i_{log(\sigma^2)}}), \tag{5}$$

which is small if dimensions of z are close to $\mathcal{N}(0,1)$. Having a loss function that is a combination of these two effects creates a continuous and meaningful latent space. D_{KL} pushes the each dimension in z space to target values of $\mu = 0$ and $\sigma = 1$ and closes gaps in latent space. The H term in the loss function keeps z well separated, that is, regions have meaning and similar inputs are clustered.

To generate samples a value is generated from $\mathcal{N}(0,1)$ for each dimension of latent space, z is then passed through the decoder.

3 Our Approach

3.1 Full Simulation

A large simulation run ($\mathcal{O}(10^{11})$ POT) is completed of the target and hadron absorber complex by the SHiP collaboration with an un-magnetized hadron absorber [4], see figure 1. This simulation is run using Pythia8 [23] for 400 GeV protons on target, output particles are then transported using GEANT4 [1]. Mixed into this production are charm and beauty events from the SHiP cascade event generator [6]. Muons that reach the end of the hadron absorber are collected and kinematic information about the start of their tracks $S = [x,\ y,\ z,\ p_x,\ p_y,\ p_z]$ is saved. To simulate the rest of the experiment, muon tracks are started in the simulation of the full experiment at these same start of track kinematics S. It is this second simulation run that the hadron absorber is magnetized. The advantage of this modular approach to simulation is that after large simulation runs of the target the level of smear on the beam and the properties of a field in the hadron absorber can be changed, if the design changes. It is these 'start of track properties', S, of both μ^+ and μ^- that we aim to generate. Our training sample is made up of 3.4×10^8 muons.

3.2 Pre-processing

Start of muon track information S is extracted from collaboration target simulation output files, μ^+ and μ^- information is separated. A significant fraction of muons ($\sim 55\%$) extracted have start $x = 0$ and $y = 0$, this introduces a non continuous feature into the x and y distributions. Typical generative networks struggle to accurately model discrete features. To deal with this the sample is split again into muons that start at (0,0) and those that do not. We now have 4 separate samples, see table 1.

Category	Particle	Saved Properties					
1	μ^+	$x \neq 0$	$y \neq 0$	z	p_x	p_y	p_z
2	μ^+	$x = 0$	$y = 0$	z	p_x	p_y	p_z
3	μ^-	$x \neq 0$	$y \neq 0$	z	p_x	p_y	p_z
4	μ^-	$x = 0$	$y = 0$	z	p_x	p_y	p_z

Table 1: Saved muon properties from start of track information S for each category sample.

Every muon in the training set is converted into a vector of 4 features (category 2 and 4) or 6 features (category 1 and 3). Two generative network structures are designed, one to train on 4 feature vectors and one for 6 feature vectors.

Real parameter values are normalized to create input training sets. Each sample is normalized separately with minimum and maximum values stored for conversion of generated output back to physical values. Feature values are normalized to values between -0.97 and 0.97. Our GAN output layer is fitted with a *tanh* activation layer (see section 4.2) moulding output values to between -1 and 1. (VAE is currently sigmoid - test with tanh) Having the input normalized slightly more tightly removes a restriction on output parameter values.

In the 6 feature case the features x and y are extremely peaked distributions, even after the split shown in table 1. This proves hard for these generative networks to model. We make the task easier by widening the distribution in a reversible manner. For each point in the distribution we take the distance from the mean, square root the modulus of this value then reapply the correct sign. This distribution is again normalized. The distribution is now wider and can be more accurately modeled. This shouldn't reduce how effectively a well trained model can understand correlations between features as the operation is smooth and reversible.

4 Network Architecture and Optimization

We run optimization tests on a small GPU cluster for 4 and 6 feature GANs and VAEs. We qualitatively run tests over hyper-parameter space running tests for significant periods of time before judging success of the hyper-parameter choice. We test performance over changes in number of nodes, number of layers, the learning rate and optimizer choice. During a training run we track progress of generated samples using the figure of merit described in section 4.1.

