Fall 12-2022

Machine Learning-Based Event Generator

Yasir Alanazi
Old Dominion University, alanaziyasir@gmail.com

MACHINE LEARNING-BASED EVENT GENERATOR

by

Yasir Alanazi
B.S. July 2010, Al Jouf University, Saudia Arabia
M.S. April 2017, University of St Thomas

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2022

Approved by:

Yaohang Li (Director)

Mohammad Zubair (Member)

Nikos Chrisochoides (Member)

Nobuo Sato (Member)

ABSTRACT

MACHINE LEARNING-BASED EVENT GENERATOR

Yasir Alanazi
Old Dominion University, 2022
Director: Dr. Yaohang Li

Monte Carlo-based event generators have been the primary source for simulating particle collision experiments for the study of interesting physics scenarios. Monte Carlo generators rely on theoretical assumptions, which limit their ability to capture the full range of possible correlations between particle's momenta. In addition, the simulations of the complete pipeline often take minutes to generate a single event even with the help of supercomputers.

In recent years, much attention has been devoted to the development of machine learning event generators. They demonstrate attractive advantages, including fast simulations, data compression, and being agnostic of theoretical assumptions. However, most of the efforts ignore faithful reproductions, and detector effects due to their complexity and rely on theories for detector simulations.

In this work, we present a new machine learning-based event generator framework free of theoretical particle dynamics assumption. We first create a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN) that selects a set of transformed features to faithfully reproduce simulated and experimental data. Then, we extend FAT-GAN by conditioning the component neural networks according to the given reaction energy and develop a Conditional FAT-GAN (cFAT-GAN) that can generate events at unrelated beam energies.

Next, we implement a conditional folding model that learns the correlations between vertex-level and detector-level events and simulates the distortions produced by the detector machines. The folding model is then integrated into a generator to reconstruct vertex-level events. This serves as a practical framework in a real experimental analysis where such effects must be incorporated.

We finally evaluate different Neural Network architectures and use machine learning techniques for model interpretation and evaluation. In addition, we analyze the GANs latent variables to extract physics resonance regions, illustrating the ability of the developed model to distinguish between the underlying physics mechanisms.

This framework has been validated on simulated inclusive deep-inelastic scattering data

along with the existing parametrizations for detector simulation. The generated results provide a realistic proof of concept for designing a machine learning-based event generator that will be a valuable tool in nuclear and particle physics programs to facilitate the studies of high-energy scattering reactions and understand different physical mechanisms.

Dedicated to my parents, my wife, and my daughters.

ACKNOWLEDGEMENTS

All Praise is Due to Allah (God)

The success of this work would not have been possible without the contributions of many people. I would like first to thank my advisor Dr. Yaohang Li for his guidance, motivation, and support that he has given me over the past five years. I am so grateful for the countless hours he spent with me discussing different research ideas that has broadened my knowledge in several science aspects. I was so fortunate to have Dr. Yaohang as my advisor.

I would like to thank my mentors Dr. Mohammad Zubair and Dr. Nikos Chrisochoides for their endless support and motivation. The feedback I received from both mentors have enlighten me to many aspects related to my research.

I would like to thank the external member: Dr. Nobuo Sato from Jefferson Lab for his valuable contributions and ideas. Dr Nobuo has a significant role in navigating and guiding us to test and validate the work. His expertise in high-energy physics has helped us to have a better understanding of the domain. I am very grateful for the countless hours we spent in weekdays and weekends to discuss several aspects of our work.

I would like also to thank the Jefferson Lab staff: Dr. Marco Battaglieri, Dr. Wally Melnitchouk, Dr. Malachi Schram, Dr. Pawel Ambrozewicz, Dr. Astrid Hiller Blin, and Kishansingh Rajput for their guidance and support and insightful feedback I received through our weekly meetings. I am very grateful for the time we spent to discuss ideas and science problems related to my research.

I would like to thank the computer science department at ODU and Jefferson lab for providing me with computing resources and funds that helped me to conduct this research.

Finally, I would like to thank my family, CS colleagues, and all of my friends for their support and motivation through my journey.

TABLE OF CONTENTS

Page

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

In high-energy nuclear and particle physics, accelerators are utilized to investigate the structure of the atomic nucleus. Particle accelerators also referred to as atom smashers, are machines that use electromagnetic fields to charge particles with energy, such as protons or electrons, to very high speeds, close to the speed of lights. Those particles are then smashed onto a stationary target or against other particles circulating in the opposite direction. The reactions transform the incoming particles into a set of outgoing particles, referred to as physics "events". Detector machines then record the results generated by the collision/interaction, which can be afterward studied by theorists and experimentalists in several scientific fields and applications. Existing particle accelerators include the Large Hadron Collider (LHC) at CERN for proton-proton collisions, Continuous Electron Beam Accelerator Facility (CEBAF) at Jefferson Lab for polarized electron-hadron scattering, Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Lab (BNL) for proton (nucleus)–proton (nucleus) collisions, as well as the future Electron-Ion Collider (EIC).

Besides particle accelerators, there exist software tools called event generators (EGs) that simulate events similar to those produced in particle accelerator labs, and collision experiments. Since the early 1970s, the simulation of EGs have mainly been implemented using Monte Carlo (MC) methods [1]. MC-based event generators (MCEGs) [2] are software libraries constructed by a combination of high-precision data from previous experiments and theoretical inputs, with the help of programming languages, such as C++, and FORTRAN.

Recently, efforts have been made in high-energy physics to construct ML-based event generators (MLEGs) as fast simulation and data compactification tools. MLEGs can be significantly faster than existing MCEGs and can serve as compactified data storage utilities, eliminating the need for maintaining MCEG event repositories. However, Most of these efforts ignore detector effects due to their complexity and they focus on using MLEGs as simulation tools to reproduce physics events with the absence of detector responses.

In this work we present a new strategy for constructing an ML-based event generator using generative adversarial networks (GANs) [3], which have been increasingly utilized recently in high-energy physics applications as a tool for fast simulations [4, 5, 6, 7, 8, 9]. There are several components that are essential to develop a realistic ML-based event

generator. First, we create a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN) [10] that selects a set of transformed features from particle momenta that can be generated easily by the generator, and uses these to produce a set of augmented features that improve the sensitivity of the discriminator. The new FAT-GAN is able to faithfully reproduce the distribution of final state electron momenta in inclusive electron scattering, as well as the complex correlations between particle momenta, without the need for input derived from domain-based theoretical assumptions. We validate the results using inclusive electron-proton scattering events generated from MC-based simulation tool, as well as real experimental data extracted from Jefferson Lab

We also extend the FAT-GAN methodology to a Conditional Feature-Augmented and Transformed Generative Adversarial Network (cFAT-GAN) [11] that can generate events for several energy beams including events of energy beams that are not presented in the training data. This provides the flexibility to test different energy reaction scenarios without the need to train the model multiple times. We test the degree of compatibility of the feature distributions generated by the trained cFAT-GAN against the simulated training data, and also explore the capability of cFAT-GAN to interpolate and extrapolate between various energy beams. We finally analyze the hidden layers of the generator model to examine the sensitivity of the trained architectures with different input energies.

Next, we incorporate smearing detector effects to have more realistic framework, where in FAT-GAN and cFAT-GAN the simulations are performed with the absence of detector effects. For this purpose, we implement a conditional folding GAN that can learn the smearing effects between the events at the interaction point ("vertex-level") to the events measured by the detector machines ("detector-level"). By conditioning the model on vertex-level event features we can enforce learning the correlations between vertex-level and detector-level events as opposed to learning a deterministic mapping between inputs and outputs. This conditional folding GAN is validated on three different scenarios, including toy, inclusive deep-inelastic scattering (DIS), and realistic examples.

The conditional folding procedure is integrated with another GAN model to build a realizable ML-based event generator framework that can faithfully reproduce the phase space of inclusive DIS. For the first time a closure-test for reconstructing vertex-level DIS events, free of theoretical assumptions, is performed.

Finally, we study the effects of different model architectures, and parameters by visualizing the loss landscapes using *filter normalization* technique [12] for several NNs architectures trained on realistic examples. This method helps in model selections and

hyper-parameters tuning and can explain why certain architectures with a set of parameters can perform and generalize better than others. We also analyse the hidden layers of our developed tool to examine the model capability of extracting physics resonance regions using non-linear transformation technique along with ML clustering algorithms.

Our results provide a new opportunity for experimental data analysis to use the GAN approach to build theory-free event generators which mitigate biases induced in reconstructing physical observables from experimental data. This work will be valuable for nuclear and particle physics programs, such as the Continuous Electron Beam Accelerator Facility (CEBAF) and the future Electron-Ion Collider (EIC).

1.1 MOTIVATION

ML-based event generators have been recently utilized in high-energy physics and have the potential to become an alternative approach to existing MC-based event generators. MLEGs present several advantages over existing MCEGs that motivate this work.

First, MLEGs can be significantly faster than MCEGs in generating events. MC simulations of the complete pipeline of particle experiments, including detector effects, often take minutes to generate a single event, even with the support from modern supercomputers [13] and distributed high performance computing platforms. In fact, several accelerator labs, such as LHC, rely on less accurate and simpler event generators because typical MCEGs take a huge amount of time to produce events. In contrast, MLEGs, after proper training, can generate millions of events per seconds. MLEGs can be great candidates to be used for large experiments in particle accelerator labs because of the time they can reduce. For example, at Jefferson Lab [14], they rely heavily on Pythia program to simulate events that can take a significant amount of time to generate a large number of events. Pythia is a MC-based software tool designed by a combination of experimental data and physics laws to simulate the physics processes that can occur in particle collisions between electrons, protons, photons and heavy nuclei [15]. In Table 1., we conduct a simple exercise to compare the events generation in minutes using Pythia that is installed in Jefferson Lab sever with a ML generative model. The ML model is a vanilla GAN [3] that has four dense layers for the generator and the same number of layers for the discriminator, implemented in Keras [16] with TensorFlow backend [17]. As we can see in Table 1., Pythia event generator takes approximately 30 minutes to generate 1M events of inclusive electron-proton scattering events, whereas the GAN needs about half a minute to generate the same number of events. In fact, many physics applications require the generation of big data for the full analysis and

experiments; therefore, it is possible to generate millions or billions of events using MLEGs. For this experiment, we use the same number of batch size = $1,000$ with the CPU specifications in Table 2. to have a reasonable comparison. It is important to mention that this simple exercise can give an overall estimation to compare the time between GANs and Pythia to generate events for a specific application, but more advanced and comprehensive comparison techniques are needed for a fair comparison.

Number of events	Pythia	GAN
10,000	0.192	0.006
100,000	2.262	0.015
1,000,000	30.34	0.621

TABLE 1.: Compare Time in Minutes of Events Generation of Pythia and GAN

Architecture	`x86_64`
CPU op-modes	32-bit, 64-bit
Threads per core	2
CPU family	6
Model name	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
CPU max MHz	3000.0000
CPU min MHz	1200.0000

TABLE 2.: CPU Specifications

Second, MCEGs rely on theoretical assumptions such as factorization and statistical models, which limit their ability to capture the full range of possible correlations existing in nature between particles' momenta. On the other hand, MLEGs are trained directly on experimental data, and can explore the true underlying probability distributions governing the spectra of particles produced in reactions.

Third, in particle accelerators we can only observe the collisions or interactions that the detector machines can observe and produce, and those produced events are called ("detector-level") events; however, the momenta of final state particles that are produced at the interaction point ("vertex-level"), exist only on theory and the distributions of the events are not known. This motivates us to explore the possibility of using ML techniques to reconstruct theory-free vertex-level events.

Finally, most accelerators facilities deploy MCEGs to simulate events collisions, and they invest a great amount of resources to support the full simulation processes. For instance, [18] reports that disk storage is between 10s to 100s of PBs per experiment for LHC at CERN. Hence, in some scenarios, those labs rely on simpler, and less accurate, event generators for their experiments [19]. This motivates machine learning researchers to investigate the possibility of using MLEGs as a data compactification tool due to their capability of significantly reducing the disk space usage. Indeed, a MLEG requires only the weights and architecture of the generator model to be saved which then can be loaded using the model dependency libraries and generate millions of events within seconds.

1.2 RESEARCH QUESTIONS

The research objective of this work is to first design a ML-based tool can faithfully reproduce particle collision events. Second objective is to implement a folding procedure that can fold vertex-level events and produce the corresponding detector-level events. Finally, we aim at building a ML-based event generator framework that integrates the folding algorithm with a ML generator to reconstruct vertex-level events free of any theoretical assumptions about the underlying physical laws of the original system.

The specific research questions can be listed as follow:

RQ1: Can we build a ML-based generator to faithfully reproduce particle collision events?

RQ2: Can we simulate the smearing detector effects using ML tools?

RQ3: Can we build a ML-based event generator framework to reconstruct vertex-level events?

1.3 OUTLINE

The remainder of the dissertation is organized as follows.

Chapter 2 explains some fundamental topics and terminologies related to our research questions, including particle accelerators, event generators, and detector machines.

Chapter 3 reviews the literature related to our research. We first review the ML generative models used to build ML-based event generators as a tool for fast simulation. We then present the simulation of detector effects, and discuss the physics-related challenges in these software tools.

Chapter 4 presents Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN) model that can faithfully reproduce inclusive electron-proton scattering events. This work addresses RQ1, and published in [10].

Chapter 5 extends FAT-GAN developed in the previous chapter to Conditional Feature-Augmented and Transformed GAN that can be trained on multiple energy reactions. This work is related to RQ1 and published in [11].

Chapter 6 presents our methodology to learn detector effects related to RQ2. We present *direct folding* and its extension *conditional folding* model to have a folding procedure that can generalize for several applications.

Having presented the folding procedure in the previous chapter, Chapter 7 integrates the *conditional folding* with a ML-based generator to construct vertex-level events. This chapter addresses RQ2 and published in [20].

Chapter 8 presents the hidden layers analysis using ML dimensionality reduction techniques, and visualizes the loss landscapes of several GANs architectures to explain and interpret the developed technology.

Finally, Chapter 9 concludes and summarizes our contributions and presents the potential future work.

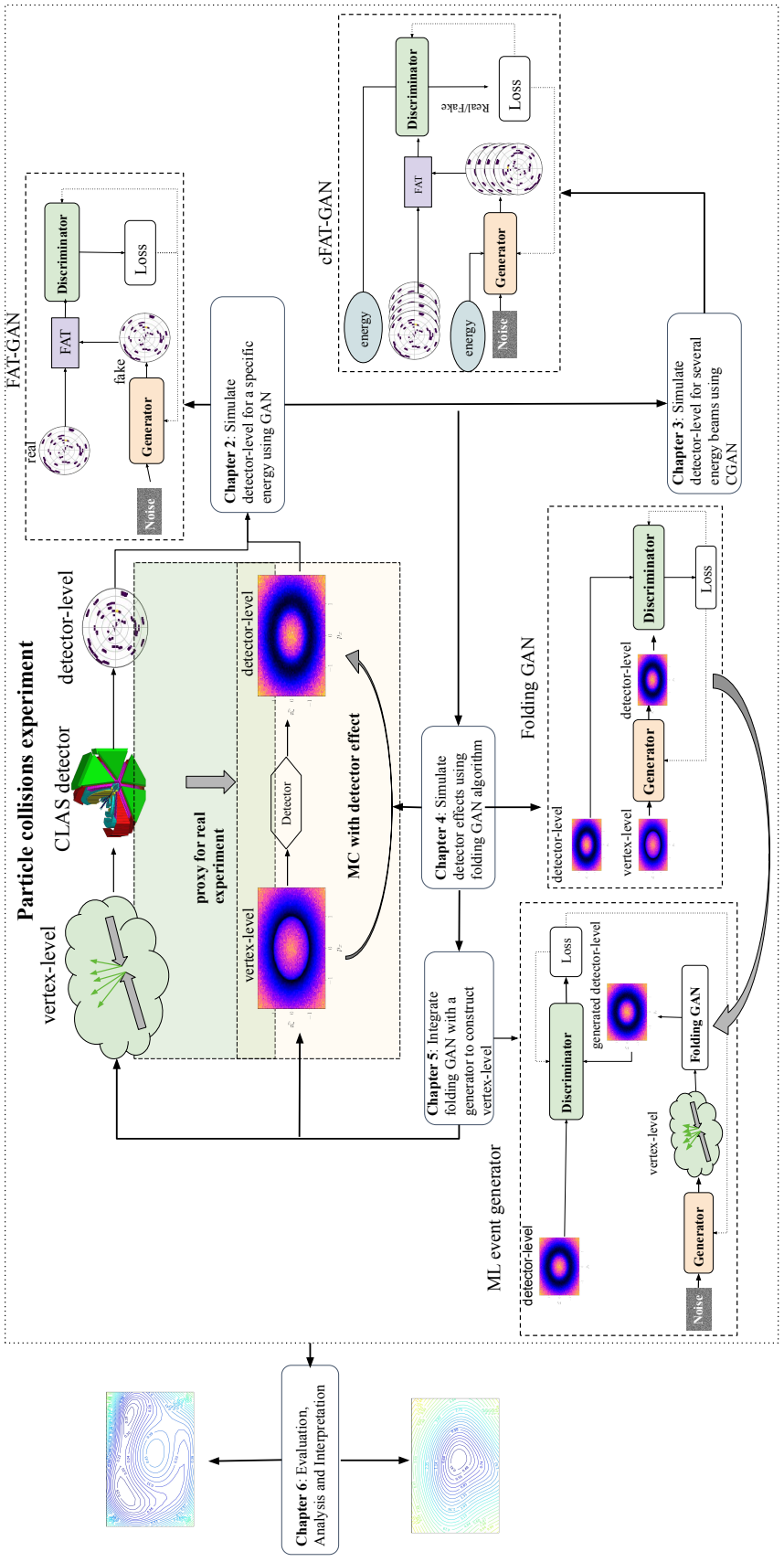


Fig. 1.: Overall diagram shows how each chapter is related to the others.

CHAPTER 2

BACKGROUND

In this chapter we explain some fundamental concepts and tools in particle physics related to our research. We first describe particle accelerators and why they are beneficial. Then, we describe event generators in particle physics which include, MC-based and ML-based event generators. We also introduce detector machines and how they affect the produced particle events, and what are the different detector effects on the pipeline of the collision events reconstructions. Finally, we explain two important concepts which are folding and unfolding.

