University of BRISTOL

SCHOOL OF PHYSICS

FINAL YEAR PROJECT REPORT

NAME:	Michele Palazzo
DEGREE COURSE:	Mathematics and Physics
PROJECT TITLE:	Generative Neural Networks for fast and reliable simulations
YEAR OF SUBMISSION:	2024
SUPERVISOR:	Kostantinos Pedritis
NUMBER OF WORDS:	5885

Declaration

I hereby declare that the report titled “Generative Neural Networks for fast and reliable simulations” is a record of the research work carried out by myself, Michele Palazzo , under the guidance of **K.Petridis, School of Physics, University of Bristol.**

I affirm that the work presented in this report is original and has been done by myself in the duration of my project period. The Generative Adversarial Network (GAN) framework used in this work was adapted from the original Vanilla GAN architecture provided by my supervisor, which is part of the research presented in the paper “Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks” by the SHiP collaboration.

The construction of the Wasserstein GAN (WGAN) has been developed with the assistance of **A.M.Marshall**, whose insights and expertise in the field have been invaluable.

The training data utilised in this study were provided by the SHiP experiment via my supervisor.

The utilisation of XGBoost for evaluation, the tuning of hyperparameters, the incorporation of feature score guided training were independently conducted by me; these constitute the novel contributions.

All sources used and referred to have been acknowledged and listed in the reference section.
Michele Palazzo
18th April 2024

Acknowledgements

I wish to express my sincere gratitude to **A.M.Marshall** for his invaluable assistance with the construction of the Wasserstein Generative Adversarial Network used in this research . I also extend my appreciation to my supervisor, **K.Petridis**, for providing the original Generative Adversarial Network and for his guidance and support throughout this project.

Special thanks are due to the SHiP Collaboration for supplying the training data used in my analyses. The findings in this research have benefited from their groundwork, particularly as reported in their publication, "Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks".

Abstract

This project explores the advancement of the Generative Adversarial Network (GAN) used in “Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks” paper with a primary focus in efficiency and fidelity of data generated. Transition from a traditional GAN to a Wasserstein GAN with Gradient Penalty (WGAN-GP) yielded no improvements in fidelity or training speed. However, by implementing an importance score guided training process a significant decrease in convergence epochs and increase in fidelity was observed. These findings demonstrate that importance score guided training can significantly reduce the convergence epochs and increase the fidelity of generated distributions, offering a promising approach for enhancing GAN-based simulations in physics research.

Contents

1 Introduction

1.1 Objectives

The primary objective in this project was to enhance the Generative Adversarial Network (GAN) used in the "Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks" paper. This can be broken down into two components. Firstly, improving the accuracy of the generated muon distributions. Secondly, potential improvements in optimisation of the network architecture for efficiency are sought, specifically targeting a faster convergence and training time. These objectives were addressed using various techniques, including: the implementation of a Wasserstein GAN, the implementation of importance score guided training and various feature engineering techniques.

1.2 Introduction to the SHiP experiment and Neural Networks

GANs represent an exciting juncture in the realm of machine learning particularly in their application to complex scientific challenges like the SHiP experiment (Search for hidden particles). The SHiP experiment, set to take place at the proposed multi-purpose fixed target facility at CERN's Super Proton Synchrotron (SPS) accelerator, is designed to detect exotic particles with long lifespans. Targeting particles with masses up to a few GeV/c^2, the experiment seeks to explore particles emerging from the decay of heavy hadrons [5].The SHiP experiment aims to discover new particles that exist beyond the theory of Standard Model of particle physics, which could provide answers to some of the most fundamental questions about the nature of our universe.

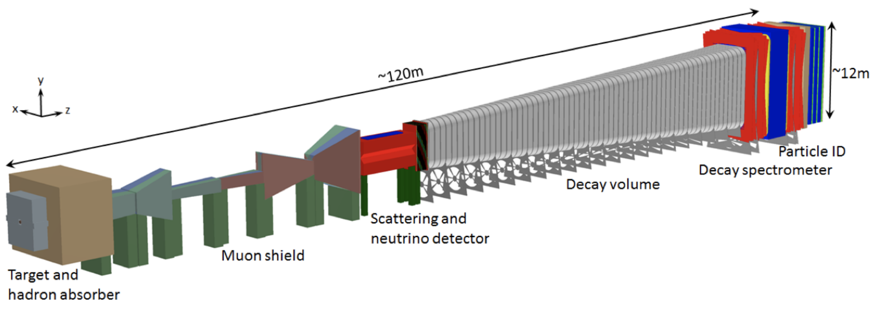


Figure 1: Overview of the SHiP experiment [4]

The SHiP set up can be visualised in Figure 1. A pivotal aspect in the set up is its active muon shield, designed to deflect muons generated in the beam dump away from the main detector. This muon shield is integral to maintaining a low-background environment necessary for detecting rare particle events. The detector setup includes a proton target followed by a hadron stopper, and the muon shield which incorporates a coil that magnetises the iron shielding[4]. In order to maximise the muon shield's efficacy, numerous simulation endeavours have been undertaken. For simulation of muon interactions, methods such as `PYHTIA8` [17] and `GEANT4` [3] were employed. The temporal efficiency of these methods are compared to a GAN in the "Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks" paper. Utilising GPU-accelerated GANs, the simulation produces approximately 3.5 million muons in a span of five minutes in comparison to approximately 1 muon per 5 minutes using `PHYTIA8` and `GEANT4` [4]. Clearly, GANs can expedite the generation of simulated data, which can provide invaluable insights in analysis and interpretation. In this project a Wasserstein GAN and an augmented version of the GAN used in "Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks" paper are implemented to simulate muons produced in interactions of the SPS proton beam.