All networks are trained with a mini-batch gradient descent approach, each training step the networks sees a small sample of training data. This is a hybrid approach between showing the full training sample each training step and showing just a single image each step (stochastic gradient descent) [21]. Although showing the full training sample would be the most accurate approach is it very time consuming and inefficient. Showing small samples is fast and it also introduces a stochastic element to the training which can effectively kick a network out of the equivalent of a local minima. The hybrid mini-batch approach allows fast and accurate training. Batch size is another tunable parameter.

In this paper both the GANs and VAEs only use dense fully connected layers. This, as opposed to any arrangement of convolutional layers is intuitively the correct approach. Filters in convolution layers instantly imply a positional dependence on features in muon parameter vectors. In generative networks for image applications this is useful, the positional relation of features are important. The networks in this case though should not have any prior expectation for this. The order that the vector $S = [x,\ y,\ z,\ p_x,\ p_y,\ p_z]$ is put together should not change the final result.

4.1 Figure of Merit

The output of models is either 6 or 4 features, a high quality output sample should not only produce individual feature values in the correct distribution but also should have the correct correlations between features. In

this high dimensional output space data points are sparse and traditional goodness of fit methods (e.g. χ^2) break down as almost all bins have low occupancy. To combat this we train a boosted decision tree (BDT) to distinguish between generated muon samples and real muon samples, this acts similarly the discriminator D of a GAN (section 2.1). The BDT is trained on a test sample of 50k real muons and a sample of 50k generated muons. After training the BDT can then be queried on an unseen 50k generated and an unseen 50k real samples, a traditional goodness of fit technique can then be applied to the 1 dimensional output distributions [27], for example see figure 4. The properties of the BDT are chosen to provide a continuous output for full samples. χ^2/N_{dof} values calculating the similarity between distributions for real and generated samples in the 1D BDT output space can then be used as an accurate figure of merit (FOM).

4.2 GAN Architecture

The final GAN architecture for the 6 feature case is shown in figure 2. Both G and D have 2 hidden layers in an inverted pyramid structure, with the number of nodes being (1536, 3072) and (768, 1536) for G and D respectively. The architecture for the 4 feature case is similar having the same D, G in the 4 feature case again has 2 hidden layers but less nodes at (512, 1024) and again in an inverted pyramid structure. The generators in both cases take an input of 100 dimensional latent space, where each dimension is a value sampled from $\mathcal{N}(0, 1)$.

Dropout layers are added between each dense layer to help prevent over fitting [25]. Batch normalization is used between every layer of G, this can increase stability at higher learning rates and add regularization which can help prevent over fitting [10]. For successful architectures batch normalization was found to decrease performance when applied to layers in D.

Leaky Rectified Linear Units activation layers are used at every hidden layer. The last layer of G has a tanh activation function, this is in accordance to the training samples normalizations between -1 and 1. The last layer of D has a sigmoid activation function providing an output between 0 and 1 representing the discriminator's judgment on the origin of the tested sample.

Mode collapse is a common failure in GANs, where the generator locks onto one particularly successful output image which is able to constantly trick the discriminator, this single output acts like a local minimum. We try qualitatively test a stabilization technique and see no effect on output. The labels of both training and generated images are blurred with Gaussian noise of $\mu = 0$ and $\sigma = 0.3$, [22]. The GAN gains stability when the job of D is made harder [24]. Even after multiple epochs we see no evidence of mode collapse or over fitting in GAN output, with or without this technique. This is due to our sample being very large and high entropy.

We find the Adam optimizer to perform best for this network architecture [12]. We find employing AMSgrad massively increased the stability of our output loss and FOM values [20]. The Adam optimizer has a tunable momentum controlling parameter β_1 that can be varied which is shown to provide mixed results, this paper uses $\beta_1 = 0.5$ as suggested by Radford et. al [19].