2.1 PARTICLE ACCELERATORS

Particle accelerators or atom smashers, are machines that use electromagnetic fields to charge particles with energy, such as protons or electrons, to very high speeds, close to the speed of lights. Those particles are then smashed onto a stationary target or against other particles circulating in the opposite direction. In particle accelerator facilities, such as the Continuous Electron Beam Accelerator Facility (CEBAF), LHC, and the Relativistic Heavy Ion Collider (RHI), physicists smash particles against each other to observe the interactions between particles. By studying these collisions, scientists have advanced our life in several scientific applications including modern healthcare, mobile phones, protecting the environments, and many others. For instance, the atoms used in the Positron Emission Tomography (PET) scanners, which is used in the detection and treatment of cancer, are normally produced in a particle accelerator [21]. Another application of great importance is the development of X-ray emitting Free Electron Lasers [22] which can be used to boost security at ports and airports by scanning cargo to detect drugs, or explosives. The study of particle collisions can also help discover new particles and answer questions like what is all matter made of, and how the nature of our universe is structured.

2.2 EVENT GENERATORS

Event generators (EGs) are software tools/libraries that simulate high energy particle physics events such those produced in particle accelerators labs. We can classify EGs into two main categories: Monte Carlo-based event generators (MCEGs), and ML-based event

generators (MLEGs). The first are widely used in high energy physics and they are still the most accurate tools being used in particle physics studies. The latter is a current research that emerged with the revolution of machine learning generative models. In fact, the use of MLEGs are rabidly growing and they have the potential to be alternative methods to existing MCEGs.

2.2.1 MONTE CARLO-BASED EVENT GENERATORS

MC-based event generators are designed using predefined physics rules and theories. Since the early 1970s, the simulation of physics events has mainly been implemented by MC methods, which transform random numbers into simulated physics events. MCEGs are constructed by a combination of experimental data as well as theoretical inputs defined by physics rules, such as the conservation of momentum, and the conservation of energy. Most of the MCEGs were written in C++ and FORTRAN and other programming languages. These software libraries have been valuable tools in facilitating the studies of high-energy scattering reactions. Commonly used MCEGs include Pythia [15], Herwig [23], and Sherpa [24] for hadronic events; MadGraph [25] and Whizard [26] for parton events; GiBUU [27] and HIJING [28] for heavy-ion events; GENIE [29] and NuWro [30] for neutrino events, as well as specialized event generators such as AcerMC [31], ALPGEN [32], and other software tools.

2.2.2 MACHINE LEARNING EVENT GENERATORS

Instead of simulating physics events from first principles as in MCEGs, MLEGs employ a data-driven approach to learn from event samples. ML generative models, including GAN [3], Variational Autoencoder (VAE) [33], and Normalizing Flows (NF) [34], have been adopted to implement MLEGs.

While MLEGs are being investigated, there exist several obstacles and challenges. Here we list some of the main challenges and they will be discussed in details in 3.3:

- How to correct data from detector effects?
- How to conserve physics laws and rules in the generated results?
- How to deal with sharp edges, spikes, and multiple peaks exist in the distributions?
- How to reproduce physics events with good agreement?

2.3 DETECTOR MACHINES

Due to the invisibility of particles, scientists are limited to reconstruct the collision events from the particles trajectory and the trace they leave after the smashing. This is possible with the help of detector machines which are complex machines installed within the particle accelerators that act as cameras to record the collisions. These detectors can measure the momenta of particles, and how much energy each particle has, and identify the type of particles as well as tracking their traces. There are several particle detectors around the world, such as A Toroidal LHC ApparatuS (ATLAS) which is the largest detector at the LHC, A Large Ion Collider Experiment (ALICE), and many other detectors.

2.3.1 THE EFFECTS OF DETECTORS

Detectors machines have a significant impact on the collision events. We classify detector effects into three categories: acceptance, smearing, and misidentification related.

Smearing Effects

An intrinsic inability of detectors to measure physics quantities with an infinite accuracy, referred to as finite resolution, contributes to measurement uncertainty. The smearing represents the difference between the reconstructed four-momentum of a particle and that at the vertex. It not only depends on the resolution of the detector, but also the materials in the trajectory and the magnetic field. Multiple scattering, which occurs during the passage of particles through different media from which the experimental apparatus is made, also affects the physics quantities' distributions by changing trajectory directions to the point that they may go out of the acceptance and not be observed at all. The materials and the magnetic field in the trajectory of a particle may trigger radiations, which will cause energy loss of the particle, and multiple scatterings, which will significantly alter the particle's momentum and may produced secondary particles that have nothing to do with the physics at the vertex. Although these effects will change the momentum of the particle produced at the vertex, the information collected by detectors still inherits a major part at the vertex. It is possible to reconstruct the four-momentum at the vertex with some uncertainty once these effects are properly taken into account.

Acceptance Effects

The experimental setup itself introduces a detection volume and acceptance, which is usually quite complicated. The acceptance describes whether a particle produced at the vertex with certain four-momentum is detected by the detector system. Sometimes the detection does not require all detectors in the system have signals, but collected information must be enough to reconstruct the four-momentum within the required accuracy as well as the particle type. Since detectors in high-energy experiments only cover part of the phase space, one can never observe all particles produced at the vertex, and information about undetected particles will be missing.

Misidentification Effects

This effect, which arises from the finite resolution, means some information collected by the detector could be wrong. One example is to identify the particle type, which depends on the efficiency of the separation of different type of particles. One can never be 100% sure about the particle type if only based on the detection of a particle.

2.3.2 FOLDING AND UNFOLDING

According to [35], the true distribution $f(t)$ of a physics variable t (vertex-level) to be measured in particle physics is not directly accessible. Because of finite resolution of the detector machines, the distribution $g(d)$ of the measured variable d is related to the unknown distribution $f(t)$ by distortions and transformations. Using MC methods the direct process ("folding") from an assumption $f(t)^{model}$ on the true distribution $f(t)$ to the expected measured distribution $g(d)$ can be simulated. However, the inverse process from the actually measured distribution $g(d)$ to the related true distribution $f(t)$ is difficult and ill-posed. In particle physics this problem is usually called ("unfolding").

$$folding\ process\ (MC): \qquad true/MC\ dist.\ f(t) \rightarrow g(d)\ measured\ dist.$$

$$unfolding\ process\ (inverse): \qquad measured\ dist.\ g(d) \rightarrow f(t)\ true\ dist.$$

Similar to deconvolution or restoration [36], which are used to solve the inverse problem[37], in particle physics unfolding represents the process to infer an unknown distribution from the measured data, using knowledge and assumptions of the distortions. Therefore, unfolding is to recover vertex-level data given detector-level data, and folding is to use MC $f(t)^{model}$ to simulate the detector distortions.

CHAPTER 3

RELATED WORK

In this chapter we first review the ML efforts using several generative models to build ML-based event generators as a tool for fast simulation. Then we describe the simulation of detector effects, and discuss the physics-related challenges in these software tools as well as methods of incorporating physics into ML models. Most of our literature review can be found in our published paper in [38].

3.1 MACHINE LEARNING-BASED EVENT GENERATORS

In this section, we review the state-of-the-art of ML efforts at building physics event generators. We review different ML generative models used in MLEGs and their specific challenges, and discuss various approaches of incorporating physics into the ML model designs to overcome these challenges.

Instead of simulating physics events from first principles as in MCEGs, MLEGs employ a data-driven approach to learn from event samples. ML generative models, including Generative Adversarial Networks (GANs) [3], Variational Autoencoders (VAEs) [33], and Normalizing Flows (NFs) [34], have been adopted to implement MLEGs, and these algorithims will be discussed in details in the next subsections.

3.1.1 GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) are the most popularly used generative models in MLEGs. The regular GAN event generator is composed of two neural networks: a generator G and a discriminator D. The former is trained to generate fake event samples, and the latter is a binary classifier to distinguish the events of the generated distribution P_G from the true events with distribution P_T. G and D are trained under the value function $V(D,G)$

$$\begin{aligned}\min_G \max_D V(D,G) = &\langle \log D(x) \rangle_{x\sim P_T} \\ &+ \langle \log(1 - D(\tilde{x})) \rangle_{\tilde{x}\sim P_G}.\end{aligned} \tag{1}$$

MLEGs	Data Source	Detector Effect	Reaction/Experiment	ML Model
[8]	Pythia8	DELPHES + pile-up effects	$Z \to \mu^+\mu^-$	regular GAN
[39]	MadGraph5 aMC@NLO	DELPHES3	$e^+e^- \to Z \to l^+l^-$, $pp \to t\bar{t}$	VAE
[9]	MadGraph5 aMC@NLO		$pp \to t\bar{t} \to (bq\bar{q}')(\bar{b}\bar{q}q')$	MMD-GAN
[19]	MadGraph5, Pythia8	DELPHES + FASTJET	$2 \to 2$ parton scattering	GAN+CNN
[40]	Pythia8 + GEANT4		Search for Hidden Particles (SHiP) experiment	regular GAN
[10] [11]	Pythia8		electron-proton scattering	MMD-WGAN-GP, cGAN
[41]	Pythia8	DELPHES particle-flow	proton collision	GAN, cGAN
[42]	Sherpa		$pp \to W/Z + n$ jets	NF
[43]	MadGraph5 + Pythia8	DELPHES	$Z \to e^+e^-$	SWAE
[44]	MadGraph5 + Pythia8	DELPHES	$pp \to b\bar{b}\gamma\gamma$	WGAN-GP

TABLE 3.: List of Existing MLEGs

As shown in the original GAN paper [3], given an optimal discriminator $D^* = P_T(x)/(P_T(x)+P_G(x))$, training G becomes identical to minimizing the Jensen-Shannon divergence (JSD)

$$\min_G V(D^*, G) = -2\log 2 + \mathrm{JSD}(P_G||P_T). \tag{2}$$

If JSD becomes 0, then $P_G = P_T$.

Although GANs have demonstrated success in many applications, training a successful GAN model is known to be notoriously difficult [45]. Many GAN models suffer from major problems including mode collapse, non-convergence, model parameter oscillation, instability, vanishing gradient, and overfitting due to unbalanced generator/discriminator combinations. Studies [39, 8] also reported a less satisfactory performance when a regular GAN is used for event generation.

Several improved GAN architectures have been employed in MLEGs to enhance GAN training:

- **Least Squares GAN (LS-GAN)**: LS-GAN [46] replaces the cross entropy loss function in the discriminator of a regular GAN with a least square term

$$\begin{aligned} \min_D V(D) = &\frac{1}{2}\langle (D(x)-b)^2\rangle_{x\sim P_T} \\ &+ \frac{1}{2}\langle (D(G(\tilde{x}))-a)^2\rangle_{\tilde{x}\sim P_G}, \\ \min_G V(G) = &\frac{1}{2}\langle (D(G(x))-c)^2\rangle_{x\sim P_G}. \end{aligned} \tag{3}$$

  As a result, by setting $b-a=2$ and $b-c=1$, minimizing the loss function of LS-GAN yields minimizing the Pearson χ^2 divergence. The main advantage of LS-GAN is that, by penalizing the samples far away from the decision boundary, the generator is pushed to generate samples closer to the manifold of the true samples.

- **Wasserstein GAN (WGAN)**: WGAN [47] used Wasserstein or Earth-Mover's distance [48] to replace JSD in the regular GAN. Under Kantorovich-Rubinstein duality, the Wasserstein distance is defined as

$$W(P_G, P_T) = \max_{w\in W} \langle f_w(x)\rangle_{x\sim P_T} - \langle f_w(G(\tilde{x}))\rangle_{\tilde{x}\sim P_G}, \tag{4}$$

  where f is a family of K-Lipschitz continuous functions, f_w, parameterized by w in parameter space W. Instead of directly telling fake events from the true ones, the discriminator or (critic) in WGAN is trained to learn a K-Lipschitz continuous function to minimize the Wasserstein distance. Compared to JSD, Wasserstein distance

provides a meaningful and continuous measure of the distance between the event distribution from the generator and the true event distribution, even when they have no overlaps, which helps guide the training of the generator toward the true event distribution and reduce the likeliness of mode collapse.

- **Wesserstein GAN Gradient Penalty (WGAN-GP)**: A problem in WGAN is to use weight clipping to maintain K-Lipschitz continuity of f_w during training, which still results in unstable training, slow convergence, and gradient vanishing. WGAN-GP [49] replaces weight clipping with gradient penalty to ensure K-Lipschitz continuity and thus further improve WGAN stability. The gradient penalty is calculated as

$$\lambda\langle(||\nabla_{\hat{x}} f_w(\hat{x})||_2 - 1)^2\rangle_{\hat{x}\in p_{\hat{x}}}, \tag{5}$$

  where parameter λ balances the Wasserstein distance and gradient penalty, and $p_{\hat{x}}$ is uniformly sampled along lines between event pairs from P_T and P_G.

- **Maximum Mean Discrepancy GAN (MMD-GAN)**: MLEGs are particularly concerned about the precise matching between the generated and the true event distributions, where MMD-GAN [50] can be used to enhance the matching precision. MMD-GAN incorporates an MMD term to the generator loss function:

$$\begin{aligned} \mathrm{MMD}^2(P_G, P_T) &= \langle k(x, x')\rangle_{x,x'\sim P_G} \\ &+ \langle k(y, y')\rangle_{y,y'\sim P_T} - 2\langle k(x, y)\rangle_{x\sim P_G, y\sim P_T}, \end{aligned} \tag{6}$$

  where $k(.)$ is a kernel function. MMD typically works well in low dimension; however, the power of MMD degrades with dimension polynomially [51].

3.1.2 AUTOENCODER

Another generative model used in MLEGs is VAE, in this section we give a brief background of Autoencoder (AE) to get more insights of the origin of VAEs and how we can generalize AE to be a class of generative models. AE is an unsupervised machine learning algorithm learns how to encode original input data into lower dimensional space using a NN called *encoder*, denoted by Ψ, then learns how to reconstruct the original data from the lower dimensional space to a representation that is similar to the original input data using another NN called *decoder*, denoted by Φ. The goal of the AE is to select Ψ and Φ functions that have the minimal error to reconstruct the training data. The loss function

used to train an AE is called *reconstruction loss*, that is a comparison of how well the output has been reconstructed from the original input. The *reconstruction loss* of a typical AE can be defined as,

$$\mathcal{L}_{\text{AE}} = \|x - \Phi(\Psi(x))\|^2 \tag{7}$$

which takes the difference between the original training data and the reconstructed one, and there are many metrics can be used, such as MSE or any other functions depending on the input and output of the training data.

Variational Autoencoders

While AE learns a function to map each input from the training data to a number and then learns the reverse mapping, Variational Autoencoder (VAE) learns probability distribution of the input data by adding a layer that estimates a mean and a standard deviation of the hidden (bottleneck) layer, and then sample from the estimated parameters to reconstruct the input data. VAE generalize the AE idea by not only learning the embeddings, but also how to generate new samples by sampling from the estimated parameters A VAE, composed of an encoder network Ψ and a decoder network Φ, is an alternative generative model employed in MLEGs. In MLEGs using a VAE, Ψ projects the events onto latent variables z, and Φ reconstructs the events from z, while z is forced to follow a standard normal distribution. Then, the loss function of the VAE is motivated by variational inference [52] via minimizing the Kullback–Leibler divergence (KLD) between the posterior $p(z|x)$ and the encoded prior distribution $q(z) = \mathcal{N}(0, 1)$:

$$\mathcal{L}_{\text{VAE}} = \|x - \Phi(\Psi(x))\|^2 + \eta \text{KLD}(q(z)||p(z|x)), \tag{8}$$

where the first term is the reconstruction error, the second term computes KLD, and η is the harmonic parameter to balance the two.

The VAE can be further improved as a Wasserstein Autoencoder (WAE) [53] by replacing the KLD term with Wasserstein distance in the loss function, for a similar reason as for the GAN:

$$\mathcal{L}_{\text{WAE}} = \|x - \Phi(\Psi(x))\|^2 + \eta' W(q(z), p(z|x)), \tag{9}$$

where η' is the harmonic parameter. [43] adopted a Sliced Wasserstein Autoencoder (SWAE) [54] for their MLEG, using a sliced Wasserstein distance to approximate $W(q(z), p(z|x))$.

Conditional Variational Autoencoder

For structured output predictions, Conditional VAE (CVAE) was proposed by [55] to make diverse predictions for different input samples. The objective function of the VAE can be modified by adding the variable c,

$$\mathcal{L}_{\text{CVAE}} = \|x - \Phi(\Psi(x))\|^2 + \eta \text{KLD}(q(z,c)||p(z|x,c)) \tag{10}$$

where we just conditioned all of the distributions with c. CVAE is an extension of VAE by adding the conditional part in the *encoder* and *decoder* to associate the input samples with labels, so at inference time we have more control to generate samples that belong to specific labels in contrast to VAE that does not have control over the generated samples.

3.1.3 NORMALIZING FLOWS

Without adopting adversarial learning, the NF is another generative model that has been used in MLEGs. The fundamental idea underlying NFs is the change of variables in probability functions. Under some mild conditions, the transformation can lead to complex probability distributions of the transformed variables. NFs use an invertible mapping (bijection) function f, often implemented as a neural network, to transform a distribution of $x \in \mathbb{R}^D$ into $y \in \mathbb{R}^D$. The transformed probability density function $q(y)$ becomes

$$q(y) = p(x) \left| \det \left\{ \frac{\partial f}{\partial x} \right\} \right|^{-1}. \tag{11}$$

With a series of mappings $f_1 \dots f_k$, an NF is obtained:

$$x_k = f_k \circ \cdots \circ f_1(x_0), x_0 \sim q_0(x_0). \tag{12}$$

The NF is able to transform a simple distribution into a complex multi-modal distribution, and has demonstrated success in collider physics simulations [42].

3.1.4 EXISTING MACHINE LEARNING EVENT GENERATORS

Table 3. lists the existing MLEGs. In the literature, GANs, VAEs, NFs, and their various improved architectures have been used to simulate physics events from different reactions and training datasets. Both [39] and [56] reported that the LS-GAN yields better performance than other generative models, not only in terms of better precision, but also that in the explored scenarios they were faster. However, at this point, it is too early to

rule out the optimal generative model architecture for general MLEGs, which requires not only computational verification, but also rigorous theoretical justifications.

3.2 DATA CORRECTION FROM DETECTOR EFFECTS

Any event generator, whether an MCEG or MLEG, that attempts to faithfully reproduce a specific reaction channel must take into account not only the primary interaction at its vertex but also the interactions of the emergent particles with materials and with the devices detecting them. The former should take into account energy losses, decay, new particle production, as well as multiple scattering effects, while the latter should carefully model detector responses to particles being detected. The experimental setup introduces a detection volume, or acceptance, which is usually quite complicated. The acceptance covers only a portion of the phase space of the reaction, and has to be modeled employing MC packages, such as GEANT4 [57], DELPHES [58], FLUKA [59], or similar.

We classify detector effects into three categories: acceptance, smearing, and misidentification related. All these effects are mitigated in the MLEGs by using well designed procedures that allow either to remove or to introduce these effects into the synthetic data. The former is known as "unfolding", and the latter as "folding." These procedures usually involve training GANs with additional information introduced by modifying loss functions to improve stability and convergence [7], using fully conditional GAN (FCGAN), where the conditioning is done on the detector response [60], or employing Wasserstein distance based loss function in a conditional GAN (WGAN) framework [61].