In an artificial neural network, the neurons in the input layer serve as initial receivers of data, passing it through weighted "synapses" to the hidden layers, and eventually to the output layer. Each layer is made up of a set of nodes, connection between nodes in these layers is associated with a weight, which modifies the strength of the signal being transferred. When data passes through the network, the so called forward propagation phase, the data from the previous layer is linearly transformed using these weights and then passed through a non-linear activation function to compute the output for the next layer. This combination of linear and non-linear transformations allows neural networks to capture complex patterns in data. The activation function decides whether a node should be activated "fire" or not. It does this by taking the result of the linear transformation and determining the output of the node. Different activation functions like ReLU, Sigmoid, Softmax, and Hyperbolic Tangent serve various purposes and are chosen based on the specific needs of the network, such as the type of output desired [14]. The hidden layers, often described as 'black boxes', process the data in a non-linear fashion that is not easily interpretable

[10]. Learning in a Neural network occurs through a process called backpropagation, where errors in output predictions are used to adjust the weights across the networks. This process is guided in a quantifiable manner by a loss function which evaluates a set of weights [7].

1.3 Introduction to GANs

A GAN consists of two competing networks, a generator and a discriminator. Both networks are trained through a simultaneous adversarial process. The generator's objective is to create data that resembles the training data and the discriminator serves a discernment role, deciding whether a sample is authentic or a product of the generator. GANs are known for generating higher quality samples compared to Variational Auto Encoders (VAEs). VAEs can introduce blurriness in the generated samples due to the inherent Gaussian assumptions in their probabilistic model, GANs, especially Wasserstein GANs, produce samples with sharper and more detailed features. Traditional GANs, also known as *Vanilla GANs*, often face challenges such as mode collapse, where certain modes of a training distribution are not represented in generation despite being present in the training sample [18]. Wasserstein GANs address this issue through the use of the Earth movers' distance (see section [2.3]), which offers a more meaningful gradient for the generator to learn from. In practice, this ensures that the entire space of the target probability distribution is effectively covered, leading to more accurate and diverse simulation outcomes [13]. One of the critical advantages of Wasserstein GANs is their ability to mitigate mode collapse. This aspect is crucial for accurately simulating probability distributions, where capturing the entire range of possible outcomes is often essential for the validity of the simulation [20].

2 Background

2.1 Data Preprocessing Protocol

In developing the Wasserstein GAN (WGAN), preprocessing both the training and generated muon data is a critical step to ensure accurate generation of position and momentum vectors of muons at their production point. The preprocessing procedure, applied to the six features of the muons' spatial and momentum components (x, y, z, p_x, p_y, p_z), is as follows:

1. **Feature Scaling**: Each feature (x_i) is scaled to have a minimum value of zero by subtracting the minimum value $(\min(x_i))$ and dividing by the feature range $(\max(x_i) - \min(x_i))$.
2. **Normalisation**: The scaled features are then normalised to the range $[-1, 1]$ through the transformation $x_i = 2 \times (x_i - \min(x_i))/(\max(x_i) - \min(x_i)) - 1$.
3. **Batch Selection for Evaluation**: A batch of preprocessed muon data is randomly selected from the dataset for model evaluation, ensuring diversity and representativeness in the evaluation process.

This preprocessing routine ensures that the input data fed into the Wasserstein GAN is standardised, facilitating the efficient training of the network. The Minimum-Maximum scaling (an effective widening of the distribution) is especially crucial for the x and y position coordinates of the muons as they are extremely peaked around the interaction point, making it difficult for the generator to replicate these peaks accurately otherwise [5].

2.2 Wasserstein GAN architecture

The *generator* begins with a randomly distributed input noise vector. This vector is drawn from a standard Gaussian distribution and serves as a seed for data generation [2] . Following the input layer, *dense layers*, also known as fully connected layers, are employed. Each node in a dense layer receives input from all nodes of the previous layer, making these layers highly interconnected [19]. They are fundamental for transforming the input vector into a space with a higher dimension that gradually morphs into the shape of the target data distribution [1]. *Batch normalisation* is applied after each dense layer. This technique normalises the output of the previous layer by adjusting and scaling the activations. The tanh *activation function* is applied at the final layer. This non-linear function maps the output of the previous layer to a range between -1 and 1, which is necessary for matching the preprocessed data scale of the muon vectors.

The *discriminator* model processes inputs of shape $(1, 6)$, matching the dimensionality of the muon vectors (x, y, z, p_x, p_y, p_z). Following the input, a *flatten* layer is employed to transform the input tensors into a format suitable for dense processing [11]. The core of the discriminator is a series of *dense layers*. Each fully connected layer (dense layer) is followed by a dropout layer which randomly drops out inputs from the last layer [19]. These layers prevent the model from becoming fixated on a certain configuration of data and therefore hindering its discriminatory capacity. The discriminator's goal is to classify the inputs as real or generated. Lastly, a dense layer with a single unit and a linear *activation function* to output the classification score.