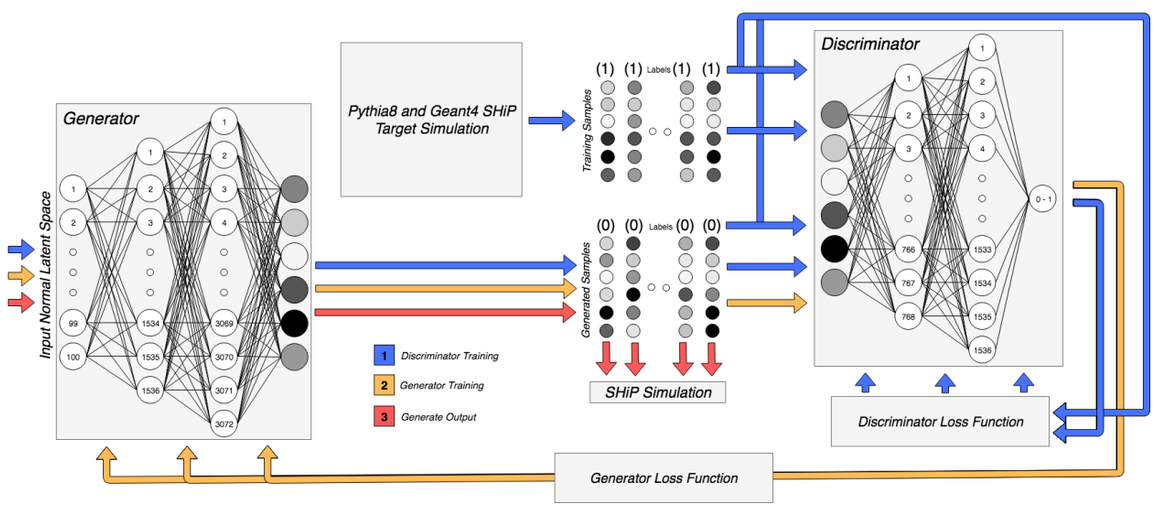


Figure 2: GAN architecture, arrows indicate the flow of samples and loss information for each stage of training and generation.

4.3 VAE Architecture

Best found VAE structure is displayed in figure 3. This has an encoder E with 2 hidden layers in a square structure both with 1250 nodes. The μ, $log(\sigma^2)$ and z layers have 5 and 3 dimensions for the 6 and 4 parameter cases respectively. The latent space representation of x is of lower dimensions, E must be efficient to effectively compress the information. In the final architecture the decoder D has one hidden layer also of 1250 nodes.

Performance of this VAE is poor with the standard loss function in equation 4. The loss function is dominated by small natural fluctuations in the D_{KL} term. To improve this we introduce a factor κ to the loss function,

$$L_v = \kappa H + D_{KL}, \tag{6}$$

where κ controls the relative importance of each term. We find a high value of $\kappa \gtrsim 1000$ to massively improve the performance of the VAE, the output more faithfully represents the input and the latent space remains as a multivariate Gaussian. Convergence time increases as κ increases, this demonstrates the dominance of the D_{KL} term when $\kappa = 1$.

We find VAE performance to be less architecture dependent than that of the GAN. VAE performance is observed to be mostly consistent over significant changes of in number of nodes in each layer. However the performance of VAE was found across a wide range of hyper-parameters to under perform relative to the best GAN architectures.