3.3 PHYSICS-RELATED CHALLENGES

Compared to many applications employing machine learning generative models to produce images, music, and arts, using MLEGs to simulate events from particle reactions poses new additional physics-related challenges for machine learning:

i Events generated by MLEGs should not violate physical laws, such as energy and momentum conservation;

ii MLEGs for generating particle physics events are required to model the distributions of event features and their correlations sufficiently precisely for the nature of particle reactions to be correctly replicated;

iii The distributions of events exhibit natural, physics driven patterns, such as discrete attributes, prominent and narrow peaks, or symmetric behavior of certain physical

quantities. On top of that they also exhibit artificial, detector-related, patterns, such as acceptance induced holes and gaps, and efficiency based regions of lower particle occupancy, which complicate MLEGs;

iv The outgoing particles, with increasing incident energy, will yield increased dimensionality of the emergent products.

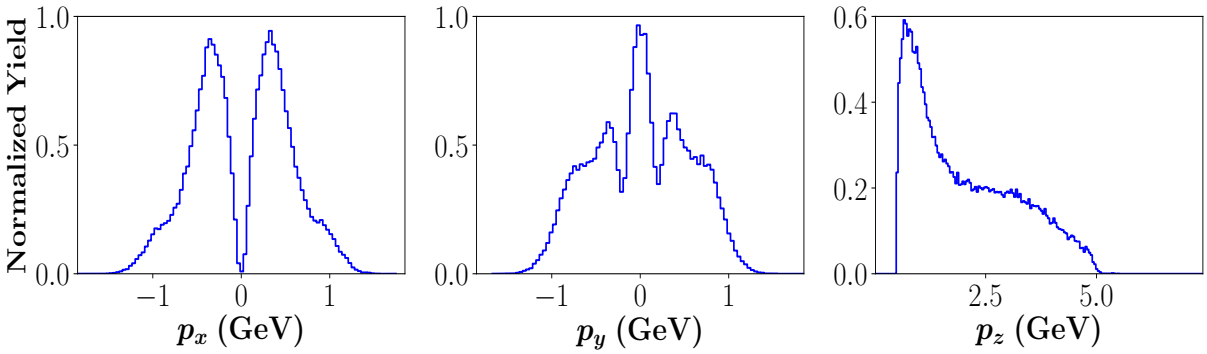


Fig. 2.: Distributions, where Sharp Peaks, Holes, and Steep Edges are Observed.

It is important to note that all existing MLEGs listed in Table 3. are trained using simulated data generated from MCEGs, such as Pythia, MadGraph, and Sherpa. When learning from real experimental data, it adds an additional level of complications to the MLEGs. Figure 2. shows the momenta plots of experimental data from an electron scattering experiment with the CEBAF Large Acceptance Spectrometer (CLAS) detector at Jefferson Lab. Due to the configuration of superconducting coils of the torus magnet, spaced by angles of 60°, the detector packages [62] shown in Fig. 3. are accordingly divided into six sectors, so that events which fall into the coils are not detected, leaving six gaps in any particle transverse momentum components plot, p_x and p_y, as in Fig. 4.. As a result, the 1D plots of p_x, p_y, and the remaining longitudinal component, p_z, as shown in Fig. 2., yield spikes, deep holes, and sharp edges, which pose difficulty for MLEGs to precisely learn their inherent physical laws as well as the detector patterns.

3.4 INCORPORATING PHYSICS INTO ML MODELS

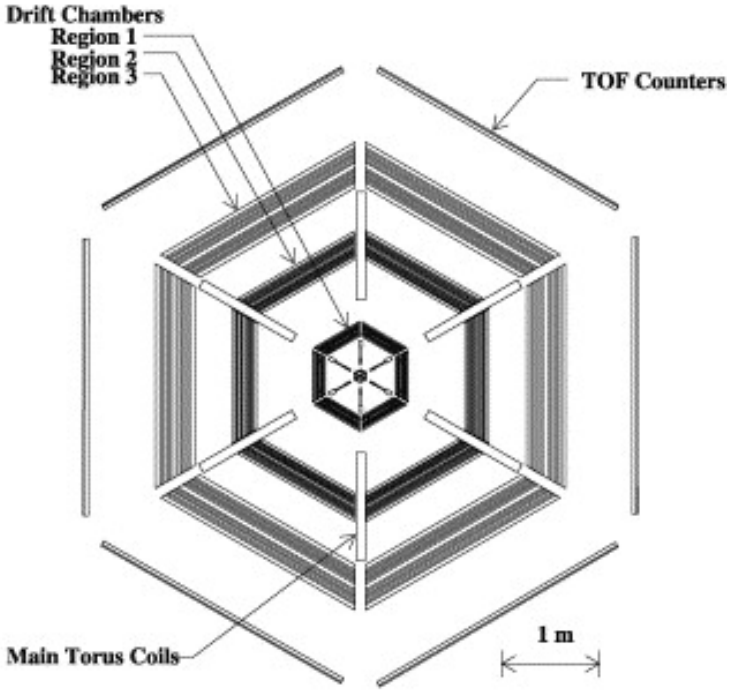


Fig. 3.: CLAS Detector (Downstream View).

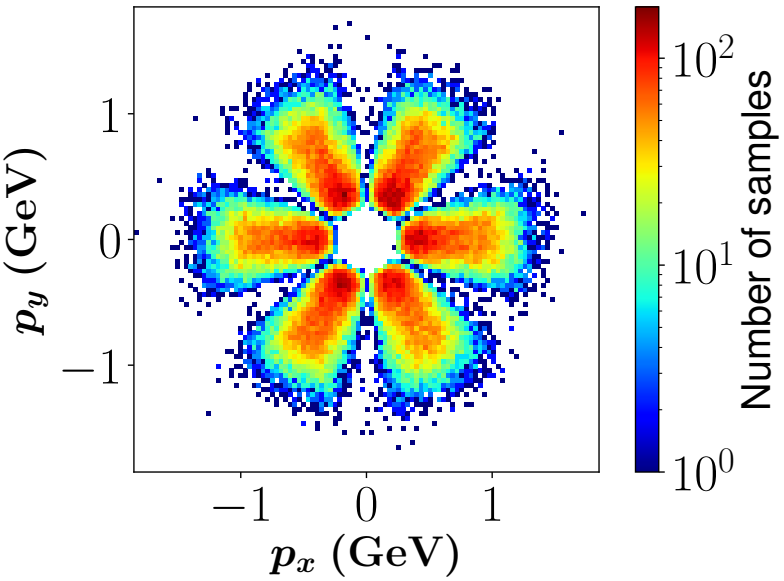


Fig. 4.: Detector Configuration Reflections.

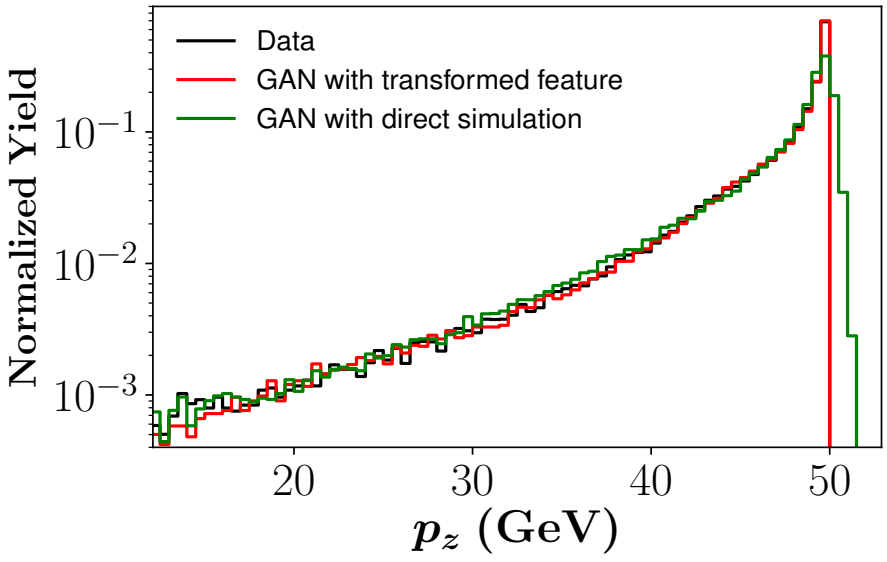


Fig. 5.: Comparison of the p_z distributions generated by the GAN with the transformed features (red), and the direct simulation GAN (green), and the true distribution from Pythia (black).

To address the above physics-related challenges, an important approach is to incorporate physical laws into the generative models. Appropriately encoding physical laws into the models can reduce the degrees of freedom of the problem and improve the performance of MLEGs. One way to incorporate a physical law into the generative models is via feature engineering. In our work of simulating inclusive scattering of electrons [10], we found that a direct simulation GAN generates unphysical events that violate energy conservation. As shown in Fig. 5., a sharp edge in the particle energy E distribution arises from energy conservation, which restricts E to be less than the input beam energy, $E_{\rm b}$. This sharp edge is difficult to learn for the inclusive GAN, whose output is the electron momentum 3-vector (p_x, p_y, p_z), as unphysical events can be generated with $E > E_{\rm b}$, which the discriminator is not sensitive enough to differentiate from the eligible physics events, particularly when $E_{\rm b} - E$ is small. In Chapter 4 we elaborate more on this and show how to prevent the ML model from generating unphysical events.

Another approach to incorporate physics into generative models is to make the latent variables physically meaningful. Typically, the noise fed to GAN or the latent variables in VAE follow certain well-known, easy-to-generate distributions such as Gaussian or uniform

without physics meanings. [43] expressed the latent distributions in SWAE with quantum field theory and thus sampling the latent space became efficient and was able to infer physics.

3.5 DISCUSSION

Compared to MCEGs that have been developed for over 50 years, MLEGs are still in their infant stage, bringing a lot of anticipation as well as many challenges and questions currently without clear answers. In this section, we discuss two questions regarding the applications of MLEGs.

3.5.1 CAN MLEGS FAITHFULLY REPRODUCE PHYSICS?

Whether MLEGs can fully represent the underlying physics of a reaction and faithfully reproduce physical events is critical to many applications where MLEGs are proposed. In the existing MLEGs listed in Table 3., the agreement between the events generated by MLEGs and the sample events is mostly measured in 1D using χ^2 or other statistical metrics. In practice, one is often interested in the correlations among event feature distributions. Another issue is related to the rare events occurring in the reaction, which are often precisely those of significant physics interest. Such rare events pose a difficult challenge to MLEGs, however, which often bias to more frequent events during their training.

The methods of incorporating physics into MLEGs, as described in Sec. 3.4, help improve the precision of MLEGs. Augmenting the generated features of MLEGs to other important physics properties of interest, as described in [20], also increases the sensitivity of the discriminator in the GAN event generator, and thus enhances the quality of the generated events. Nevertheless, at this point, there is lack of a comprehensive evaluation framework to thoroughly evaluate the quality of MLEG events in comparison with those from MCEGs or from experiment, particularly in quantifying the correlation among event features with physics meaning, as well as measuring the quality of the rare events.

3.5.2 CAN MLEGS PROVIDE NEW PHYSICS INSIGHTS?

Can an MLEG go beyond the manifold of its training event data and bring physical insight into regions without any training (experimental) data? We start the discussion of this question from the extrapolation capability of neural networks. A major drawback of a neural network is its difficulty with extrapolation. The theoretical explanation is rooted in the "universal approximation property" of a feed-forward neural network [63], *i.e.*, a

neural network can approximate any continuous function and many discontinuous functions by adjusting its parameters with respect to its training samples. Therefore, for the space outside of the range of the training samples, the output of a neural network is not reliable. MLEGs trained using GANs, VAEs or NFs are fundamentally neural networks, which thus inherit the extrapolation problem. [11] showed that an MLEG based on cGAN yields good agreement for interpolating events between training beam energy levels, but not as good agreement for extrapolating events beyond training beam energy levels, particularly for those related to beam energies further from the training beam energies.

There are two potential ways to extend MLEGs to generate correct events in unknown regions. One way is to apply regularization, forcing the generator to adopt a simple model to limit the degrees of freedom of its neural network and avoid overfitting. More specifically, incorporating known physical laws into the regularizers helps generalize the neural networks and reduce their variation to explore the unknown physical laws and theories. The other way is to generate artificial data samples within the unknown region by either physics theory or simulation to correct the behavior of MLEGs. Both approaches can also be combined to allow MLEGs, at least at some extent, to extrapolate.

3.6 CONCLUSION

In this literature review section, we review ML generative models used in existing MLEGs, including GANs, VAEs, NFs, and their enhanced architectures. Some studies reported that LS-GAN yields better event quality over other generative architectures, but these lack theoretical justifications. MLEGs, as practical tools to simulate physical events, pose additional challenges related to physics, and we review methods of incorporating physics into MLEGs to address these challenges. We further explore the two questions on the capability of MLEGs on faithful reproduction of physics, and extrapolation. Answers to these open questions will have significant impact on the applications of MLEGs.

Unlike many generative model applications, such as producing sharp looking images or fancy objects, where the distribution agreement between the generated samples and the truths is often not strictly enforced, the general requirements underlying MLEGs are to precisely reproduce a specific target distribution. The generative models developed in MLEGs, which incorporate domain knowledge into the machine learning algorithms to faithfully generate samples mimicking complex target distributions, have the potential to be applied to broader applications, such as bioinformatics [64] and cosmology [65].

CHAPTER 4

FEATURE-AUGMENTED AND TRANSFORMED GAN

In this chapter we present our Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN) that was published in [10]. In this work we intentionally ignore the detector effects and investigate the possibility of faithfully reproducing simulated and experimental events with the absence of detector effects. The major contribution of the FAT-GAN is to incorporate physics into the ML model that significantly improves the results. We begin the discussion in Sec 4.1 with an overview of FAT-GAN. Then, we describe the data used to validate this work, and explain the model architectures. After that, we discuss the problem of generating nonphysical events when we directly use the training features as inputs and demonstrate our approach to solve this issue. Finally, we show the results on simulated and experimental events, and investigate how the generated results can capture complex correlations between the training features. This chapter addresses **RQ1:** Can we build a ML-based generator to faithfully reproduce particle collision events?.

4.1 INTRODUCTION

We apply a Wasserstein GAN (WGAN) [49] to build a MLEG that simulates particle production in electron-proton scattering that is free of theoretical assumptions about underlying particle dynamics. The difficulty of efficiently training a MLEG lies in learning the complicated patterns of the distributions of the particles physical properties. Our model selects a set of transformed features from particle momenta that can be generated easily by the generator, and uses these to produce a set of augmented features that improve the sensitivity of the discriminator. The implemented FAT-GAN model is able to faithfully reproduce the distribution of final state electron momenta in inclusive electron scattering, without the need for input derived from domain-based theoretical assumptions.

4.2 DATA DESCRIPTIONS

We use GANs to mimic two samples of inclusive electron-proton scattering events: the first is generated from the Pythia MCEG [15] at a center-of-mass energy of 100 GeV, and the second is experimental data from CLAS [66] at Jefferson Lab with beam energy of 5.5 GeV. In our initial analysis, we train the GAN only on the scattered electron momenta. Events

are represented as an array of the electron four-momenta $p_\mu = (E; \boldsymbol{p})$, where in Cartesian coordinates the three-momentum is given by $\boldsymbol{p} = (p_x, p_y, p_z)$, and the energy is given by $E = \sqrt{p_x^2 + p_y^2 + p_z^2 + m^2}$, where m is the electron mass. Throughout this paper we will work in units of GeV for all momentum and energy variables.

4.3 MODEL ARCHITECTURE

The generator produces both generated features and augmented features. The three generated features describe the three degrees of freedom of the scattered electron (components of the momentum $\boldsymbol{p}$), while the augmented features, such as energy E, transverse momentum $p_T = \sqrt{p_x^2 + p_y^2}$, and the longitudinal to transverse momentum ratio p_z/p_T, can be calculated from the generated features to represent other physical characteristics of the particle. The augmented features are used to improve the sensitivity of the discriminator.

The input to the generator "G" is a 100-dimensional white noise array centered at 0 with unit standard deviation. The generator network consists of 5 hidden dense layers, each with 512 neurons, activated by a leaky Rectified Linear Unit (LReLU) function. The last hidden layer is fully connected to a three-neuron output, activated by a linear function representing the generated features. A customized Lambda layer is then incorporated to calculate the augmented features from the generated features. Inspired by the idea of importance sampling [67], the augmented features are carefully selected to improve the sensitivity of the discriminator in distinguishing the GAN-generated events from the Pythia input. These augmented features, together with the generated features from the generator, are concatenated and fed to the discriminator as input. As for the generator, the neural network in the discriminator "D" also consists of 5 hidden dense layers, each with 512 neurons, activated by a LReLU function. To avoid overfitting in classification, a 10% dropout rate is applied to each hidden layer. The last hidden layer is fully connected to a single-neuron output, activated by a linear function, where "1" indicates a true event and "0" is a fake event. The overall architecture of the inclusive GAN event generator is illustrated in Fig. 6..