2.3 Optimisation and Loss Function

Optimisation techniques are pivotal for the models performance and stability. The WGAN uses the Adam optimiser for both the generator and discriminator, with carefully selected hyperparameters to ensure efficient training dynamics [12]. In selecting these parameters different combinations were employed however the parameters used in the old GAN architecture were found to be optimal (determined by AUC scores see section [2.5]) The Adam optimiser is utilised for its adaptive learning rate capabilities, which allow for more precise adjustments to the model weights during training. For both the generator and discriminator the Adam parameters are set identically. The specific configurations are as follows:

- Learning rate: 0.0002
- β_1: 0.5 (controls the exponential decay rate for the first moment estimates)
- β_2: 0.9 (controls the exponential decay rate for the second moment estimates)

The implementation of *Wasserstein loss* is a defining feature of the WGAN, providing a more stable training process and a meaningful way to measure the distance between the distribution of generated data and real data. This distance is known as the *Earth mover's distance*. Informally, Earth Mover's Distance quantifies the minimum "effort" needed to reshape one probability distribution to match another [16]. This distance is implemented into the loss function instead of traditional cross-entropy loss employed in the Vanilla-GAN.

The *discriminator's loss* is the difference between its average score on real data and its average score on fake data. This is represented as:

$$\mathscr{L}_D = \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] \tag{1}$$

where $\mathscr{L}_D$ is the loss of the discriminator, $D(x)$ is the discriminator's score for a real data point x, $D(G(z))$ is the discriminator's score for a generated data point by the generator G, p_{data} is the data distribution, p_i is the generator's input noise distribution, and i is a sample from the noise distribution.

The *generator's loss* is the negative of the discriminator's average score on the generated data. This encourages the generator to produce samples that the discriminator will score as realistic. The generator loss is formulated as:

$$\mathscr{L}_G = -\mathbb{E}_{z\sim p_z(z)}[D(G(z))] \tag{2}$$

where $\mathscr{L}_G$ is the loss of the generator.
The use of these loss functions aligns with the principles of the Wasserstein GAN, focusing on minimising the Earth mover's distance between two different distributions.

2.4 Gradient Penalty and Training Process

The Gradient Penalty (GP) method enhances the training stability of Wasserstein GANs (WGAN-GP) by ensuring the discriminator complies with the 1-Lipschitz condition, a condition stronger than continuity but not as strong as derivability [6]. This is achieved by penalising the gradient norm of the discriminator's output with respect to its input. The GP is applied to interpolated samples between real and fake data, which are generated by mixing real and fake data with random coefficients.
The gradient penalty function can be mathematically represented as follows:

$$GP(\lambda) = \lambda \cdot \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[(\|\nabla_{\hat{x}}D(\hat{x})\|_2 - 1)^2\right] \tag{3}$$

where λ is the penalty coefficient, $\hat{x}$ are the interpolated samples, $\mathbb{P}_{\hat{x}}$ is the distribution of these interpolated samples, D represents the discriminator function, and $\nabla_{\hat{x}}D(\hat{x})$ denotes the gradient of the discriminator's output with respect to $\hat{x}$.

During training, the discriminator is updated more frequently than the generator, in a specified ratio of 1 to 5, to maintain balance between the two networks. This process involves alternating between training the discriminator with both real and fake data (including the application of gradient penalty) and updating the generator to produce more realistic distributions. The training step can be outlined as follows:

1. For each batch of real data:
   (a) Generate a batch of fake data from random latent vectors.
   (b) Compute the discriminator loss using both real and fake data, including the gradient penalty term.

   (c) Update the discriminator weights by backpropagating the loss.

2. Update the generator by:
   (a) Generating a new batch of fake data.
   (b) Computing the generator loss based on the discriminator's output.
   (c) Backpropagating the loss to update the generator weights.

This training methodology, underpinned by the gradient penalty, attempts to mitigate common issues such as mode collapse[18], providing a robust framework for generating high-quality synthetic data that closely resembles the real data distribution.

2.5 Evaluation Using an XGBoosted Classifier

At specific intervals (50 epochs) during the training process of the Wasserstein GAN, the evaluation phase is initiated using the *XGBoost classifier* [8], this evaluation aims to assess how well the generated muons mimic the real muons, providing a quantitative measure of the GAN's performance.

The evaluation process begins by generating a set of muons using the current state of the generator. This is achieved by sampling noise vectors from a normal distribution and passing them through the generator. Simultaneously, a corresponding set of real muons is prepared for comparison. The real and generated images are then combined to form a dataset, with labels indicating their origin (real or generated - 0 and 1 respectively). This dataset is subsequently split into training and testing sets (using a 50-50 split), ensuring a balanced representation of both image types.

The *XGBoost classifier* is then trained on the training set (100,000 generated and real muons). It learns to distinguish between the real and generated distributions based on their inherent characteristics. The classifier's configuration is carefully chosen, incorporating parameters such as maximum depth, learning rate, and the number of estimators to optimise its discriminative capability. Upon completion of training, the classifier's performance is evaluated on the testing set. The accuracy of classification is calculated, providing insight into the classifier's ability to differentiate between real and generated images.

In order to obtain a quantifiable measure of the model's performance, the *Receiver Operating Characteristic (ROC) curve* and the Area Under the Curve (AUC) are computed. These metrics offer a comprehensive view of the classifier's effectiveness across various thresholds, illustrating its sensitivity and specificity.

The ROC curve is plotted, contrasting the True Positive Rate (TPR) against the False Positive Rate (FPR), with the AUC value serving as an aggregate measure of performance across all possible classification thresholds. An AUC score of 0.5 would indicate a classifier which finds the two classes to be indistinguishable. The distributions of the classifier's predicted probabilities for real and generated images are also examined, offering visual insights into the differentiation capability of the model. These distributions are plotted and analysed, contributing to our understanding of the GAN's current state. Analysing the parts of the distributions where the classifier is confident in its decision can give insights on model improvement.