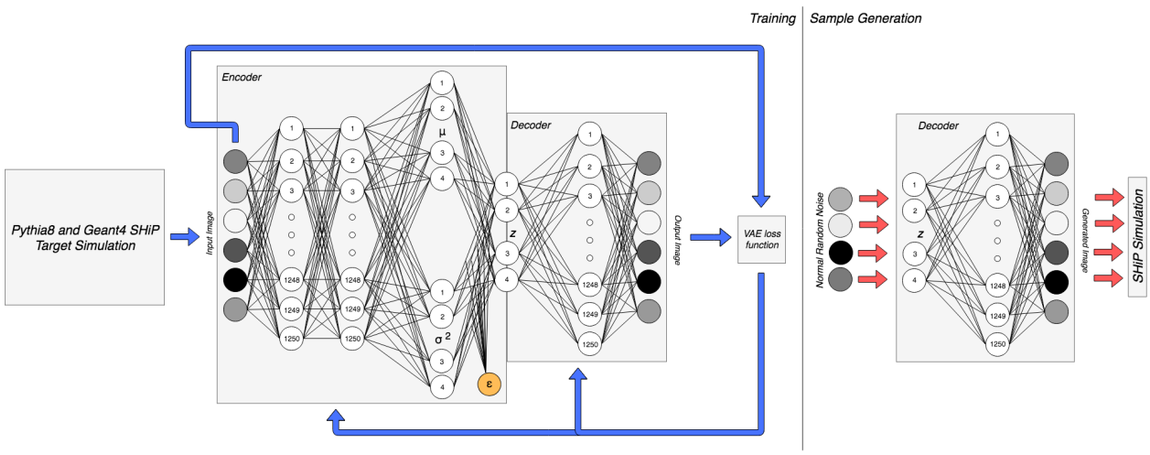


Figure 3: VAE architecture. Sample generation process is shown separately.

5 Method Comparison

5.1 Compare

We can judge generated output qualitatively with visual inspection of generated data plotted against full simulation data. This can be comparing both 1D histograms of feature output values or 2D histogram between each unique pairs of features, in the 2D histograms there is information about how well the network is grasping the correlations between parameters. We can also judge quantitatively looking for alignment in output from our figure of merit (section 4.1).

We find that our most successful GAN architecture, given enough training time, outperforms any tested VAE architecture. We can qualitatively understand the improvement to come from the adversarial component of the GAN. All the VAE has to learn from is the value of the loss function, but this loss function encodes both the requirement for a multivariate Gaussian latent space as well as accurate reconstruction of initial image. All VAE training information rolled into single value. For each training step the value for the loss is computed comparing an input batch to output batch, where the output batch (when $\mu \sim 0$ and $\sigma \sim 1$) no longer has a connection to the input encoded space, all variation comes from $\epsilon = \mathcal{N}(0, 1)$. High quality generated samples can still produce a large loss value. In the GAN approach this is not the case.

The GAN takes longer to reach optimal values for the FOM. Training can sometimes have slow improvement for a while, before wild improvement probably due to D finally grasping a correlation/characteristic of the training sample.

From here on we will focus solely on the higher quality generated data from the GAN architecture described in section 4.2 and shown in figure 2.

6 GAN Generated Sample

The BDT figure of merit output plots for the final GAN models chosen for generation in this paper are presented in figure 4. A full generated sample is prepared by combining output from this 4 GANs in the correct physical production fractions.

6.1 Comparison to Training Sample

The training sample (full simulation) has some muon production channels enhanced by a factor of 100, namely vector meson decays and γ conversion. Weights are taken into account in the training of the GAN to negate this effect, the GAN produces muons from different sources in the correct physical ratios. In all comparison plots in this paper (excluding figures 8 and 9) muons are sampled from the training sample in the physically correct proportions as to make comparison possible.

A one dimensional comparison of each of the 6 features of the muons is made between the full simulation and generated data in figure **??** (REMOVED). Figure 5 combines features plotting total momentum of each muon and transverse momentum of each muon, the alignment here displays how well the GAN understands the correlations between parameters. Figures 6(a) and 6(b) display directly appreciable physical correlations between features. Figure 6(b) displays the x and y position of the start of tracks, and figure 6(a) shows how similar generated and full simulation samples are in momentum phase space.