4.3.1 LOSS FUNCTION

The discriminator D is trained to give $D(\boldsymbol{p}) = 1$ for each sample $\boldsymbol{p}$ generated by Pythia, and $D(\widetilde{\boldsymbol{p}}) = 0$ for each sample $\widetilde{\boldsymbol{p}}$ produced by the generator. The discriminator is optimized against the Wasserstein loss with gradient penalty [49] to improve training stability and reduce the likeliness of mode collapse. The loss function L_D of the discriminator is defined

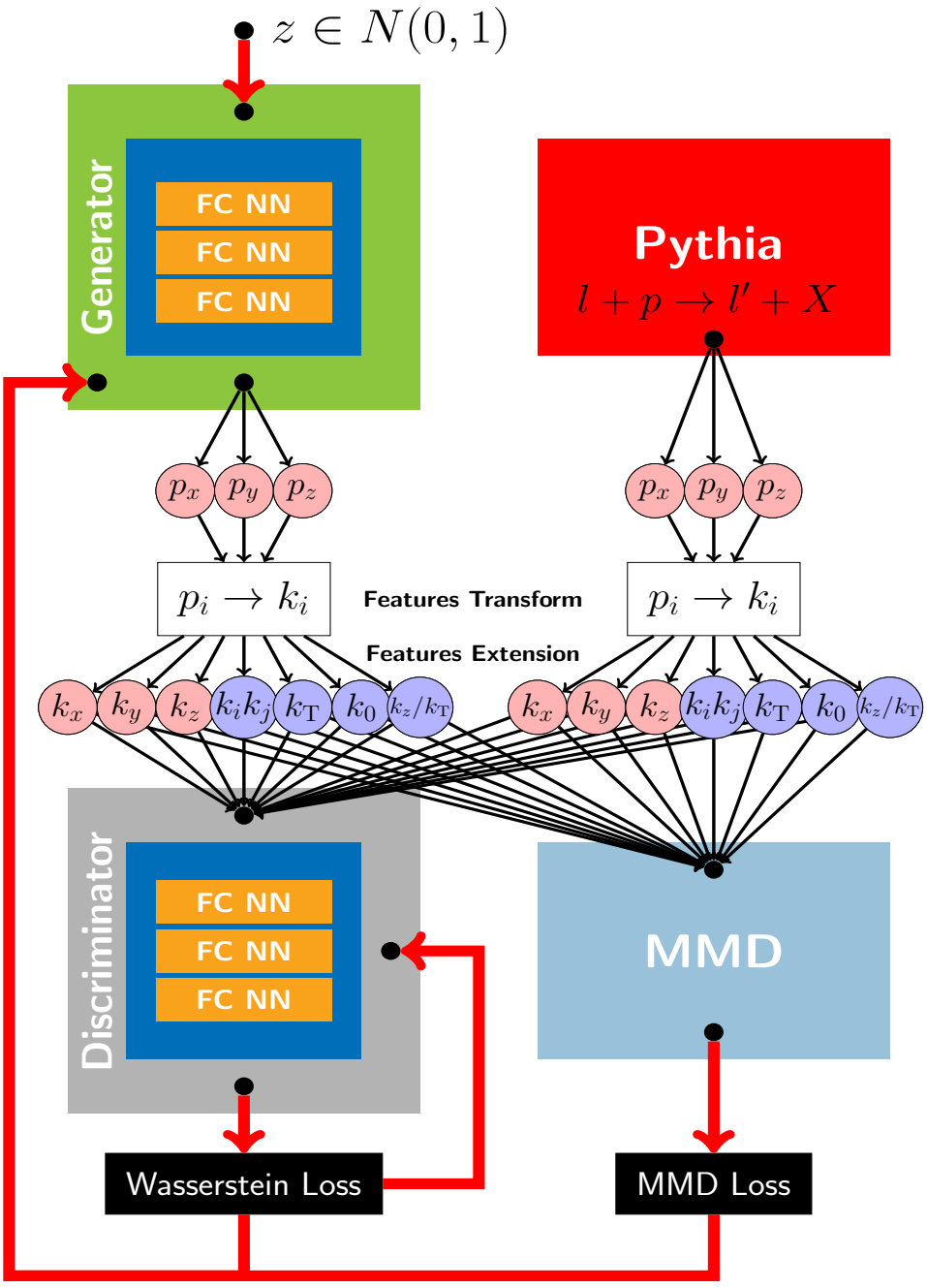


Fig. 6.: Architecture of the Inclusive FAT-GAN Event Generator.

as

$$L_D = \big(\mathbb{E}[D(\widetilde{\boldsymbol{p}})] - \mathbb{E}[D(\boldsymbol{p})]\big) + \lambda\, \mathbb{E}_{\widehat{\boldsymbol{p}}\sim P_{\widehat{\boldsymbol{p}}}}\big[(\|\nabla_{\widehat{\boldsymbol{p}}} D(\widehat{\boldsymbol{p}})\|_2 - 1)^2\big], \tag{13}$$

where $\mathbb{E}$ denotes the expectation value. The first term in L_D measures the Wasserstein distance [47]. The second term is the gradient penalty, where $\widehat{\boldsymbol{p}}$ is a random sample from $P_{\widehat{\boldsymbol{p}}}$, defined by a uniform distribution along the straight lines between pairs of samples from the true Pythia event data and the generator's output. The coefficient λ is a harmonic parameter to balance the Wasserstein distance and the gradient penalty.

To ensure that the distributions of the event features created by the generator match well with those of Pythia, we incorporate a two-sample test based on kernel MMD in our inclusive GAN event generator. To compare two distributions, the MMD employs a kernel-based statistical test methods to determine if the two samples are drawn from different distributions. As a result, the loss function L_G of the generator G includes a Wasserstein distance term from the discriminator D and an MMD term [50],

$$L_G = -\mathbb{E}[D(\widetilde{\boldsymbol{p}})] + \eta\, \mathrm{MMD}^2(\boldsymbol{p}, \widetilde{\boldsymbol{p}}), \tag{14}$$

where η is the balancing hyperparameter. The MMD term is defined as

$$\begin{aligned} \mathrm{MMD}^2(\boldsymbol{p}, \widetilde{\boldsymbol{p}}) = {} & \mathbb{E}_{\boldsymbol{p}_a, \boldsymbol{p}_{a'}\sim P_{\boldsymbol{p}}}[k(\boldsymbol{p}_a, \boldsymbol{p}_{a'})] \\ & + \mathbb{E}_{\boldsymbol{p}_b, \boldsymbol{p}_{b'}\sim P_{\widetilde{\boldsymbol{p}}}}[k(\boldsymbol{p}_b, \boldsymbol{p}_{b'})] \\ & - 2\, \mathbb{E}_{\boldsymbol{p}_a\sim P_{\boldsymbol{p}}, \boldsymbol{p}_b\sim P_{\widetilde{\boldsymbol{p}}}}[k(\boldsymbol{p}_a, \boldsymbol{p}_b)], \end{aligned} \tag{15}$$

where $k(\boldsymbol{p}_a, \boldsymbol{p}_b)$ is a positive definite kernel function. Here we select a Gaussian kernel such that $k(\boldsymbol{p}_a, \boldsymbol{p}_b) = \exp[-(\boldsymbol{p}_a - \boldsymbol{p}_b)^2/2\sigma^2]$, where σ is the hyperparameter determining the MMD resolution, tuned to the same order of magnitude as the width of the event features.

4.4 FEATURES REPRESENTATIONS

The physical observables characterizing the scattered electron properties are illustrated in Fig. 7., including the momentum and energy components of the electron four-vector p_μ [Fig. 7.(a)–(e)], scattering angles [Fig. 7.(f)–(g)], and derived quantities x_{bj} and Q^2 (see Sec. 4.5.2 below). The energy and momentum distributions generated by Pythia exhibit rather large variations, with the ratio between the most populated regions and those with rare events reaching up to $\sim 10^4$.

More seriously, a sharp edge in the E distribution arises from energy conservation, which restricts E to be less than the incident beam energy, E_{b}. This sharp edge is very difficult

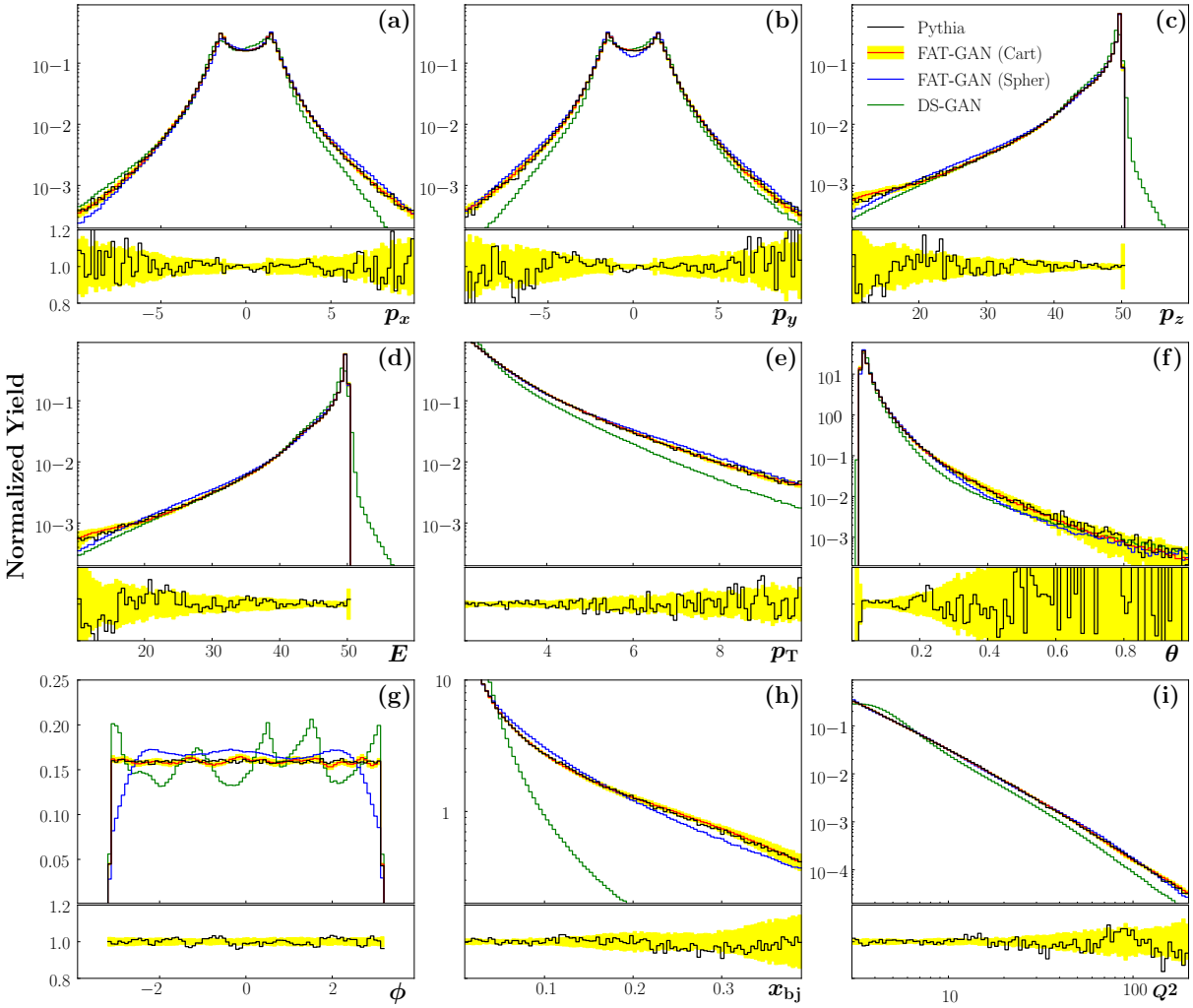


Fig. 7.: The distributions of the physical properties p_x, p_y, p_z, E, p_T, θ, ϕ, $x_{\rm bj}$ and Q^2 of the scattered electron (see text) generated by Pythia (black lines), FAT-GAN (Cart) (red lines and yellow bands), FAT-GAN (Spher) (blue lines), and DS-GAN (green lines). The FAT-GAN (Cart) is in good agreement with Pythia. All momentum and energy variables are in units of GeV, and the angles θ and ϕ are in radians.

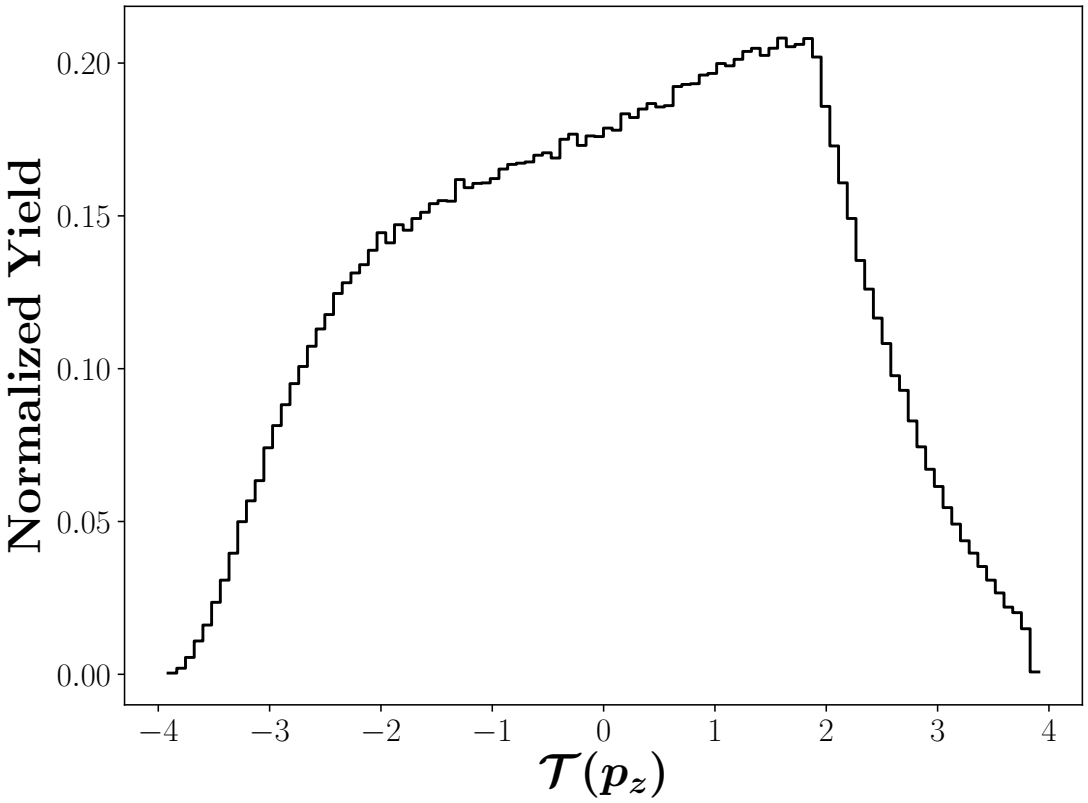


Fig. 8.: Distribution of the Transformed Feature $\mathcal{T}(p_z)$.

for the inclusive GAN to learn, as unphysical events can be generated with $E > E_{\rm b}$, which the discriminator is not sensitive enough to differentiate from the eligible physical events, particularly when $E_{\rm b} - E$ is small. The sharp edge in the E distribution also leads to sharp edges in the p_z and θ distributions. The difficulty of learning sharp edge distributions has also been reported in [8].

To address the problem of learning sharp edge distributions, we transform the momentum properties to specific generated features that allow their distributions to be generated more easily, while avoiding production of unphysical particles. For the p_z distribution, instead of directly using p_z as a generated feature, we use the transformed variable $\mathcal{T}(p_z) = \log[(E_{\rm b} - p_z)/(1\ {\rm GeV})]$ in the generator. The original distribution with a sharp edge is now converted to a distribution that is more like a Gaussian, with significantly reduced variation, as illustrated in Fig. 8..

Although $\mathcal{T}(p_z)$ does not have actual physical meaning, the transformation ensures that

the GAN will not generate events with unphysical p_z values. As well as making it easier for the generator to produce, the transformed distribution $\mathcal{T}(p_z)$ improves the sensitivity of the discriminator as a classifier. As a result, the generator learns to generate $(p_x, p_y, \mathcal{T}(p_z))$, and $p_z = \exp[(E_\mathrm{b} - \mathcal{T}(p_z))/(1\ \mathrm{GeV})]$ is later calculated by the customized Lambda layer as one of the augmented features.

In addition to p_z, the variables p_T, E, and p_z/p_T are also calculated in the customized Lambda layer as augmented features in order to improve the sensitivity of the discriminator. The generated features and augmented features are concatenated as the input to the discriminator.

4.5 RESULTS

In this section we compare the event feature distributions from the FAT-GAN and the Direct-Simulation GAN ("DS-GAN") that directly simulates the momenta (p_x, p_y, p_z). We also compare the efficiency of the FAT-GAN implementation in Cartesian coordinates ("FAT-GAN (Cart)") with the FAT-GAN in spherical coordinates ("FAT-GAN (Spher)"). The latter is an alternative representation to describe a particle in terms of the variables (E, θ, ϕ), where $\theta = \arctan(p_z/p_T)$ is the polar angle between p_z and the transverse plane, and $\phi = \arctan(p_y/p_x)$ is the azimuthal angle. The FAT-GAN (Spher) generates ($\mathcal{T}(E)$, $\mathcal{T}(\theta)$, ϕ) in spherical coordinates as the generated features, and then p_x, p_y, p_z, p_T, E, and p_z/p_T as the augmented features. The features $\mathcal{T}(E)$ and $\mathcal{T}(\theta)$ are converted from E and θ using a logarithmic transformation similar to that applied to p_z to remove the sharp edges in the distribution.

4.5.1 FEATURE DISTRIBUTIONS

The event feature distributions from the DS-GAN, FAT-GAN (Cart), and FAT-GAN (Spher) are compared in Fig. 7. with those generated from Pythia. In general the DS-GAN results do not match as well with Pythia compared to FAT-GAN (Cart) and FAT-GAN (Spher). Moreover, the tails beyond the sharp edges in Fig. 7.(c) and (d) indicate that some unphysical events are generated with $E > 50$ GeV. The FAT-GAN (Cart) yields match better with Pythia than do the FAT-GAN (Spher) yields, particularly for the ϕ distribution. The four momentum components (E; p_x, p_y, p_z), as well as p_T, θ and ϕ, are also well reproduced relative to Pythia, and have a minimal number of unphysical events.

Compared to the reaction events simulated in [9] and [19], where the typical ratio between the peak and tail events is up to 10, the ratio in our scattering electron features can be up to

10^4. Nevertheless, even for the rare events that are 10^{-3} of the number of the peak events, the FAT-GAN (Cart) agrees well with Pythia in their distributions, including the symmetry of ϕ shown in Fig. 7..