In addition to evaluating the classifier's performance, the importance scores assigned to different features by the XGBoost model are assessed. These importance scores provide insights into which features contribute most significantly to the classification task. By analysing these scores, the most discriminative features between real and generated muons can be identified.

Importance scores are computed based on the frequency with which a feature is used in decision trees within the ensemble model and the magnitude of the improvement in model performance attributable to that feature. Features with higher importance scores are deemed more influential in distinguishing between real and generated muons. Understanding these scores helps in identifying the key characteristics that differentiate the two classes of data and can guide further refinement of the GAN architecture or training process.

2.6 Enhanced Vanilla GAN training architecture and protocol

In conjunction with the development of a Wasserstein Generative Adversarial Network (WGAN), a novel training protocol leveraging feature importance scores was explored. This innovative approach aims to refine the GAN model initially described in the context of "Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks". The changes made include alterations in preprocessing techniques and the training regime. The preprocessing protocol was altered to apply a Gaussian filter to the x and y position features instead of min-max scaling. Given a feature vector X_i for $i \in \{x, y\}$, the Gaussian filtered output $X_{i,\text{filtered}}$ is computed as:

$$X_{i,\text{filtered}} = \text{GaussianFilter}(X_i, \sigma) \tag{4}$$

The σ parameter controls strength of filter and was set to a value of 4. The remaining features were preprocessed as in subsection 2.1. Training begins with the deployment of an XGBoost classifier designed to discern between real and generated muons. The classifier's performance, detailed in subsection 2.5, underpins the extraction of feature importance scores. These scores, reflecting the relative impact of each feature on the classification accuracy, guide the emphasis placed on different aspects of the data during GAN training. Post classification, the XGBoost model's feature importance scores are extracted. The extracted importance scores are then incorporated into the GAN training pipeline. Specifically, the input noise vector to the generator was adjusted based on these scores, shifting priority to features with higher importance. Let N denote the original noise vector, W the importance weights derived from the XGBoost classifier, and α the emphasis factor. The adjusted noise vector N_{adj} is given by:

$$N_{adj} = N \cdot (1 + \text{reshape}(W(0,1)) \cdot \alpha) \tag{5}$$

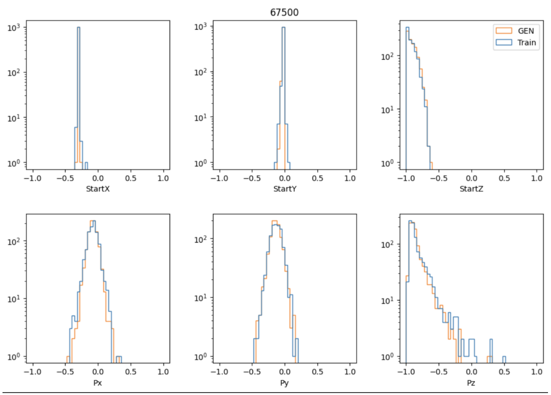
The adjusted noise vector is fed into the model every 50 iterations. This is because for every time the adjusted noise vector is utilised an XGboost evaluation is required and evaluating any more frequently yields small fidelity improvements for a large cost in training time. This methodology underscores a targeted effort to bridge the gap between GANs' theoretical potential and their practical efficacy in simulating complex distributions.

3 Results

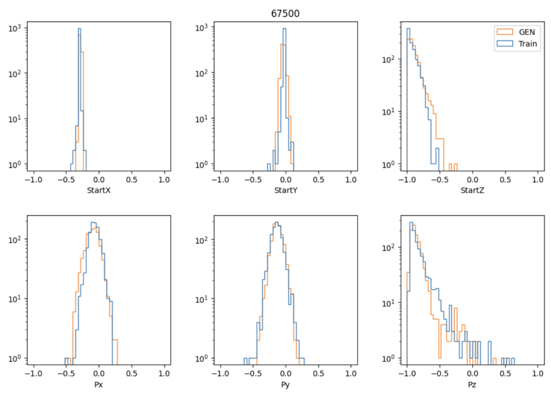
3.1 Wasserstein and Vanilla GAN distribution quality comparison

The original architecture used in the "Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks" paper and a Wasserstein GAN were each trained for 67500 epochs and the output quality of both were assessed using the boosted decision tree described in subsection 2.5. The real and generated muon feature distributions can be seen in Figure 2. This epoch number was chosen to assess the long term holistic behaviour of both models. Upon ocular inspection it may appear that both have yielded sensible results but the

Figure 2



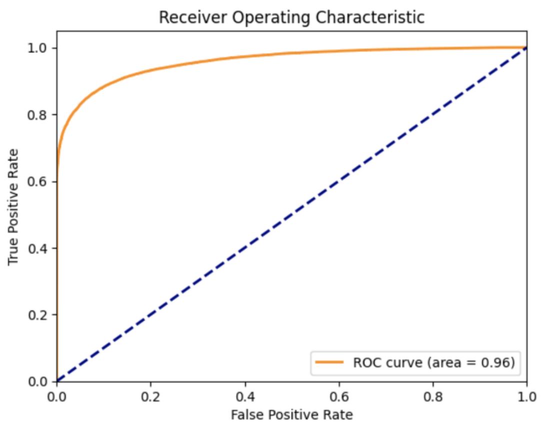
(a) Predicted and generated values (produced by the Vanilla GAN) for position and momentum in x, y and z after 67500 epochs.