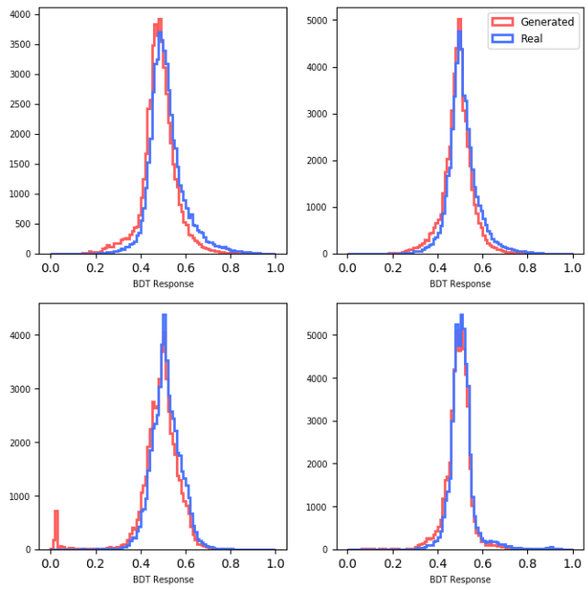


Figure 4: Distributions of BDT output values for both generated and full simulation samples. Results are separated for each of the 4 GAN generators used to generate data for this paper.

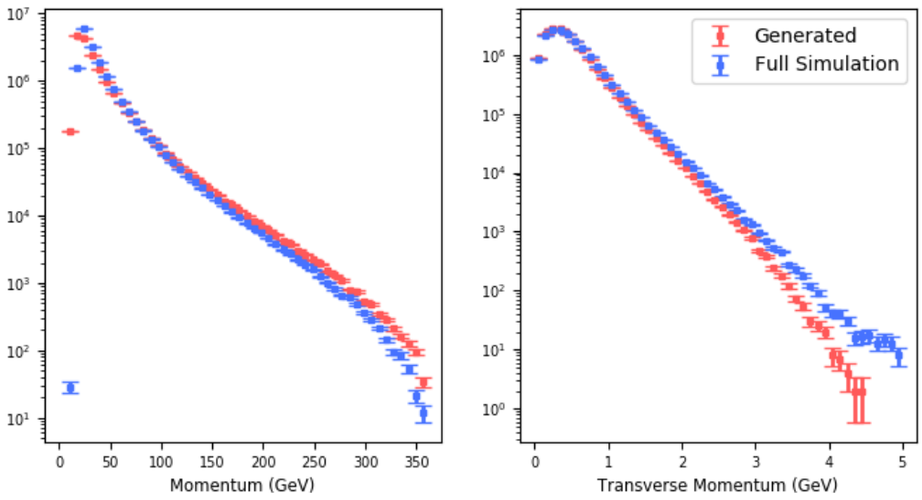


Figure 5: Distributions of total momentum and transverse momentum of both the full simulation and GAN generated sample at the start of muon tracks.

6.2 Benchmarking

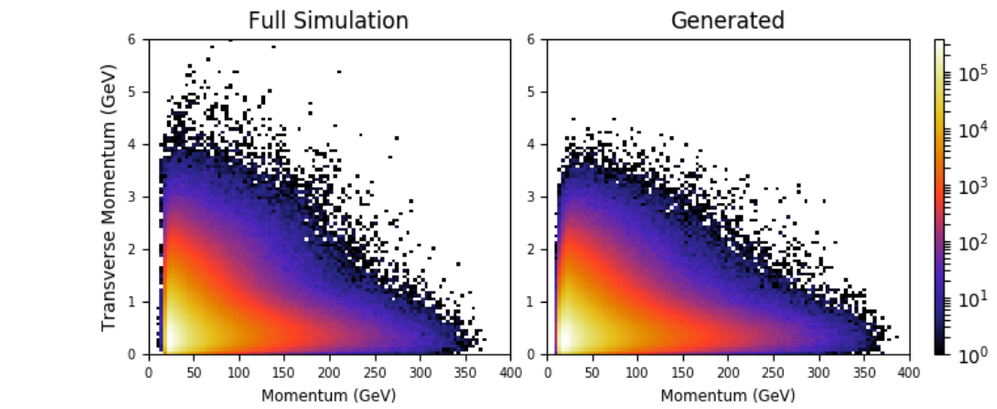
As mentioned in section 1, in the full simulation one muon leaves the hadron absorber every $\mathcal{O}(5)$ minutes on a single CPU. With only a small expense in the fidelity of the generated sample GANs can produce data a lot faster. We measure a speed up of $\mathcal{O}(10^6)$. This is running with Keras (v2.1.5) on a Tensor Flow backend (v1.8.0) all on a single Nvidia Pascal P100 GPU card. This speed up figure includes all computation required to un-normalize the GAN output back into physical values. Generation using GANs is an order of magnitude slower on CPU.