4.5.2 FEATURES INTER-CORRELATIONS

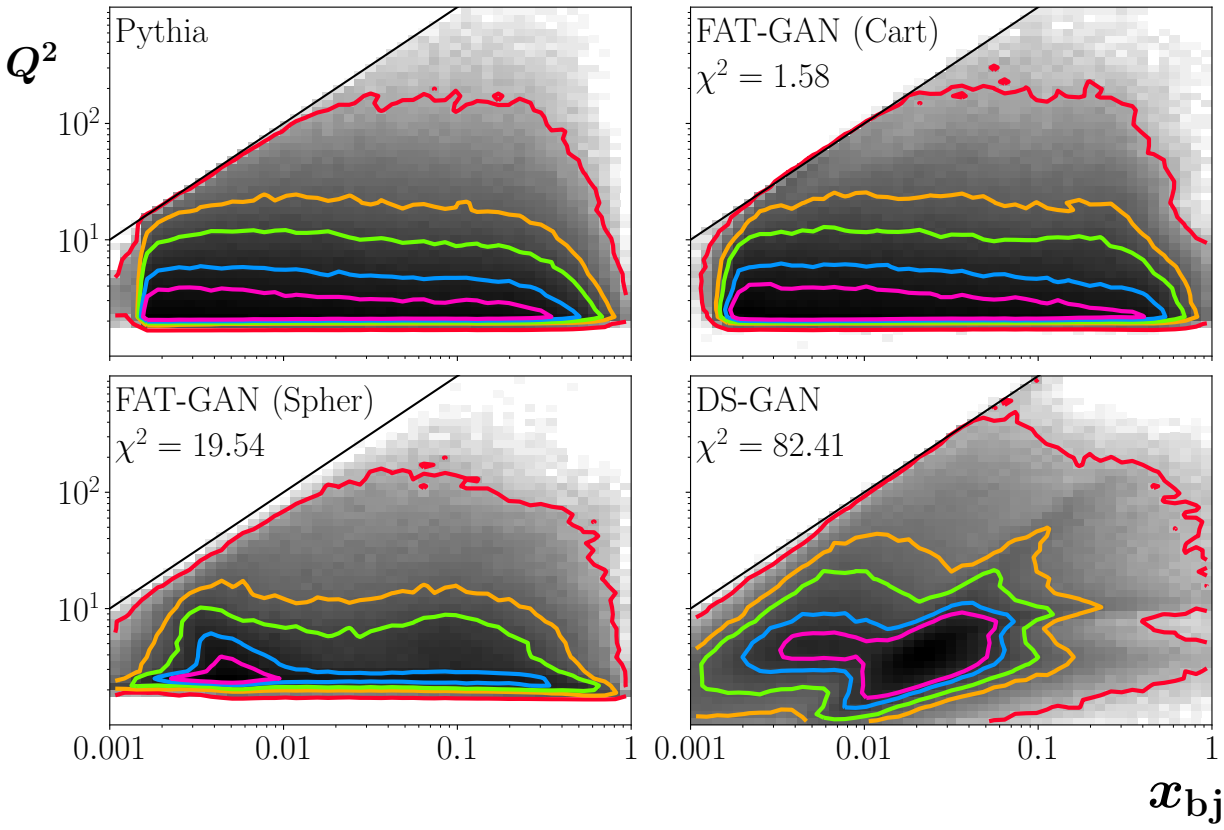


Fig. 9.: Joint distributions of Q^2 and x_{bj} for FAT-GAN (Cart), FAT-GAN (Spher) and DS-GAN compared to Pythia. The χ^2 values per number of bins are indicated in the GAN-generated panels.

In addition to the electron momentum and energy, we examine two additional physical quantities that are typically used to characterize electron scattering, namely the squared four-momentum of the exchanged virtual photon Q^2 and the Bjorken scaling variable x_{bj},

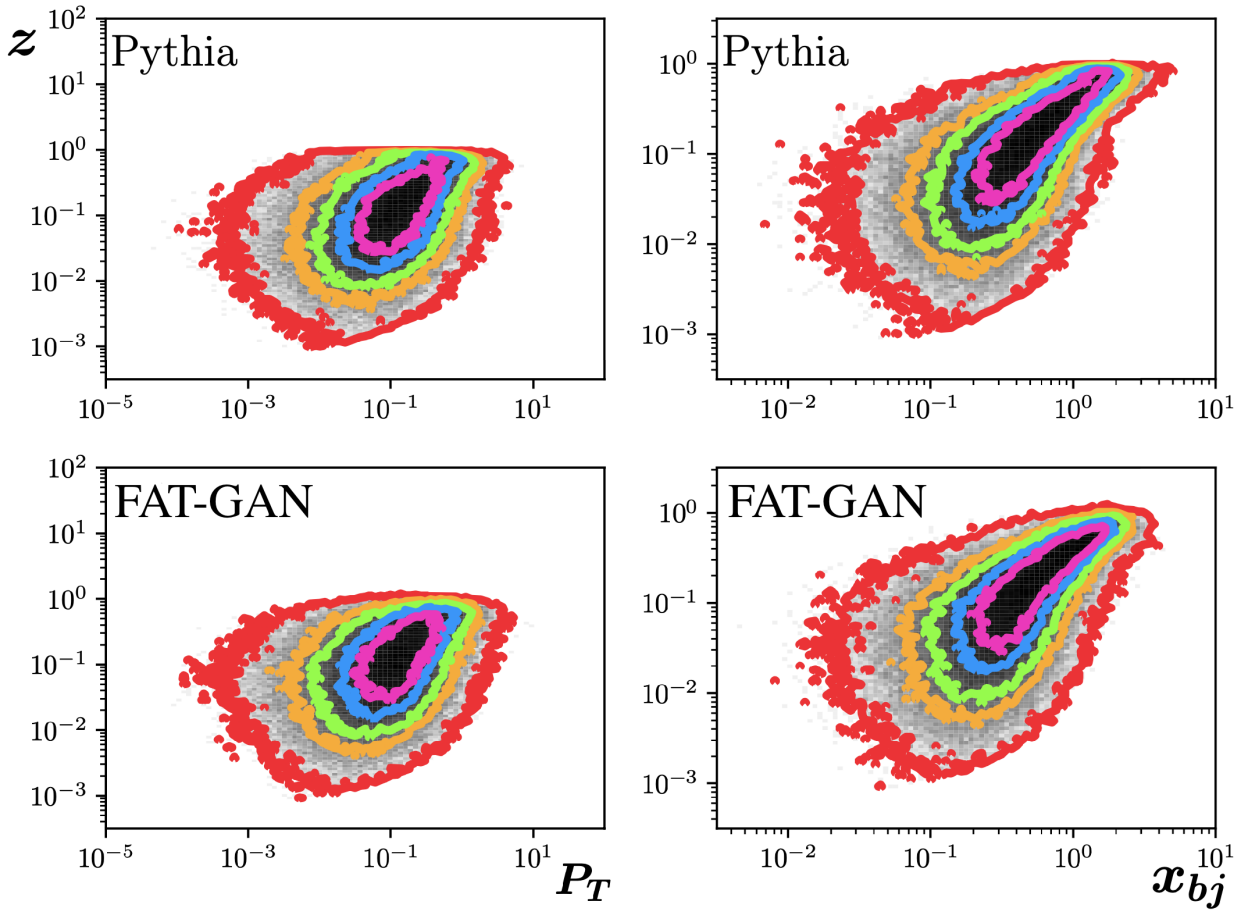


Fig. 10.: Joint distributions of P_T and Z, and x_{bj} and Z for FAT-GAN (Cart) (lower row), compared to Pythia (top row).

neither of which are explicitly generated as features in the FAT-GAN. In terms of the beam and scattered electron momenta and energies, the photon virtuality can be written as

$$-Q^2 = q \cdot q \equiv q_0^2 - \boldsymbol{q}^2, \tag{16}$$

where $q_0 = E_{\mathrm{b}} - E$ and $\boldsymbol{q} = (-p_x, -p_y, \sqrt{E_{\mathrm{b}}^2 - m^2} - p_z)$ are the energy and three-momentum transfer, respectively, and the " $\cdot$ " symbol denotes the dot product operation in Minkowski spacetime. The Bjorken variable is defined to be the dimensionless ratio

$$x_{\mathrm{bj}} = \frac{Q^2}{2P \cdot q}, \tag{17}$$

and kinematically ranges from 0 to 1. As shown in Fig. 7.(h) and (i), the x_{bj} and Q^2 distributions from the DS-GAN deviate significantly from those generated by Pythia. In

contrast, the FAT-GAN (Cart) yields match better than those from the DS-GAN and the FAT-GAN (Spher).

We further examine the Q^2–$x_{\rm bj}$ joint distributions, as shown in Fig. 9.. The FAT-GAN (Cart) yields a good match with the Pythia Q^2-$x_{\rm bj}$ joint distribution, as indicated by the contour lines, with a χ^2 value per number of bins of 1.58. In contrast, somewhat worse agreement is observed for the FAT-GAN (Spher), with $\chi^2 = 19.54$, and very poor agreement for the DS-GAN, with $\chi^2 = 82.41$. This indicates that our FAT-GAN model in Cartesian coordinates not only learns the four-momentum vector accurately, but also their inter-correlations. Similarly, we examine the $x_{\rm bj}$–P_z and P_t–P_z. We can see the FAT-GAN (Cart) (lower row) shows a good agreement with the Pythia (top row).

4.5.3 COMPARISON OF COORDINATE REPRESENTATIONS

The Cartesian representation and the spherical representation of a particle are equivalent physically. However, the Cartesian and spherical representations exhibit different learning efficiencies in the FAT-GAN. As shown in Figs. 7. and 9., the FAT-GAN (Cart) demonstrates better agreement in the distributions of physical properties than the FAT-GAN (Spher). In Fig. 11. the convergence of the FAT-GAN (Cart), FAT-GAN (Spher) and DS-GAN is compared, as measured by the χ^2 values for the $x_{\rm bj}$ distributions of Pythia and GAN-generated events along training epochs. Although both FAT-GANs yield better convergence than the DS-GAN, the FAT-GAN (Cart) demonstrates better efficiency and generally lower χ^2 values that are close to 1 than the FAT-GAN (Spher).

Overall, the FAT-GAN (Spher) is found to have a degraded performance compared to the FAT-GAN (Cart). The main reason is that the physical properties in spherical coordinates are less favorable than those in Cartesian coordinates for training the FAT-GAN. In particular, both the distributions of E and θ in spherical coordinates exhibit sharp edges (see Fig. 7.(d) and (f)). Moreover, the ϕ distribution has shape boundaries at both ends, which poses additional complications for the GAN to learn.

4.5.4 FAT-GAN ON EXPERIMENTAL DATA

Having studied the performance of the FAT-GAN on synthetic Pythia data, we now compare the results of the FAT-GAN (Cart) simulations with inclusive scattering data for a 5.5 GeV electron beam impinging on a liquid hydrogen target in CLAS at Jefferson Lab [**?**]. A clean event sample of 100k reconstructed electron events was used for the comparison with the trained FAT-GAN, illustrated in Fig. 12., which shows that the distributions of the events

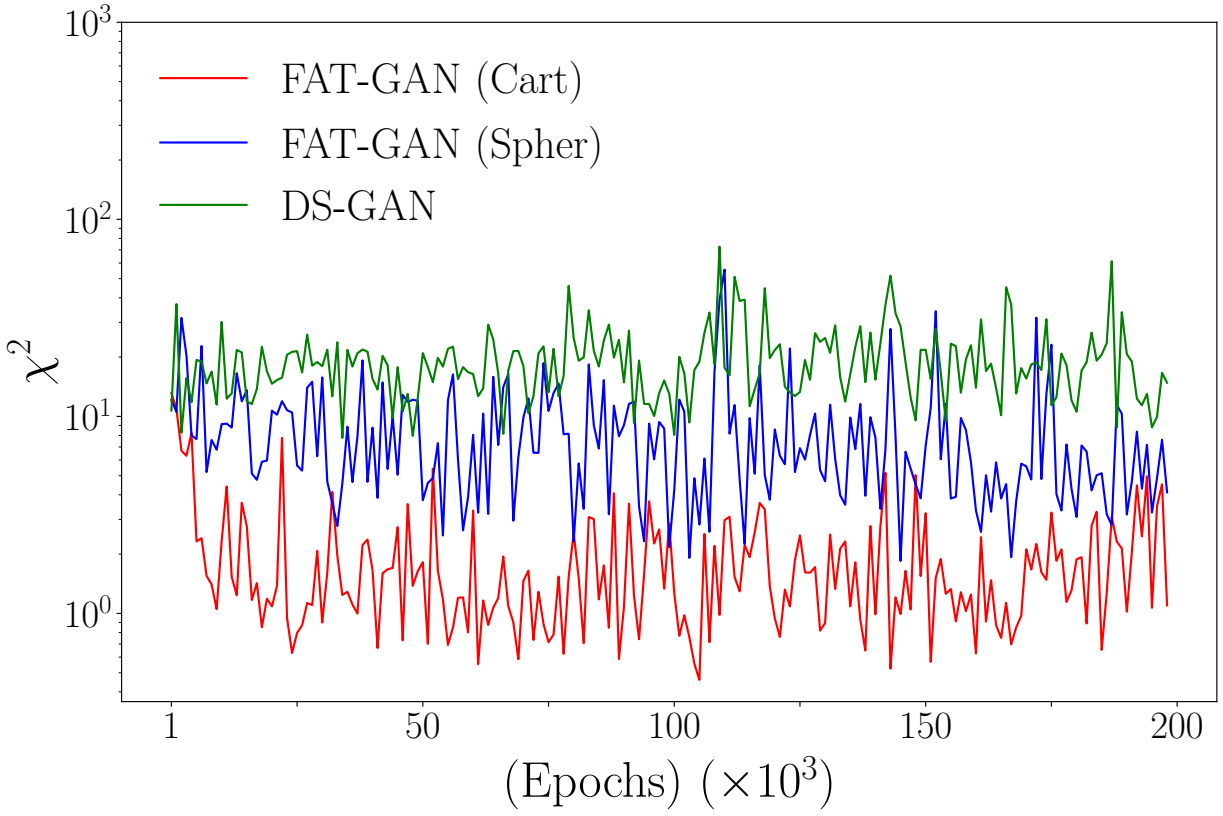


Fig. 11.: Comparison of χ^2 values for $x_{\rm bj}$ distributions of Pythia events and those generated by the FAT-GAN (Cart), FAT-GAN (Spher) and DS-GAN, with respect to the number of training epochs.

generated by FAT-GAN match well with those of the experimental data. Note, however, that in contrast to the smooth distributions from the Pythia MCEG in Fig. 7., the CLAS data here have not been corrected for detector effects, and therefore display artifacts such as the shoulder in the $x_{\rm bj}$ distribution. Nevertheless, the comparison in Fig. 12. indicates that the physical properties p_T, θ, $x_{\rm bj}$, and Q^2, which are not directly used in the training, are correctly captured by the FAT-GAN trained on the experimental event data.

4.6 CONCLUSION

In this work we investigate the use of GANs to simulate electrons in the final state of high-energy inclusive electron-proton collisions. We report that selecting the appropriate

features as generated features or augmented features plays a critical role in building a successful GAN event generator. While the physical properties of the particles often exhibit challenging distribution patterns for a GAN to learn, using transformed features enables the generator to more easily generate and avoid unphysical events. Augmenting in addition the feature space for the discriminator to become more sensitive, our FAT-GAN demonstrates good agreement with simulated as well as experimental data in mimicking event feature distributions and their inter-correlations. We also find that the selection of the appropriate coordinate representations impacts the GAN performance. Although the FAT-GANs presented are specific to electron-proton scattering, the feature selections and transformation strategy can be generalized to GANs for simulating other reactions under different conditions, as well as learning exclusive events.

The FAT-GAN package is available at `https://github.com/ijcai2021/FAT-GAN`.

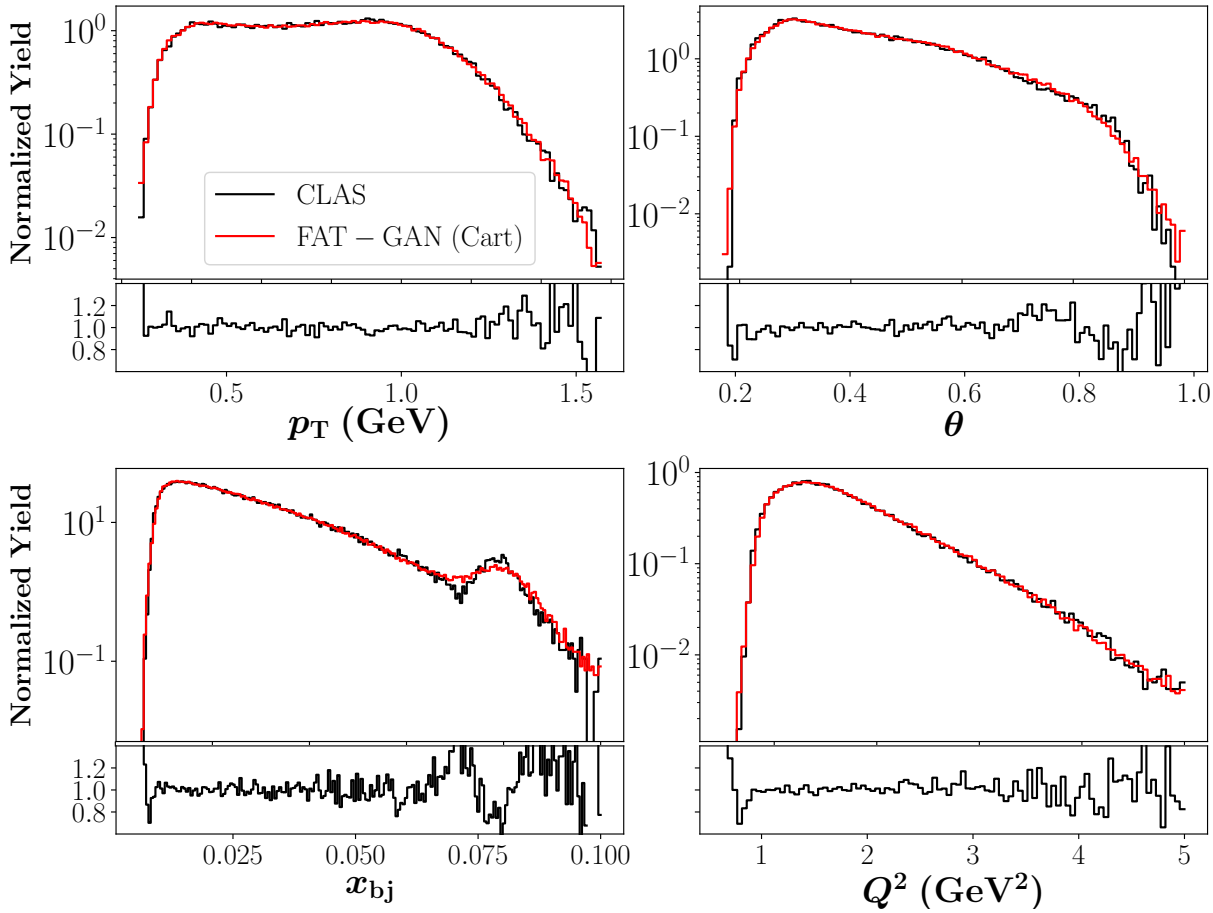


Fig. 12.: Comparison of the distributions of the physical properties p_T, θ, $x_{\rm bj}$ and Q^2 generated by FAT-GAN (Cart) with those from CLAS at Jefferson Lab (uncorrected for detector effects). The ratio of FAT-GAN (Cart) to CLAS data is shown at the bottom of each panel.