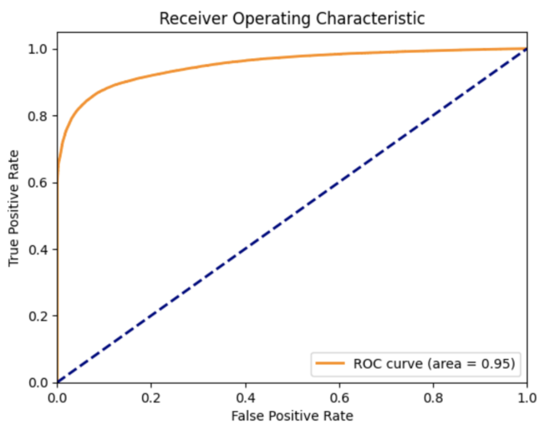
(b) Predicted and generated values (produced by the Wasserstein GAN) for position and momentum in x, y and z after 67500 epochs.

ROC curve and AUC scores, yielded by the boosted decision tree, give a quantifiable measure of how easily the these generated and real classes can be distinguished. The high AUC score visible in Figure 3a and Figure 3b suggest that in both cases, the decision tree was capable of discerning between the two classes with a high level of certainty. The dotted line in both figures represents a curve with an AUC score of 0.5 in which the generated and training distributions are indistinguishable. The importance scores from the boosted decision tree can assist with understanding which muon feature is most influential in the classification process.

Figure 3



(a) ROC curve produced from boosted decision tree evaluation at 67500 epochs from the Vanilla-GAN



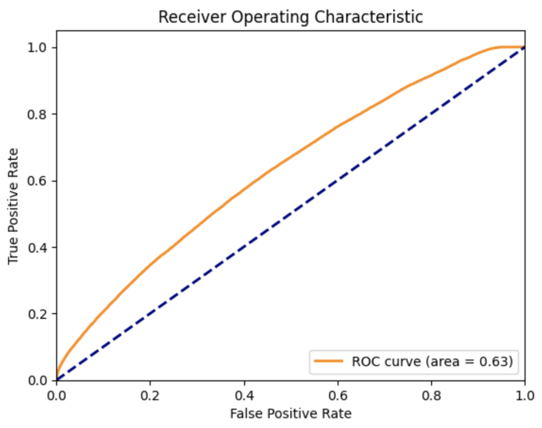
(b) ROC curve produced from boosted decision tree evaluation at 67500 epochs from the Wasserstein-GAN

Table 1: Feature Importance Scores for XGBoost

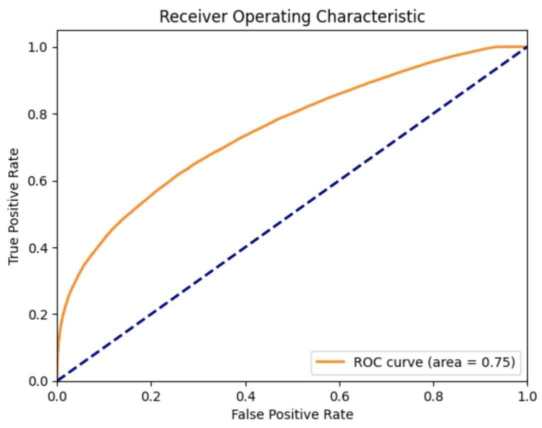
Feature	**Importance scores - Vanilla GAN**	**Importance scores - Wasserstein GAN**
x	0.3578	0.4191
y	0.3272	0.3647
z	0.0711	0.0491
p_x	0.1604	0.0102
p_y	0.0361	0.0183
p_z	0.0473	0.1386

The high importance associated with position in the x and y, seen in Table 1, suggests that the boosted decision tree relies on these two features heavily to make a distinction between the real and generated samples. Both iterations of the Generative Adversarial Networks (GANs) exhibit challenges in accurately modeling distributions that are highly peaked. This observation is derived from the characteristics of the feature distributions under examination, particularly noting that the distributions of the x and y features are significantly more peaked in comparison. To further confirm the problematic nature of the x and y position distributions, they were removed from the model and the new ROC curves and AUC scores were assessed.

Figure 4



(a) ROC curve produced (with removal of x and y features) from boosted decision tree evaluation at 6000 epochs from the Vanilla-GAN.



(b) ROC curve produced (with removal of x and y features) from boosted decision tree evaluation at 5750 epochs from the Wasserstein-GAN.

The lower AUC scores in both cases, visible in Figure 4 is confirmation that both models are limited by their ability in generating the x and y distributions. The lower AUC score for the Vanilla GAN indicates that it was able to generate higher fidelity data than the Wasserstein GAN. It is reasonable to hypothesise that in both cases, the models suffer from mode collapse. In order to try and mitigate the hypothesised mode collapse various types of feature engineering were implemented to try and increase the spread of the x and y position distributions. This was done under the assumption that the generator in both models is better at learning from distributions with a larger spread. These transformations may be inverted after training in order to visualise the true distributions.

3.2 Feature Engineering to Mitigate Mode Collapse

A number of different feature engineering techniques were implemented to both GAN models so as to either increase the spread of the x and y features or smooth their peak. *Kernel Density Estimation* (KDE) was implemented from `sklearn.neighbors`[15] module to smooth the x and y distributions. This technique was chosen to address the sharp peaks and reduce the impact of outliers in the distributions, potentially contributing to mode collapse. By adjusting the bandwidth parameter, optimal balance was sought that would sufficiently smooth the data while preserving essential characteristics of the distribution.