7 Tails of distribution and adaptations to training sample

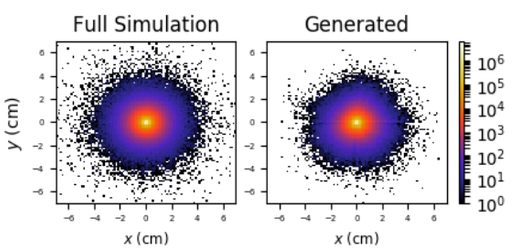
Dope the GAN to generate potentially dangerous regions

8 Using Generated Sample in FairShip

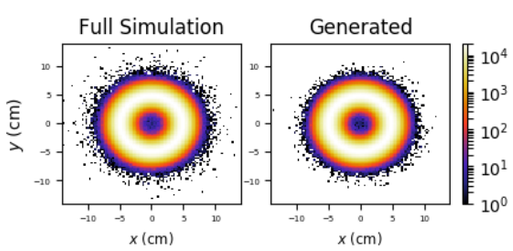
To further verify the success of the GANs in generating a sample that accurately represents the fully simulated training sample, we run the generated sample through the rest of the SHiP simulation (after the bottleneck). We start μ^+ and μ^- tracks at generated positional values with generated kinematics and transport these muons down the experiment with Geant4. We track hits in downstream sensitive volumes and detectors. Hit maps for these volumes in runs of both generated and full simulation muons are displayed in figure 8. The enhancement effect mentioned in section 6.1 means some hits in these full simulation maps are weighted up by a large factor. These hit maps are some what normalized to this effect and z axes are matched, but the enhancement causes the real sample hit maps to be less smooth. After accounting for this in the judgment



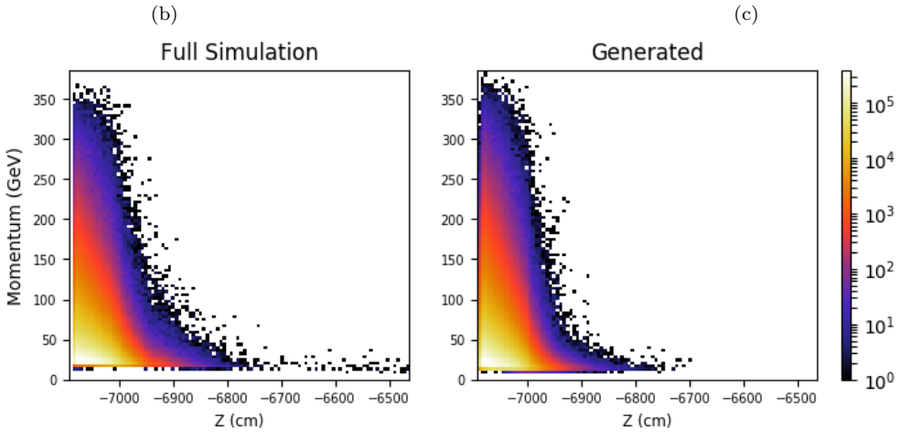
(a)



(b)



(c)



(d)

Figure 6: (a) Distribution of muons on a plot of total momentum against transverse momentum at the start of muon tracks. Plotted are both the full simulation and GAN generated samples. (b) Distribution of x and y position of muons at the start of their tracks. (c) Distribution of x and y position after a realistic beam smearing has been applied. (d) Distribution in a plot of total momentum against z start position, where high z is downstream. (NOTE: high Z muons have very low mom - safe)