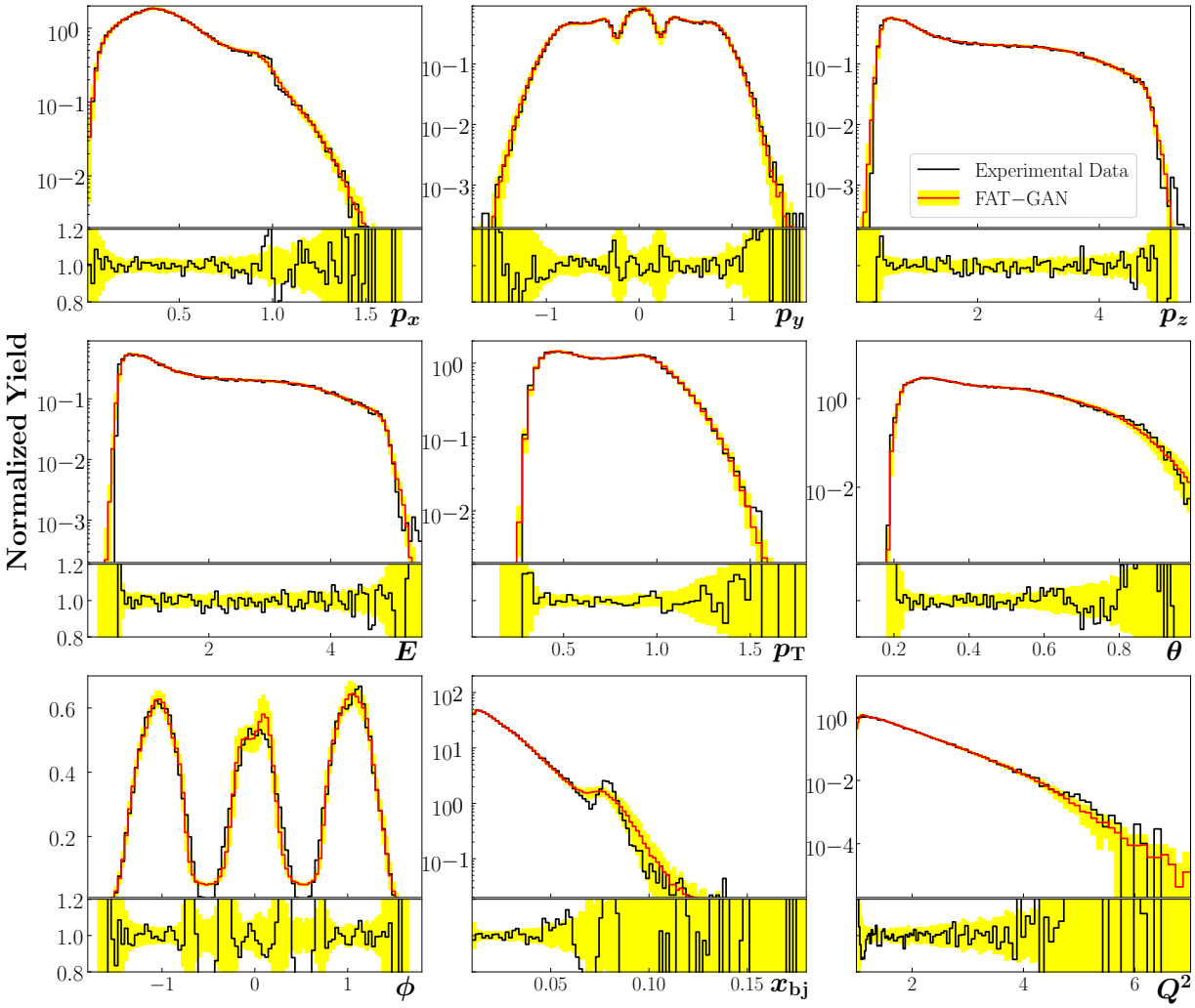


Fig. 13.: AS in Fig. 7., but with CLAS Data

CHAPTER 5

CONDITIONAL FEATURE-AUGMENTED AND TRANSFORMED GAN

In this chapter, we extend the FAT-GAN model to develop a Conditional Feature-Augmented and Transformed Generative Adversarial Network (cFAT-GAN) that was published in [11]. The Major contribution of cFAT-GAN is to allow the model to generate events for different energy beams including events of the energy beams that are not part of the training data. We begin the discussion in Sec 5.1 with the motivation behind extending the FAT-GAN. Then, we give a brief background of conditional GAN prior work. Aftter that, we explain the model architecture and the feature representations. Finally, we present the results and examine the cFAT-GAN ability to interpolate and extrapolate between different energy beams. This chapter is related to **RQ1:** Can we build a ML-based generator to faithfully reproduce particle collision events?.

5.1 INTRODUCTION

One constraint of FAT-GAN is that the model depends on a given collision energy, which can only generate events corresponds to that energy. To generate events at unrelated collision energies, the model must be retrained to capture the new phase-space dataset. However, many practical applications require the event generator to have the flexibility of allowing users to specify the reaction energy as an input to produce the corresponding synthetic events. To overcome this limitation, we extend the FAT-GAN framework by conditioning the component neural networks according to the given reaction energy. We demonstrate that cFAT-GAN, can reliably produce inclusive event feature distributions and correlations for a continuous range of reaction energies by automatically interpolating and extrapolating from a set of trained energies. As a proof of concept, we train on events generated by the Pythia MCEG [15] as a proxy for experimental data and we restrict the GAN's task to generate a specific final-state particle (electron) as in [10], which is referred to as an inclusive event generator.

The cFAT-GAN implementation can be found at: `https://github.com/AnonymousICMLA/cFAT-GAN`.

5.2 CONDITIONAL GAN PRIOR WORK

MCEGs have been vital tools for the analysis and design of high energy scattering experiments and phenomenological investigations since their conception in the 1970s. General generator models such as Pythia [15], HERWIG [23], and Sherpa [24] function to simulate the theoretical physics processes that occur in high-energy lepton-lepton, lepton-hadron, and hadron-hadron collisions to a user's specifications. Recently, GANs [3] have been used in different applications in the high-energy field, such as simulating energy deposits of particles [68, 5] and jets [6, 7], where they focus on feature representations of the experimental data in contrast to event based samples. Hashemi *et al.* [8] have trained GAN at the event level in which they generate the four-momentum in $Z \to \mu^+\mu^-$ events generated by Pythia. In addition, Butter *et al.* [9] incorporated maximum mean discrepancy (MMD) [69] on their event generator in order to resolve features that have spiky and sharp distributions. To the best of our knowledge, there have not been any prior works that condition GAN on the reaction energy.

cGANs have shown large improvements in computer vision applications, for example, Image-to-Image Translation [70], convolutional face generation [71], and video generation [72]. There are also current efforts that aim to improve speech system performance in noisy environment [73] where the model is conditioned on the noisy spectrogram. cGANS have also been used in natural language processing like [74], which is conditioned on the context of the sentences. In cFAT-GAN, we condition the FAT-GAN networks on the beam energy and we show that cFAT-GAN can successfully generate events from continuous range of beam energies, both interpolatively and extrapolatively, expanding the scope of the GAN-based event generators.

5.3 MODEL ARCHITECTURE

We extend the FAT-GAN framework to a conditional model in which the generator and discriminator are conditioned on the collision beam energy, E, as shown in Fig. 14.. Similar to FAT-GAN, our GAN framework adopts the architecture of WGAN developed in [49].

We consider a data sample consisting of inclusive electron-proton scattering events at five beam energies, $E = 10$, 20, 30, 40, and 50 GeV in equal parts.

The inputs to the generator, G, is a 100-dimensional white noise vector and the incident lepton energy, E. The two inputs for G are concatenated and passed through the generator network consisting of 8 fully connected hidden layers, each with 512 nodes and activated by leaky Rectified Linear Unit (ReLU) function. The final hidden layer is fully connected to a 3-node output activated by a linear function representing the generated features. These

Event Features	k'_x	k'_y	$\mathcal{T}(k'_z)$	k'_z	k'_T	E'	k'_z/k'_T
True Samples	✓	✓		✓			
FAT (True Samples)			✓		✓	✓	✓
Generator	✓	✓	✓				
FAT (Generator)				✓	✓	✓	✓
Discriminator	✓	✓	✓	✓	✓	✓	✓

TABLE 4.: Generated features from the generator and Pythia. The augmented and transformed features calculated from these generated features.

generated features and E are passed to a customized lambda layer to calculate the augmented features. The augmented features and E are concatenated to the generated features and fed to the discriminator as input.

Mirroring the generator, the discriminator network, D, consists of 8 fully-connected hidden layers, each with 512 nodes, activated by a leaky ReLU function. To avoid overfitting in classification, a 10% dropout rate is applied to each hidden layer. The last hidden layer is fully connected to a single-neuron output, activated by a linear function, where "1" indicates a true event and "0" indicates a fake event. In the discriminator network, E acts as an input feature, serving to differentiate event distributions according to the given beam energy. The generator loss function, L_G, and the discriminator loss function, L_D, are respectively defined as:

$$L_G = -\mathbb{E}[D(\sqrt{s}, \widetilde{\boldsymbol{F}}))] \tag{18}$$

$$\begin{aligned} L_D &= \big(\mathbb{E}[D(\sqrt{s}, \widetilde{\boldsymbol{F}}))] - \mathbb{E}[D(\sqrt{s}, \boldsymbol{F})]\big) \\ &+ \lambda\, \mathbb{E}_{\widehat{\boldsymbol{F}} \sim P_{\widehat{\boldsymbol{F}}}}\big[(\|\nabla_{\widehat{\boldsymbol{F}}} D(\sqrt{s}, \widehat{\boldsymbol{F}})\|_2 - 1)^2\big] \end{aligned} \tag{19}$$

where $\mathbb{E}$ signifies the expectation value, $\widetilde{\boldsymbol{F}}$ is a synthetic feature vector including generated and augmented features, and $\boldsymbol{F}$ is a true feature vector.

5.4 AUGMENTED FEATURES

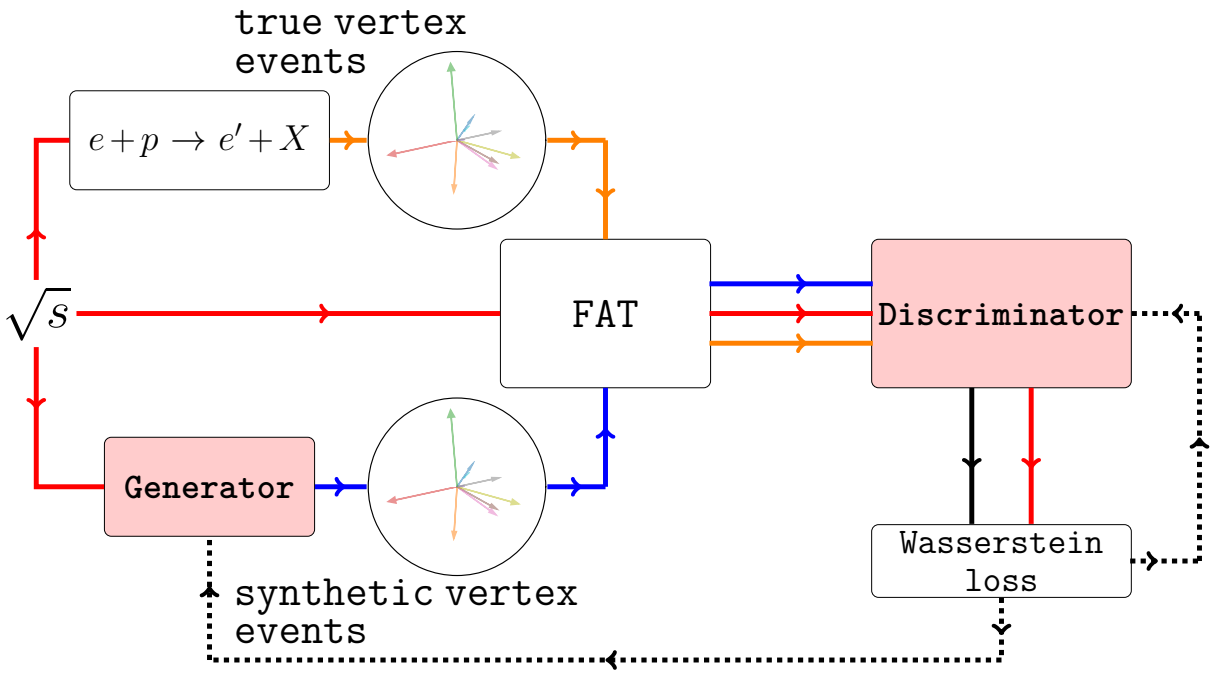


Fig. 14.: Architecture of the cFAT-GAN event generator The red arrows represent the flow of the incident energy, E; the dotted arrows represent the back propagation of the loss. The blue and orange arrows indicate the flow of the synthetic and true samples in cFAT-GAN framework, respectively.

Our previously devolved FAT-GAN [10] introduced a new machine learning approach to constructing MCEGs tailored to the specific challenges that accompany the qualifications of current event generators. Such qualifications require the GAN-based generator to operate within the laws of physics and model the distributions of event features and their correlations to a much higher degree of precision than what is required in other applications.

To improve GAN training in FAT-GAN, feature transformations were used to simplify the distributions that the generator must learn and, more importantly, to prevent the generation of non-physical events. Additionally, a set of augmented features calculated from the generated features is included in the training set and generator output to enhance the sensitivity of the discriminator.

Inclusive electron-proton scattering refers to the process $e(k)+p(P) \rightarrow e(k')+X$, where k and P are the momenta of the incident electron and target proton, respectively, and k' is the momentum of the final state detected electron. The state X represents the sum of particles that are produced in the reaction, but not detected, other than the electron. In the c.m. frame of the initial state electron and proton, we can approximate the electron and proton 4-momentum vectors by $k \approx (\sqrt{s}/2)(1;0,0,1)$ and $P \approx (\sqrt{s}/2)(1;0,0,-1)$, with the invariant c.m. energy squared $s = (k+P)^2$.

Energy conservation requires that the scattered electron energy $E' \equiv k'_0$ is less than the incident energy, $E \equiv \sqrt{s}/2$. A typical feature in the reaction is that the distribution of E' has a sharp edge at the end of the phase space that is rather difficult for the GAN to learn, as the discriminator lacks the sensitivity to identify the non-physical events where $E' > E$, particularly when $E' - E$ is very small. Such sharp edge in the phase space also occurs for the the scattered electron's longitudinal momentum distribution, k'_z.

To address the problem of learning sharp edge distributions, a variable transformation $\mathcal{T}(k'_z) = \log(E - k'_z)$ is chosen to replace the k'_z as a generator feature. Thus, the generator produces the 3 features corresponding to $k'_x, k'_y, \mathcal{T}(k'_z)$. The $\mathcal{T}(k'_z)$ distribution turns to be closer to a normal distribution, making it easier for the generator to learn, and the same time prevents the generation of non-physical values $E' > E$. To enhance the sensitivity of the discriminator, additional features, including E', $k'_T = \sqrt{{k'_x}^2 + {k'_y}^2}$, and k'_z/k'_T, are computed as augmented features to be used as inputs fro the discriminator. The generated, augmented, and transformed features are listed in Table 4..

5.4.1 BEAM ENERGY REPRESENTATION

In the original implementation of conditional GAN [75], and many other conditional

applications, the generator and discriminator networks are conditioned on categorical labels that are often one-hot encoded. Common image generation applications are well suited to this discrete representation, allowing the user to specify the type of image to generate from a trained set of distinct classes. Unlike image classification, scattering events from collisions at different c.m. energies are not suited for sorting into distinct classes, as the feature distributions changes subtly but non-trivially as a function of the reaction energy.

To encode the c.m. energy $\sqrt{s}$ as a continuous input feature to the cFAT-GAN, we calculate the mean and standard deviation of the incident beam energies ($\equiv \sqrt{s}/2$) represented in the training set, and then apply a standard normalization. The normalized beam energies serve as a continuous representation and the same normalization is applied to an untrained energy, placing it in reference to the trained energies in the generator's latent space. Thus, the fully trained generator, when passed the normalized untrained energy, is able to automatically interpolate between different training energies or even extrapolate beyond.

The normalized beam energy passed as an input feature in the GAN is also used in the custom Lambda layer in which the augmented features are calculated and concatenated to the generator output. The augmented feature, k'_z, calculated from the generated feature $\mathcal{T}(k'_z)$, requires the incident beam energy E. Thus, the normalized energy is passed to the Lambda layer, un-normalized, and used to calculate $k'_z = \exp[E - \mathcal{T}(k'_z)]$. The transformation of k'_z at different beam energy levels is illustrated in Fig. 15.. By incorporating E into the transformation of k'_z, the cFAT-GAN inherits an important property from the FAT-GAN, which eliminates the chance of generating non-physical events under varying beam energies which is easier for the generator to mimic.

5.5 RESULTS

In this section, we examine the trained generator and test its ability to produce the true underlying event distributions. More importantly, we analyze the interpolation and extrapolation capability of the trained cFAT-GAN on five different training energy samples and test its predictability to generate samples for near by energy of the training energies as well as energies away from the training set. Finally, we analyze the sensitivity of the generator and discriminator networks to the conditional beam energy by visualizing the latent variables in the hidden layers of each network for various input energies.

5.5.1 EVENT FEATURE DISTRIBUTIONS

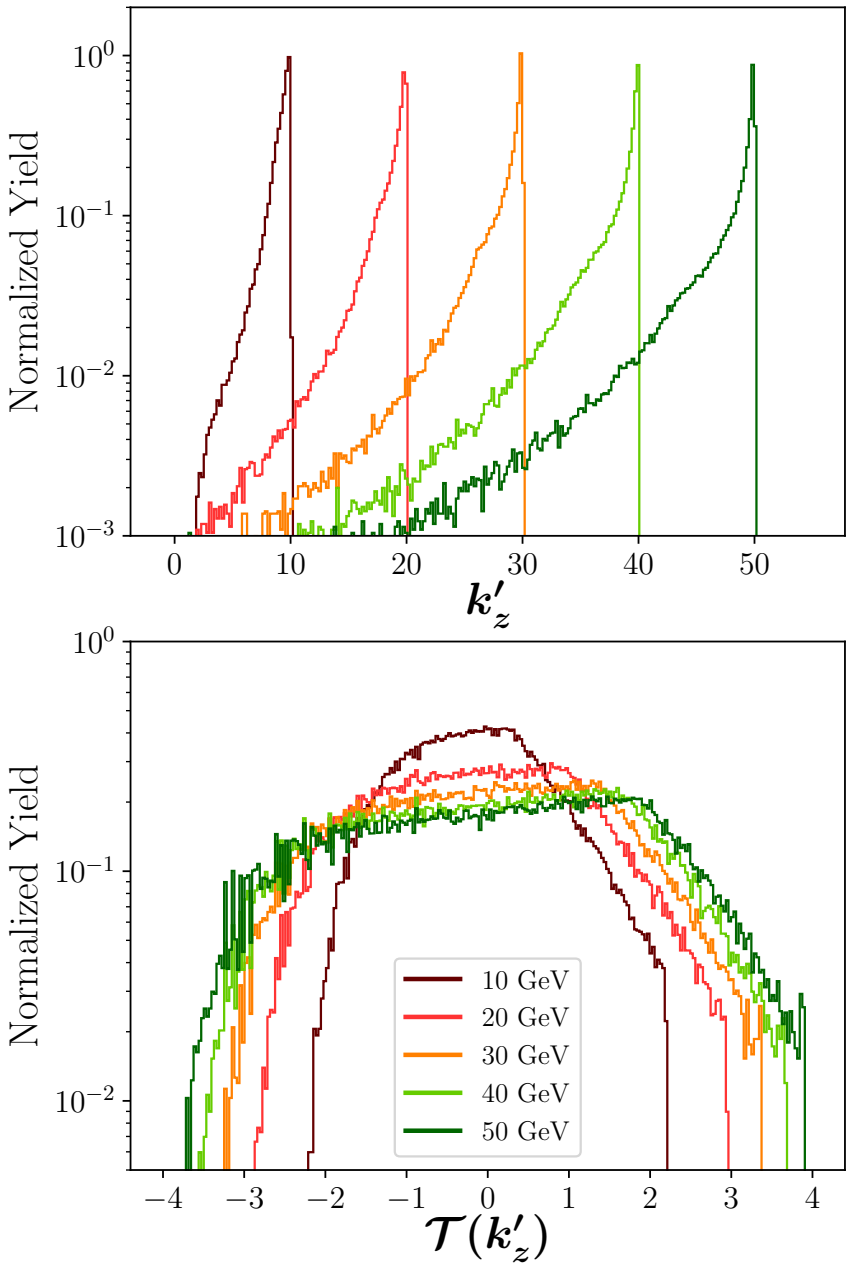


Fig. 15.: The distributions in k'_z and $\mathcal{T}(k'_z)$ for the 5 beam energies: $E = 10$, 20, 30, 40 and 50 GeV. The transformation prevents the cFAT-GAN from generating non-physical events with respect to its given beam energy input. The transformation also converts distributions with sharp edges into smooth distributions that are easy for the generator to mimic.