Contrary to expectation, the application of KDE did not yield the anticipated improvement. Instead, it resulted in distributions characterised predominantly by peaks and troughs, deviating significantly from the original data's nature. This outcome suggests that the smoothing process inadvertently emphasised the existing modes while suppressing the intermediate data points, leading to an exaggerated alternation between high and low-density regions. Such a transformation was counterproductive, as it further complicated the generative models' task of accurately capturing the underlying data distribution.

Given the symmetrical nature of the x and y features, transforming the data into *spherical coordinates* presented a novel approach to re-contextualising the features' relationships. By converting Cartesian coordinates (x, y) into their radial (r) and angular (θ) components, the aim was to facilitate a more holistic learning of the spatial distributions by the generators. While spherical transformations provided a unique perspective on the data, the improvement in distinguishing real from generated samples was minuscule, evidenced by high AUC scores. The resulting θ distribution was jagged with a sharp central peak which posed the generator similar issues as before.

A *Gaussian filter* was applied directly to the x and y features. This process involves replacing the value of each data point with a weighted average of its neighbors, where the weights decrease according to a Gaussian distribution as the distance from the center point increases.[9] The primary goal of this smoothing was to reduce the impact of the sharp peaks and noise, thereby facilitating the generators' learning process. Min-max scaling was still applied to the other features. AUC scores decreased in both the Vanilla and Wasserstein GAN with the lowest AUC scores being 0.77 and 0.87 respectively. This represents a large improvement on both original models and confirms both GANs work better learning from less peaked distributions.

3.3 Importance score guided training GAN performance

Leveraging the discriminative power of XGBoost importance scores, our Generative Adversarial Network (GAN) adopts a refined training methodology detailed in subsection 2.6. The resultant training duration averaged 45.17 seconds per 250 epochs over 11,000 iterations, which is a significant increase to previous models due to more frequent classifier evaluations.

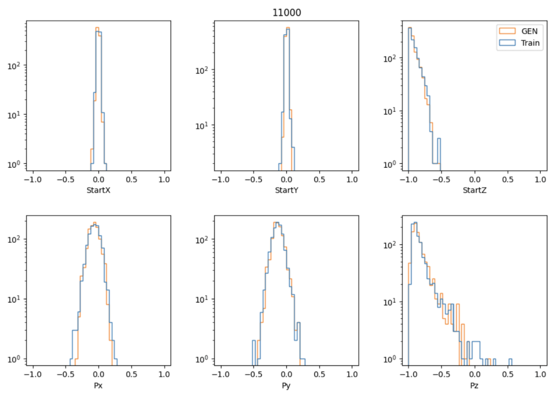
The incorporation of XGBoost importance scores into the GAN training framework was evaluated through the ROC curve AUC scores. Results show an improvement in fidelity of the muon distributions. This approach managed to obtain a AUC score of 0.72 after being trained for 11000 iterations. The ROC curve and the relevant distributions can be seen in Figure 5. These findings suggest that such targeted interventions in the training process can yield some dividends in the quest for high-fidelity generative models.

The importance scores evaluated at 11000 epochs (seen in Table 2) offer interesting insights on the current limitations of the enhanced GAN. These importance scores are vastly different from the ones in Table 1. The importance scores are more evenly distributed and suggest that the x and y distributions are not as impactful on the classifiers discernment process. This suggest that the model is more limited in its general generation ability rather than its ability to generate a specific distribution.

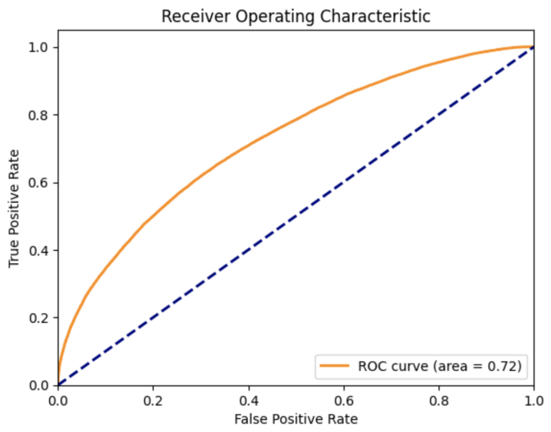
3.4 GAN training efficiency and convergence comparison

Efficiency in training is a crucial aspect of developing and deploying Generative Adversarial Networks (GANs), especially in applications requiring rapid prototyping or iterative experimentation. This section presents a comprehensive comparison of the training speeds of the traditional Vanilla GAN, the Wasserstein GAN and the enhanced Vanilla GAN incorporating XGBoost importance scores along with the time for the each GAN to converge.

Figure 5



(a) Muon feature distributions produced by the Vanilla GAN with importance score enhanced training(11000 epochs).



(b) ROC curve produced by the Vanilla GAN with importance score enhanced training(11000 epochs).

Table 2: Feature Importance Scores for XGBoost-evaluated at 11000 epochs (Enhanced Vanilla GAN)

Feature	Importance scores - enhanced Vanilla GAN
x	0.0624
y	0.1259
z	0.2086
p_x	0.2326
p_y	0.1928
p_z	0.1774

The total training time for each GAN model was measured from the start to the end of the training process, spanning 10,000 epochs. This measurement began the moment the training loop was initiated and concluded once the model completed its final epoch, ensuring that the entire duration of the model's training was captured. The total training time is reported in seconds, providing a macroscopic view of each model's training efficiency.