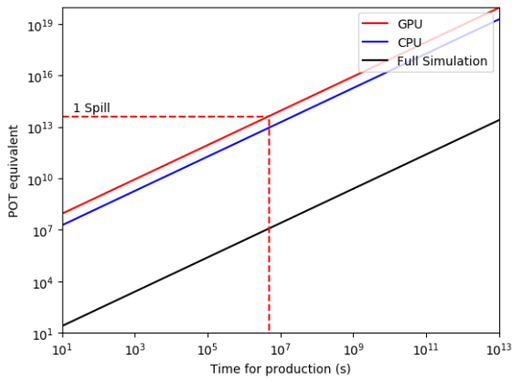


Figure 7: Plot of time for production against muon background simulated for POT equivalent. Assuming 1 muon with energy is higher than the 10 GeV cut leaving the hadron absorber represents 750 POT.

of alignment it is clear figure 8 shows off a success in the accuracy of the generated sample.

Classification of signal and background muons uses reconstructed track properties from hits in the straw-tubes at the end of the decay volume in the SHiP geometry. GAN generated muons must also produce tracks here that are physically accurate and comparable to those of the full simulation muons. Figure 9 shows a histogram of reconstructed track momentum. Generated muons inducing such a good response from the reconstructed tracks is evidence that GAN generated muons faithfully represent the training sample. The muons must be starting with physically correct kinematics, propagating correctly through detector and creating physically realistic tracks. Agreement between full simulation and generated muons here demonstrates that the GAN generated samples really are high enough quality to be usable to gain statistics and training data for background studies.

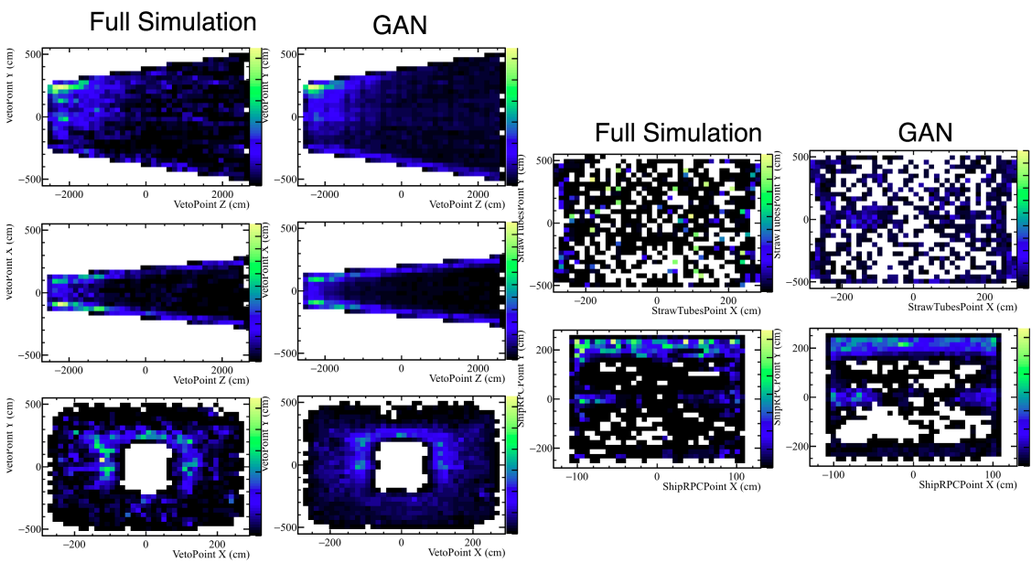


Figure 8: Hit maps in downstream sensitive volumes of the SHiP simulation software. Effects of GAN generated muons and full simulation muons on detectors can be compared. NEED TO REMAKE THIS FIGURE WHEN DONE WITH GAN

track_mom.png

Figure 9: Reconstructed track momentum values for GAN generated and full simulation muons.