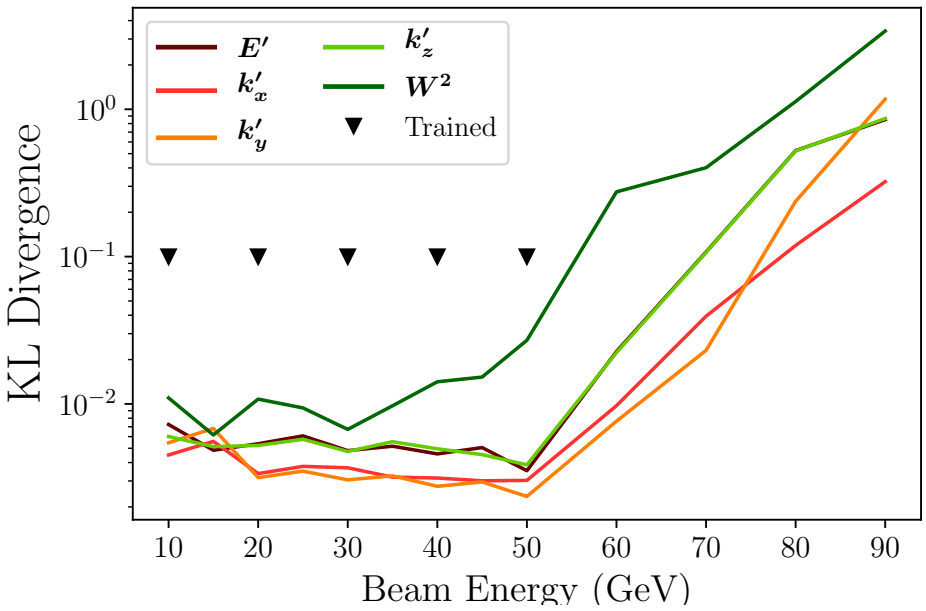


Fig. 16.: Kullback-Leibler (KL) divergence calculated between the synthetic and true E', k'_x, k'_y, and k'_z distributions at each energy represented in Fig. 17.a and 17.b.

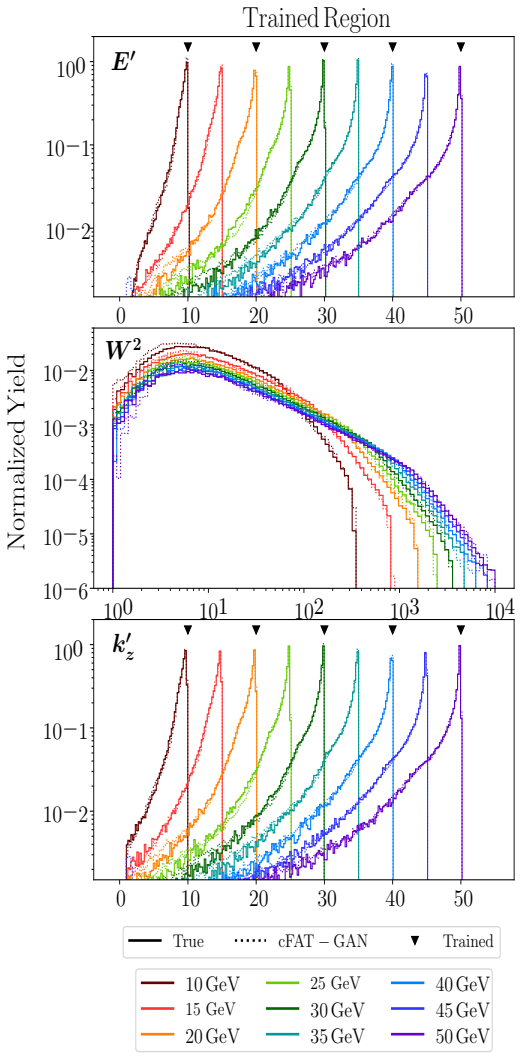
We directly examine the cFAT-GAN's ability to generate key event features for which the reaction energy is a significant variable. To further analyze the generator's performance, we consider the invariant mass squared of the unobserved hadronic final state, W^2, which is defined as

$$W^2 = (q + P)^2, \tag{20}$$

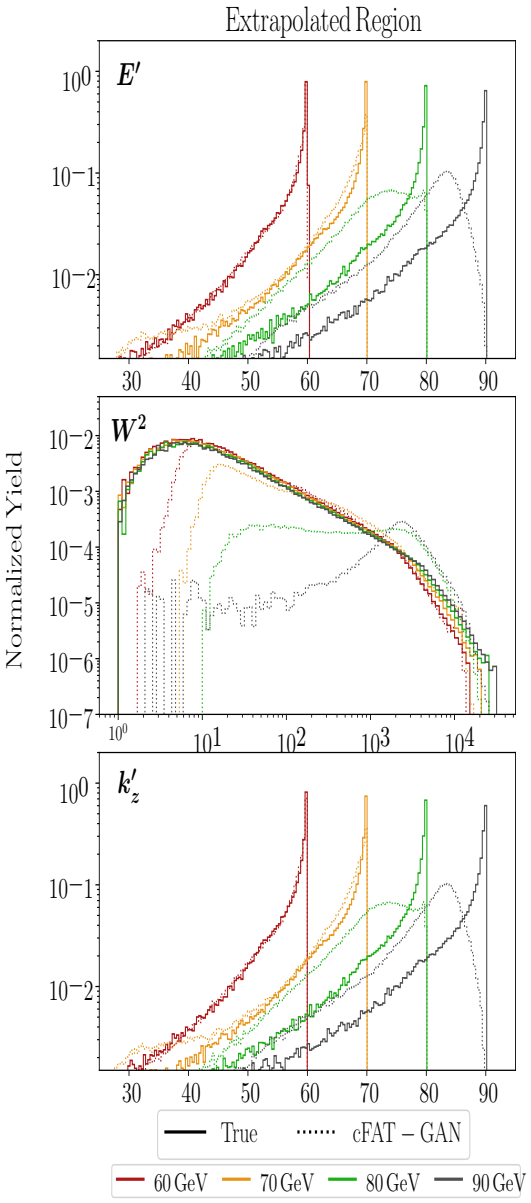
where $q = k - k'$ is the 4-momentum transfer from the electron to the proton. The E', W^2, and k'_z distributions from the cFAT-GAN are compared with the corresponding true distributions, for the five trained energies and eight untrained energies used in Figs. 17.a and 17.b.

In addition, we employ the Kullback-Leibler Divergence statistic to quantify the distance between the synthetic and true distributions in Fig. 16. at different energies.

As shown in Fig. 17.a, one can find that the cFAT-GAN generated feature distributions of E', W^2, and k'_z at the five trained energies—10, 20, 30, 40, and 50 GeV—are shown in to match well with the corresponding true distribution with KL-divergence values around or less than 10^{-2} (Fig. 16.). This indicates that cFAT-GAN inherits FAT-GAN's capability of reproducing event distributions of the trained beam energies. From Fig. 17.a, one can find



(a) Comparison of the synthetic (dotted) and true (solid) E', W^2, and k'_z feature distributions at the 5 trained energies: E = 10, 20, 30, 40, and 50 GeV and 4 untrained interpolated energies: E = 15, 25, 35, and 45 GeV. The black triangles specify the trained energy levels.



(b) Comparison of the synthetic (dotted) and true (solid) E', W^2, and k'_z feature distributions at 4 untrained extrapolated energies: E = 60, 70, 80 and 90 GeV.

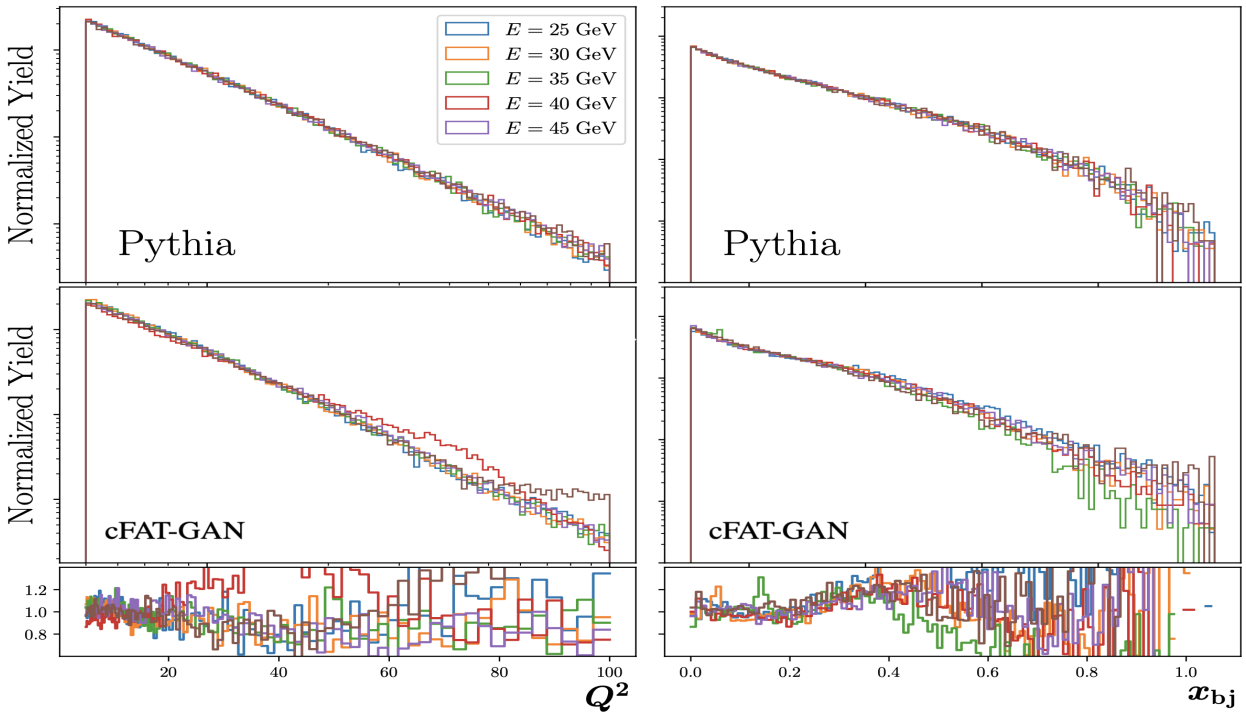


Fig. 18.: Comparison of Q^2 and x_{bj} variables between Pythia and cFAT-GAN for several energy reactions.

that the synthetic event distributions of E', W^2, and k'_z at the interpolated energies—15, 25, 35, and 45 GeV— also show good agreement with their true counterparts, similar to those of the trained energies. The good agreement is confirmed in the KL-divergence plot shown in Fig. 16.. This indicates that the generator has learned the general relationship between the event distributions and the beam energy in the training process. Thus, cFAT-GAN is able to interpolate automatically when it is fed the appropriate energy to produce reliable physical events at energies between those trained. Our results indicates that for our current ML architecture, 10 GeV is an effective trained energy level gap, across which cFAT-GAN can accurately interpolate at event level in electron-proton scattering. We also investigate the extrapolation capability of the cFAT-GAN, whose results are shown in Fig. 17.b. For the event distributions at energies not far from the maximum trained energy level, such as 60 GeV, relatively good agreement between the synthetic and true distributions is still found, but with KL-divergence values higher than those within the scope of the trained energies. The disagreement continues to enlarge at higher extrapolated beam energies, as shown in Fig. 16.. This is as expected, since higher beam energies move farther away from

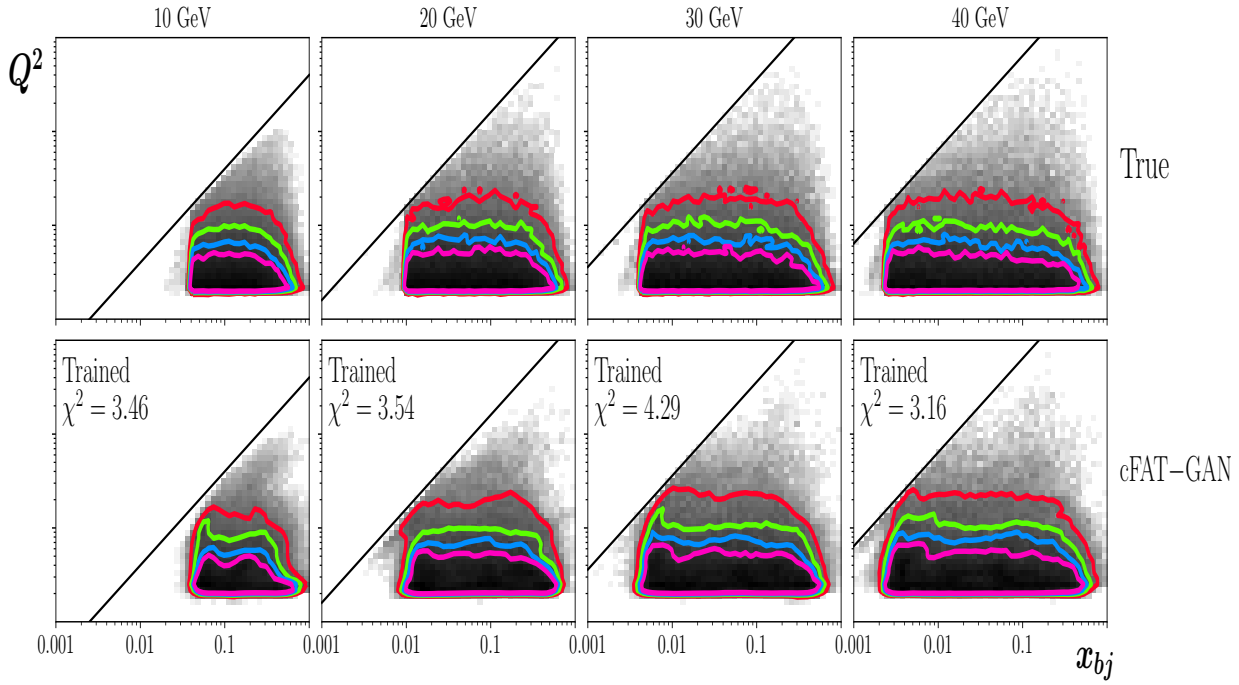


Fig. 19.: Synthetic and true joint distributions of Q^2 and x_{bj} for trained energies: $E = 10$, 20, 30, and 40 GeV

the closest well-known point, 50 GeV.

5.5.2 PREDICTABILITY FOR NEW ENERGY BEAMS

We next examine the degree of predictability of new features that were not included as part of the feature set. To that end we consider two variables typically encountered in electron-proton reactions: $Q^2 = -q^2$, the square of the invariant mass of the exchanged virtual photon, and $x_{\mathrm{bj}} = Q^2/2P \cdot q$, the Bjorken scaling variable.

In Fig. 19., we plot the joint distributions of Q^2 and x_{bj}, calculated from the generated and true events at each trained energy. We define $\chi^2 = \frac{1}{n}\sum_i^n (g_i - t_i)^2/t_i$, where n is the total number of bins, and g_i and t_i are the numbers of points falling in a given bin for the generated and true distributions, respectively, in order to statically measure the agreement between the two distributions at each beam energy.

By inspecting the iso-contour lines, one finds that the cFAT-GAN joint distributions generally match well with the corresponding true joint distributions across all the trained

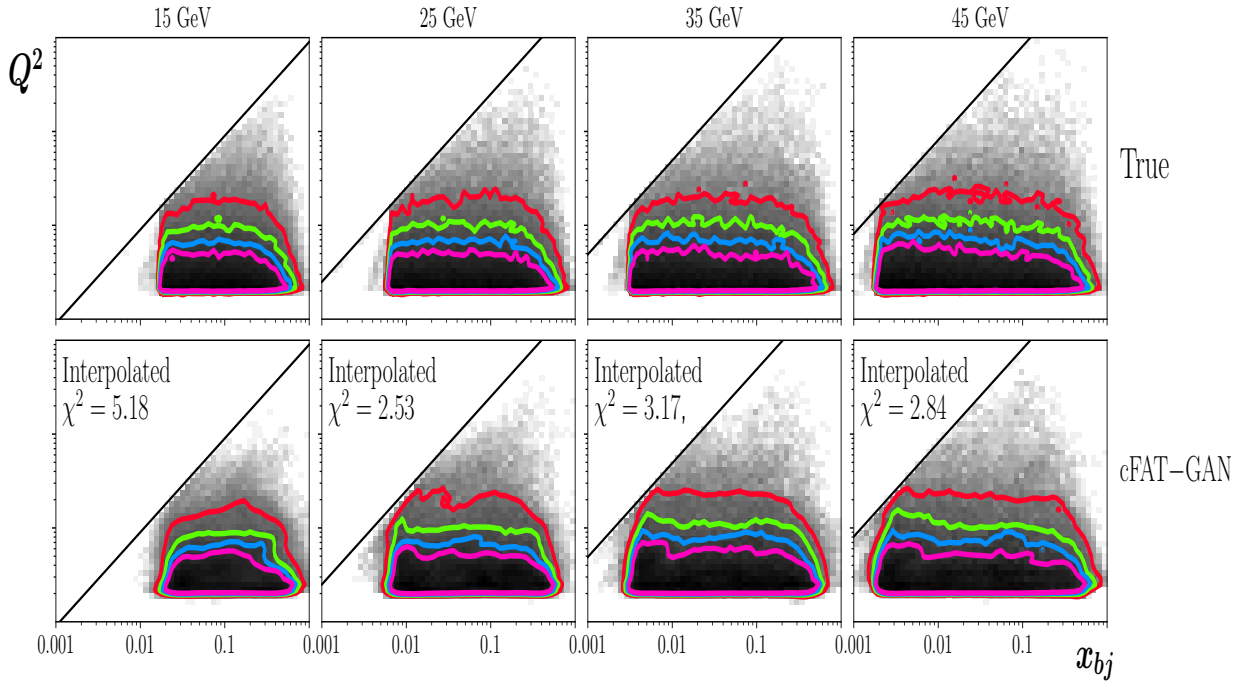


Fig. 20.: Synthetic and true joint distributions of Q^2 and x_{bj} for interpolated energies: $E =$ 15, 25, 35, and 45 GeV

energies, with χ^2 values ranging from ≈ 2 to ≈ 5. Moreover, the joint Q^2–x_{bj} distributions for interpolated beam energies in Fig. 20. also demonstrates good agreement with those generated by Pythia, with similar χ^2 values as events at the trained energies. Our results indicate that the cFAT-GAN framework not only learns the distributions of the observables within the range of the trained beam energies correctly, but also learns the correlation among predicted features not included in the training feature set.