Convergence of each GAN model is defined by a stable state criterion based on the discriminator's loss fluctuation. Specifically, a model is considered to have converged when the discriminator's loss does not fluctuate more than 5 percent between epochs, assessed over a rolling window of 5 consecutive epochs. This approach was adopted to account for the inherent noise in the training process and to ensure that the convergence identified reflected a genuine stabilisation in the model's learning process rather than transient fluctuations. To implement this, the discriminator's loss was recorded at the end of each epoch. A comparison was then made between the loss at the current epoch and the loss five epochs prior. If the percentage change in loss remained within the 5 percent threshold for five consecutive checks, the model was deemed to have converged. This method allowed for a nuanced assessment of learning efficiency, pinpointing the epoch at which the model began to demonstrate consistent performance. During this assessment the models were each stripped to only produce graphs of the muon feature distributions as any other tasks increased training time. The training performance of the three GAN variants is summarised in Table 3.

Model	Training time - 10000 epochs (seconds)	Convergence Epoch
GAN	439.4	9735
VGAN with Importance score guided training	2002.7	1105
WGAN	492.1	19760

Table 3: Summary of training time and convergence epochs for different GAN models.

The training time data indicate a significant increase in computational overhead when feature importance scores are integrated into Vanilla GANs training process, as evidenced by the training duration of Vanilla GAN with feature importance augmentation compared to the standard Vanilla GAN model. Despite this increase, the augmented Vanilla GAN model achieved convergence significantly faster, suggesting that the incorporation of feature importance scores enhances the model's learning efficiency.

In contrast, WGAN shows a moderate increase in training time over Vanilla GAN but requires a substantially higher number of epochs to converge. This behavior underscores the inherently different training dynamics of Wasserstein GANs, which, while providing theoretical advantages in stability and distribution matching, necessitates longer training periods to achieve optimal performance.

4 Discussion

This study attempts to enhance the Generative Adversarial Network (GAN) architecture for "simulating muons within the SHiP experiment" framework, with a focus on improving the fidelity of generated distributions and optimising the network's architecture for efficiency. Through the implementation of a Wasserstein GAN (WGAN), importance score-guided training, and various feature engineering techniques, an efficiency and fidelity improvement was sought.

Contrary to the anticipated theoretical benefits, the WGAN demonstrated an increase in training time compared to its Vanilla GAN counterpart. This outcome was surprising, given the Wasserstein loss function's purported capability to provide a more stable training process and to effectively mitigate issues such as mode collapse. However, the empirical evidence suggested that not only did the WGAN require a longer time to train, but it also failed to surpass the Vanilla GAN in terms of the Area Under the Curve (AUC) scores from the Receiver Operating Characteristic (ROC) analysis. These higher AUC scores indicate that the classifier found it easier to distinguish between real and generated muons, suggesting a decrease in the fidelity of the generated distributions by the WGAN. One of the primary objectives of incorporating WGAN into this project is to address the prevalent issue of mode collapse. Theoretical insights into WGAN suggest that the Earth Mover's (Wasserstein) distance offers a more meaningful gradient for the generator, which should, in theory, cover the target probability distribution more effectively. However, the persistence of mode collapse in the WGAN model indicates that the theoretical advantages of Wasserstein loss did not translate into practical improvements in our specific application of simulating muon distributions.

A reflection on the architecture of the Wasserstein GAN utilised in this study reveals inherent limitations that may have contributed to its under-performance in simulating muon distributions for the SHiP experiment. Notably, the WGAN model employed is relatively simplistic in design, adhering closely to the foundational principles of Vanilla GANs without extensive customisation for the unique challenges presented by the muon simulation task. The highly specialised and peaked nature of muon distributions may require more intricate network architectures capable of capturing subtle nuances in the data, beyond the architecture used in this project. The observed limitations underscore the necessity for domain-specific customisation of GAN architectures when applied outside their conventional scope. For tasks like muon simulation, where the data does not conform to the patterns typically found in images, a tailored approach that considers the specific characteristics of the data is crucial. This might involve altering layer architectures and training protocol.

The introduction of importance score-guided training into the Generative Adversarial Network (GAN) framework marks a significant methodological innovation in this study. Despite an initial increase in training time, this approach yielded markedly improved results, particularly in terms of model convergence and the fidelity of generated muon distributions. A standout advantage of this methodology is its significantly faster convergence compared to the standard Vanilla GAN and WGAN models. Despite the augmented Vanilla GAN model's longer overall training times due to the computational overhead of integrating importance scores, it achieved operational stability at an earlier stage. This accelerated convergence is indicative of the model's improved efficiency in navigating the feature space, leveraging the guidance provided by the importance scores to make more informed adjustments to the generator. Beyond the practical benefit of faster convergence, the importance score-guided training method demonstrates a qualitative improvement in the generated data's fidelity. The nuanced directionality afforded by this approach resulted in muon simulations that more closely mirrored the complexities of the actual distributions encountered in the SHiP experiment. This alignment suggests that the model was able to capture and replicate the underlying patterns of the muon data with greater accuracy. While the benefits of importance score-guided training are clear, it's important to consider the trade-offs associated with this approach. The increased computational demands of calculating and integrating importance scores into the training loop contribute to longer training times. This factor is particularly relevant in settings where computational resources are limited or where rapid model iteration is necessary. Incorporating importance scores adds a layer of complexity to the GAN training process, increasing the difficulty of model tuning (in the XGBoosted classifier) and optimisation.