9 Discussion and Conclusion

We have used modern machine learning methods to approximate the output of a complex and computationally expensive target simulation at the SHiP experiment. We show it is possible to create models that extremely accurately model training data, without prior inference these models can understand complex correlations and characteristics. These techniques produce unique samples unseen in the training data that obey all features of the real sample.

We have shown that generated muons behave correctly when fed into the full experiment simulation after the bottleneck. The output is physically correct and comparable to the full simulation.

Add paragraph justifying method and shutting thomas up.

If we are to assume that the kinematic phase space covered by the training data is representative of the full reach of muons leaving the hadron absorber from 400 GeV proton collisions, then so is the output of the GANs. If we assume this to be correct we can rely on huge muon production from these GANs as being representative.

The generative models developed in this paper can produce muons $\mathcal{O}(10^6)$ times faster than the full target simulation. This can offer a huge increase in understanding of statistically limited muon background studies at SHiP.

On a more general note, we show methods of using generative networks to approximate bottlenecks in a particle physics simulations. We discount VAEs for this particular application, finding GANs to produce much higher quality generated results. We demonstrate pre-processing ideas that both lead to increases in generated performance and are are easily implementable. It is clear that GANs can offer a good solution to quickly sampling from complicated intractable high dimensional distributions.

10 Acknowledgement

We would firstly like to thank the SHiP collaboration. Secondly, this work was carried out using the computational facilities of the Advanced Computing Research Centre, University of Bristol - http://www.bristol.ac.uk/acrc/. A Titan Xp was also used for this research which was donated by the NVIDIA Corporation.

References

[1] Sea Agostinelli, John Allison, K al Amako, John Apostolakis, H Araujo, P Arce, M Asai, D Axen, S Banerjee, G 2 Barrand, et al. Geant4—a simulation toolkit. *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003.

[2] A Akmete, A Alexandrov, A Anokhina, Shuji Aoki, Erica Atkin, N Azorskiy, JJ Back, A Bagulya, A Baranov, GJ Barker, et al. The active muon shield in the ship experiment. *Journal of Instrumentation*, 12(05):P05011, 2017.

[3] Johannes Albrecht, G Amadio, N Anh-Ky, L Aphecetche, J Apostolakis, M Asai, L Atzori, M Babik, G Bagliesi, M Bandieramonte, et al. A roadmap for hep software and computing r&d for the 2020s. 2018.

[4] SHiP Collaboration et al. Technical proposal, a facility to search for hidden particles (ship) at the cern sps, 2015. *arXiv preprint arxiv:1504.04956*.

[5] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science*, 1(1):4, 2017.

[6] H Dijkstra and T Ruf. Heavy flavour cascade production in a beam dump. *SHiP-NOTE-2015-009*, 2015.

[7] Martin Erdmann, Jonas Glombitza, and Thorben Quast. Precise simulation of electromagnetic calorimeter showers using a wasserstein generative adversarial network. *arXiv preprint arXiv:1807.01954*, 2018.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[15] James William Monk. Deep learning as a parton shower. *arXiv preprint arXiv:1807.03685*, 2018.

[16] Pasquale Musella and Francesco Pandolfi. Fast and accurate simulation of particle detectors using generative adversarial neural networks. *arXiv preprint arXiv:1805.00850*, 2018.

[17] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Accelerating science with generative adversarial networks: an application to 3d particle showers in multilayer calorimeters. *Physical review letters*, 120(4):042003, 2018.

[18] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pages 2352–2360, 2016.

[19] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[20] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. 2018.

[21] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[22] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[23] Torbjörn Sjöstrand, Stefan Ask, Jesper R Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O Rasmussen, and Peter Z Skands. An introduction to pythia 8.2. *Computer physics communications*, 191:159–177, 2015.

[24] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[26] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pages 1–13, 2018.

[27] Constantin Weisser and Mike Williams. Machine learning and multivariate goodness of fit. *arXiv preprint arXiv:1612.07186*, 2016.

[28] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint*, 2017.