5 Conclusions

The integration of feature importance scores into the GAN training process represents a significant methodological innovation. This approach not only introduces a novel direction in the training of GANs but also demonstrates the potential for machine learning techniques to complement and enhance one another. Despite the increased computational demand and longer training times, importance score-guided training leads to faster convergence and improved fidelity of generated distributions. This finding underscores the utility of directing the generative process toward features deemed crucial by an auxiliary classifier, resulting in simulations that more accurately reflect the complexities of the target data. The study highlights a crucial trade-off in GAN training: the balance between computational efficiency and the effectiveness of the generative model. While faster convergence and enhanced fidelity are desirable outcomes, they come at the cost of increased training times due to the computational overhead introduced by calculating and integrating importance scores. The success of this approach in simulating muons for the SHiP experiment suggests its potential applicability across a wide range of domains. Any field requiring high-fidelity simulations could benefit from incorporating feature importance-guided training, especially when dealing with complex or nuanced data distributions.

The findings invite further research into optimising the computational efficiency of integrating importance scores, exploring alternative metrics of feature importance and extending this training methodology to other challenging simulation tasks. Or more generally, a more direct incorporation of the XGBoost classifier into the training process because of its powerful discriminative capacity. Overall, the Wasserstein GAN yielded disappointing results on all assessed metrics however a refinement and research into a Wasserstein GAN with a more tailored architecture would reveal interesting insights into the applicability of Wasserstein GANs in non-image data.

6 Appendices

6.1 Appendix A

Code for all three GANs used in this project can be found via the following GITHUB link: `https://github.com/mpalazzo02/MUON-GAN`

6.2 Appendix B

Abbreviations

- *GAN* - Generative Adversarial Network
- *Vanilla GAN* - Standard GAN framework used in the "Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks " paper
- *WGAN* - Wasserstein Generative Adversarial Network
- *WGAN-GP* - Wasserstein Generative Adversarial Network with Gradient Penalty
- *SHiP* - Search for Hidden Particles
- *CERN* - European Organization for Nuclear Research
- *SPS* - Super Proton Synchrotron
- *VAE* - Variational Auto Encoder
- *ROC* - Receiver Operating Characteristic
- *AUC* - Area Under the Curve
- *XGBoost* - Extreme Gradient Boosting
- *GEANT4* - Geometry and Tracking (simulation software)
- *PYTHIA6* - A computer simulation program for particle collisions
- *KDE* - Kernel Density Estimation

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, et al. tf.Dense — TensorFlow api documentation. `https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense`, 2024. Accessed: [27/03/24].

[2] Martín Abadi, Ashish Agarwal, Paul Barham, et al. tf.random.normal — TensorFlow api documentation. `https://www.tensorflow.org/api_docs/python/tf/random/normal`, 2024. Accessed: [27/03/24].

[3] S. Agostinelli, J. Allison, et al. *Geant4—a simulation toolkit*. *Nuclear Instruments and Methods in Physics Research Section A*, 506(3):250–303, 2003.

[4] C. Ahdida, R. Albanese, et al. *SHiP Experiment - Progress Report*. Technical report, CERN, Geneva, 2019.

[5] C. Ahdida and R. Albanese et.al. *Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks*. *JINST*, 14:P11028, 2019. 7 figures.

[6] Benjamin Bedregal and Ivan Pan. *Some typical classes of t-norms and the 1-Lipschitz Condition*. In *Proceedings of the Ninth Brazilian Symposium on Neural Networks (SBRN'06)*, pages 184–189, 10 2006.

[7] Jason Brownlee. *Loss and Loss Functions for Training Deep Learning Neural Networks*. `https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/`, 2019. Accessed: [28/03/24].

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, San Francisco, CA, USA, 2016. ACM.

[9] Ron Dror. *Image Analysis Notes*. `https://web.stanford.edu/class/cs279/notes/image-analysis-notes.pdf`, 2015. Course notes for CS/CME/BIOPHYS/BMI 279, Stanford University, Fall 2015. [Online; accessed 06/04/24].

[10] Su-Hyun Han, Ko Woon Kim, SangYun Kim, and Young Chul Youn. *Artificial Neural Network: Understanding the Basic Concepts without Mathematics*. *Dementia and Neurocognitive Disorders*, 17(3):83–89, 2018.

[11] Keras. Flatten layer - Keras Documentation. `https://keras.io/api/layers/reshaping_layers/flatten/`, 2024. Accessed: 09-04-24.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017.

[13] Y. Li, Q. Wang, and J. Zhang. *The Theoretical Research of Generative Adversarial Networks: An Overview*. *Neurocomputing*, 2021.

[14] Priyesh Patel, Meet Nandu, and Purva Raut. *Initialization of Weights in Neural Networks*. *Journal of Machine Learning Research*, 4(2):73–79, 02 2019.

[15] F. Pedregosa, G. Varoquaux, et al. *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. *The Earth Mover's Distance as a Metric for Image Retrieval*. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[17] T. Sjöstrand, S. Mrenna, and P. Skands. *PYTHIA 6.4 physics and manual*. *Journal of High Energy Physics*, 2006(05):026–026, May 2006.

[18] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. *VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning*, 2017.

[19] Aston Zhang, Zachary C. Lipton, and Mu Li. *Dive into Deep Learning*. `http://d2l.ai/chapter_multilayer-perceptrons/dropout.html`, 2024. Accessed: 09-04-24.

[20] S. Zhang, Y. Gao, and Y. Jiao. *Wasserstein-Wasserstein Auto-Encoders*. *arXiv preprint arXiv:1902.09323*, 2019.

Certification of ownership of the copyright

Author	Michele Palazzo
Title	Generative neural networks for fast but reliable simulations
Date of submission	18/04/24