5.5.3 LATENT VARIABLES ANALYSIS

The results present so far have tested the degree of compatibility of the feature distributions generated by the trained cFAT-GAN against those from the true samples. In this section, we examine the latent variables within the hidden layers of the neural networks in cFAT-GAN in order to examine the sensitivity of the trained architectures with different input energies. To extract the generator's latent variables, the generator network was fed with a batch of 10,000 noise vectors, each concatenated with a beam energy at either 10, 20, 30, 40, or 50 GeV, and the output was adjusted to include the second, fifth, and eighth hidden

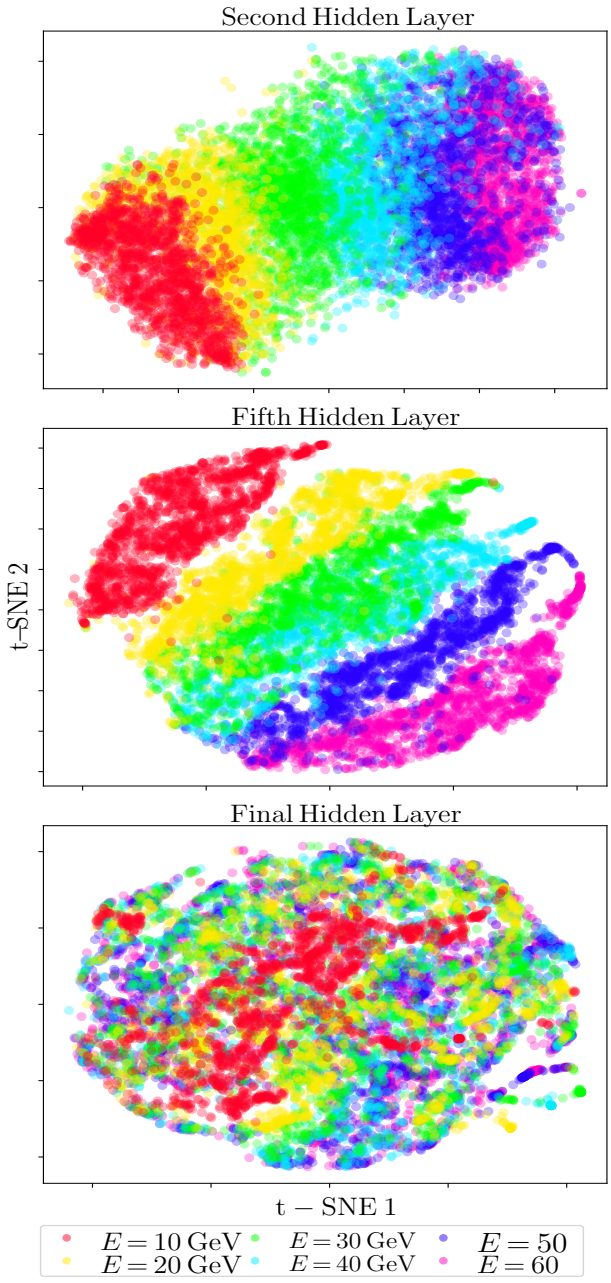


Fig. 21.: 2-D t-SNE visualization of the latent variables in the second, fifth, and final hidden layers of the generator for the five beam energies E =10, 20, 30, 40, and 50 GeV, and one extrapolated energy of 60 GeV.

layers. Repeating this process for each trained energy and one extrapolated energy, 60 GeV, with the corresponding labels, we employ t-Distributed Stochastic Neighbor Embedding

(t-SNE) [76] for dimensionality reduction of the latent variables from 512 dimensions to 2 dimensions to facilitate visualization. Fig. 21. shows the two principal 2-D projection from the three selected hidden layers for each energy. One can clearly find the spectrum of the event distributions at different beam energies on the low-dimensional manifold of the hidden layers. One can also find that the latent variables for the extrapolated energy, 60 GeV, fit into the progression of the pattern within the generator hidden layers, as would be expected if it had been a trained energy, appearing closest to the 50 GeV projection. This illustrates the network's ability to learn the relationships between trained energies that enable the cFAT-GAN to perform well at interpolated and close extrapolated energies. Similar to many deep learning applications, in the final hidden layer, the neural networks extract highly abstract patterns, where the events with respect to different reaction energy levels are mixed up and the extracted patterns are not quite explainable.

Since the discriminator in cFAT-GAN acts as a classifier, trained to distinguishing the "true" and the generated events, we compare the latent variables in the discriminator neural networks when fed samples of cFAT-GAN and Pythia generated events. The trained cFAT-GAN discriminator was fed with 100,000 sample events from either Pythia or the generator at each of four selected beam energies—10, 50, 35 and 60 GeV—and the output was adjusted to include the eighth hidden layer. To reduce the dimensionality of the latent variables, we employed Principal Component Analysis (PCA), fitting to the principle components from the latent variables corresponding to the "true" input, and then transforming the hidden layer activation values, with respect to the Pythia and synthetic samples, from 512 dimensions to 2 dimensions. As shown in Fig. 22., the latent variables from the eighth hidden layer corresponding to Pythia and synthetic events are projected to a two-dimensional space spanned by the first and second principle components. The Pythia events (contours) and generator events (heat map) demonstrate good agreement, indicating that the trained discriminator considers the GAN generated samples very close to Pythia generated samples at 10 and 50 GeV trained energies, as well as the untrained energies including 35 GeV (interpolated) and 60 GeV (extrapolated).

5.6 CONCLUSION

In this work we describe the implementation of a cFAT-GAN framework by incorporating the conditional beam energy as an input feature. The cFAT-GAN not only demonstrates faithful reproduction of events at the trained beam energies, but also correctly generates

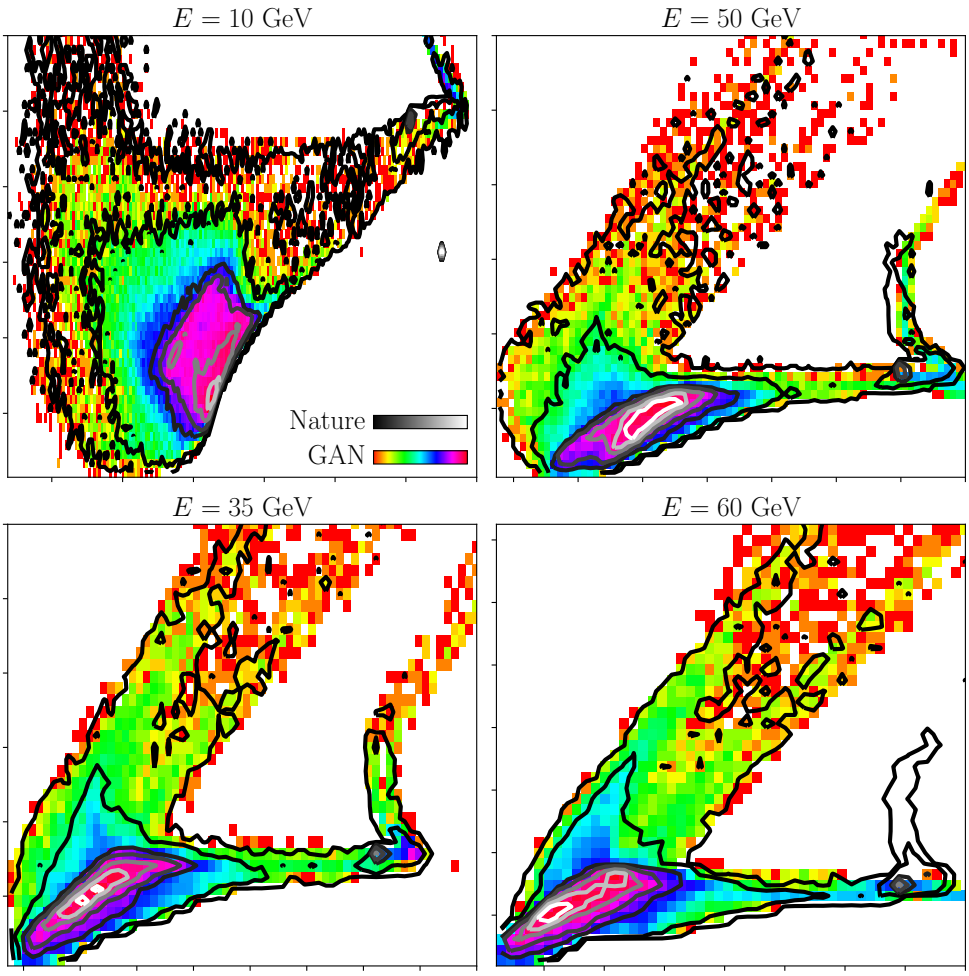


Fig. 22.: 2-D PCA visualization of the latent variables of the final hidden layer of the discriminator when feeding Pythia samples (contours) and generated samples (heatmap) for $E = 10$ and 50 GeV trained energies, as well as 35 and 60 GeV untrained energies.

events for interpolated energies. Moreover, the cFAT-GAN can extrapolate events for untrained energies with good agreement, provided the extrapolated energy is not too far from the trained energy level. The range and the number of energies selected for training are critical parameters that can be tuned to improve the model's ability to interpolate and extrapolate to untrained energies, and expand the energy spectrum for which the cFAT-GAN can correctly reproduce physics events. While the conditional beam energy presented in this paper is specific to inclusive electron-proton scattering, the cFAT-GAN methods can be applied to GANs simulating exclusive scattering reactions or other types of collision reactions.

CHAPTER 6

SIMULATION OF DETECTOR EFFECTS

In this chapter we present the simulation of smearing detector effects that is called ("folding"). This chapter mainly addresses **RQ2:** Can we simulate the smearing detector effects using ML tools?. As discussed in 2.3.2, folding is the process to simulate detector effects by distorting vertex-level events to generate detector-level events. Unfolding is to reconstruct vertex-level events. We begin the discussion in Sec 6.1 with a background of the detector effects. Then, in Sec 6.2 we discuss some of the challenges to develop a folding procedure that satisfies physics constraints. In sec 6.3 we describe the dataset used to test this work. After that, in Sec 6.4 we introduce our *direct folding* model to simulate the smearing detector effects of toy and realistic examples and discuss its limitations. Then, we extend the *direct folding* to *conditional folding* model to solve some of the limitations discussed in 6.2. Finally, we show the results of the *conditional folding* model using several examples.

6.1 INTRODUCTION

After the particle collisions or interactions happen, we observe the collision data or "events", and that is where particle detectors come in. A particle detector, as discussed in 2, is a machine installed within the accelerator that acts as a camera to detect and track particles, and can measure the particle energy, momentum, spin, and other attributes. Hence, experimentalists can only observe the data that the detectors detect, which is called ("detector-level") events. Detector machines have limited resolutions, and can significantly alter the true events ("vertex-level"). The difference between vertex-level and detector-level events called ("detector-effects"). More details of detector effects can be found in section 2.

6.2 FOLDING CONSTRAINTS AND CHALLENGES

To implement a ML folding procedure, one needs to address not only ML obstacles, but also physics constraints and laws. There are several challenges need to be taken into account when implementing a folding procedure:

- The physical intuition indicates that the entire mapping between vertex-level to detector-level is statistical in nature. Therefore, a simple ML function, such as a neural network,

that maps vertex-level events to detector-level events will be deterministic where each vertex-level event will always be mapped to a detector-level event, and this violates the physical intuition.

- The folding algorithm needs to fold events that are not the same as the training data. In other words, the training data might be events that cover the full phase space, and then we need to fold events that cover part of the detector-level space. This is an important aspect of the folding because true-vertex events are unknown, hence we need an algorithm that can learn the behaviour of folding instead of directly mapping from one space to another as in other GAN applications, such as (image-to-image translation).

- Small changes in the detector-level distribution can cause large changes in the reconstructed true distribution, so one needs to develop an accurate folding that can simulate detector effects with high accuracy.

- The folding algorithm needs to reproduce the training features as well as their correlations precisely to be correctly measured and analyzed.

6.3 DATA DESCRIPTIONS

For this work, we define three sets of data that will be used to test our methodology: the first is a toy example generated from random Gaussian and uniform distributions. The second data set is inclusive deep-inelastic scattering (DIS) reaction. The third is a realistic MC data that is generated based on real experimental data from CLAS collaboration [66] at Jefferson Lab.

For the toy data, we generate two sets of files for training and validation. For the training, we define a toy example called toy_1, which is a random normal distribution defined as, $X \sim \mathcal{N}(\mu, \sigma^2)$, where X is the training vertex-level events that are sampled from a Gaussian distribution centered at 0 with unit standard deviation. We also define the training detector-level events as, $Y = (X + \epsilon)$ where Y is detector-level events, which can be obtained by shuffling vertex-level with 0.05 random normal noise to have a proxy for detector smearing effects. For the test set, we define another example called $toy2$ sampled from an uniform distribution, and we can define $toy2$ such that, $X_1 \sim \mathcal{U}(a, b)$ where X_1 is the test vertex-level events sampled from a uniform distribution. We also define the test

detector-level Y_1 by shuffling the vertex-level with random noise, so $Y_1 = (X_1 + \epsilon)$ and similarly we define the noise value to be 5%.

For the DIS data, we generate inclusive electron-scattering events in the resonance region by MC sampling of an existing parametrization [77] that is known to describe well the data at these low-energy kinematics. We also use models for the pure resonance behavior [78, 79] to explore the effects seen in the cross sections stemming from the nucleon resonance contributions separated from the remaining resonant and nonresonant channels. The DIS data contains training and validation datasets, where each file is represented as electron three momenta ($l'x$, $l'y$, $l'z$).

Realistic example is based on CLAS [66] data, which has two particles: proton and pion in which they have been measured in high-statistics experiments at Jefferson Lab and extensively analysed within traditional approaches[80, 81, 82, 83]. This realistic data is generated from MC simulations that can be a very good proxy to CLAS experimental data.

6.4 DIRECT FOLDING GAN

We implemented a folding GAN which we call ("*direct folding*") to investigate the possibility of solving detector effects. The direct mapping model folds vertex-level into detector-level events. The model is based on LSGAN architecture, discussed in Chapter 3, which contains a generator that takes vertex-level inputs and a discriminator that supervises on detctor-level events. As shown in Fig. 23., the input to the generator "G" is a n-dimensional vector array represents vertex-level events. The generator network consists of 5 hidden dense layers, each with 512 neurons, activated by a leaky Rectified Linear Unit (LReLU) function. The last hidden layer is fully connected to a n-dimensional output, activated by a linear function representing the generated detector-level events. The output of the generator is fed to the discriminator as input.

As for the generator, the neural network in the discriminator "D" also consists of 5 hidden dense layers, each with 512 neurons, activated by a LReLU function. To avoid overfitting in classification, a 10% dropout rate is applied to each hidden layer. The last hidden layer is fully connected to a single-neuron output, activated by a linear function, where "1" indicates a true detector-level events and "0" is a fake detector-level events.

6.4.1 RESULTS

To test this methodology we start with a minimal toy example and validate that this approach can at least work for a simple scenario before we proceed to more realistic and

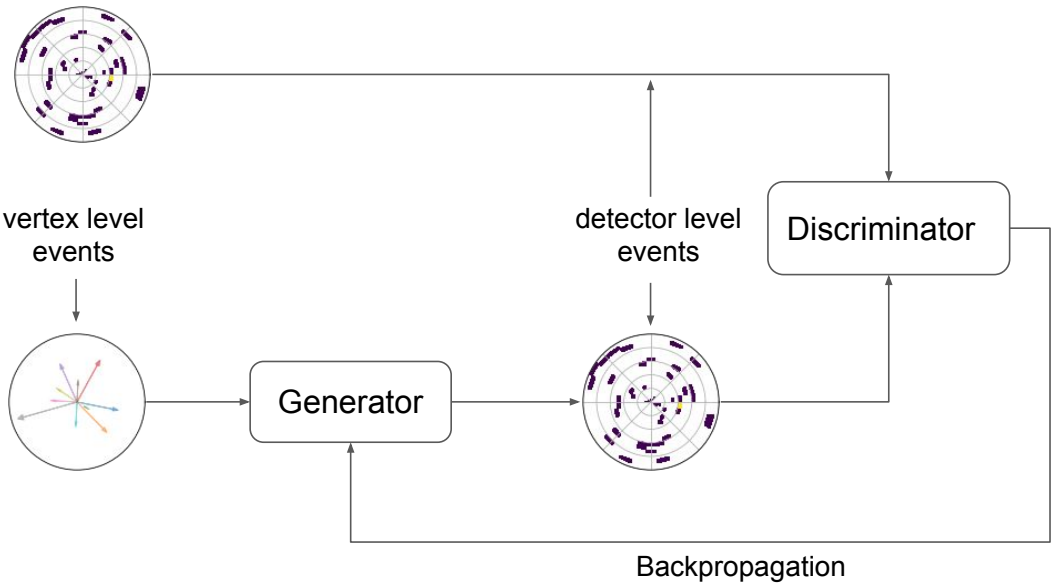


Fig. 23.: Direct Folding Setup: The input to the generator is vertex-level events that produces the corresponding detector-level events. The discriminator supervises only on detector-level events to assure the generated event by the generator match the true samples.

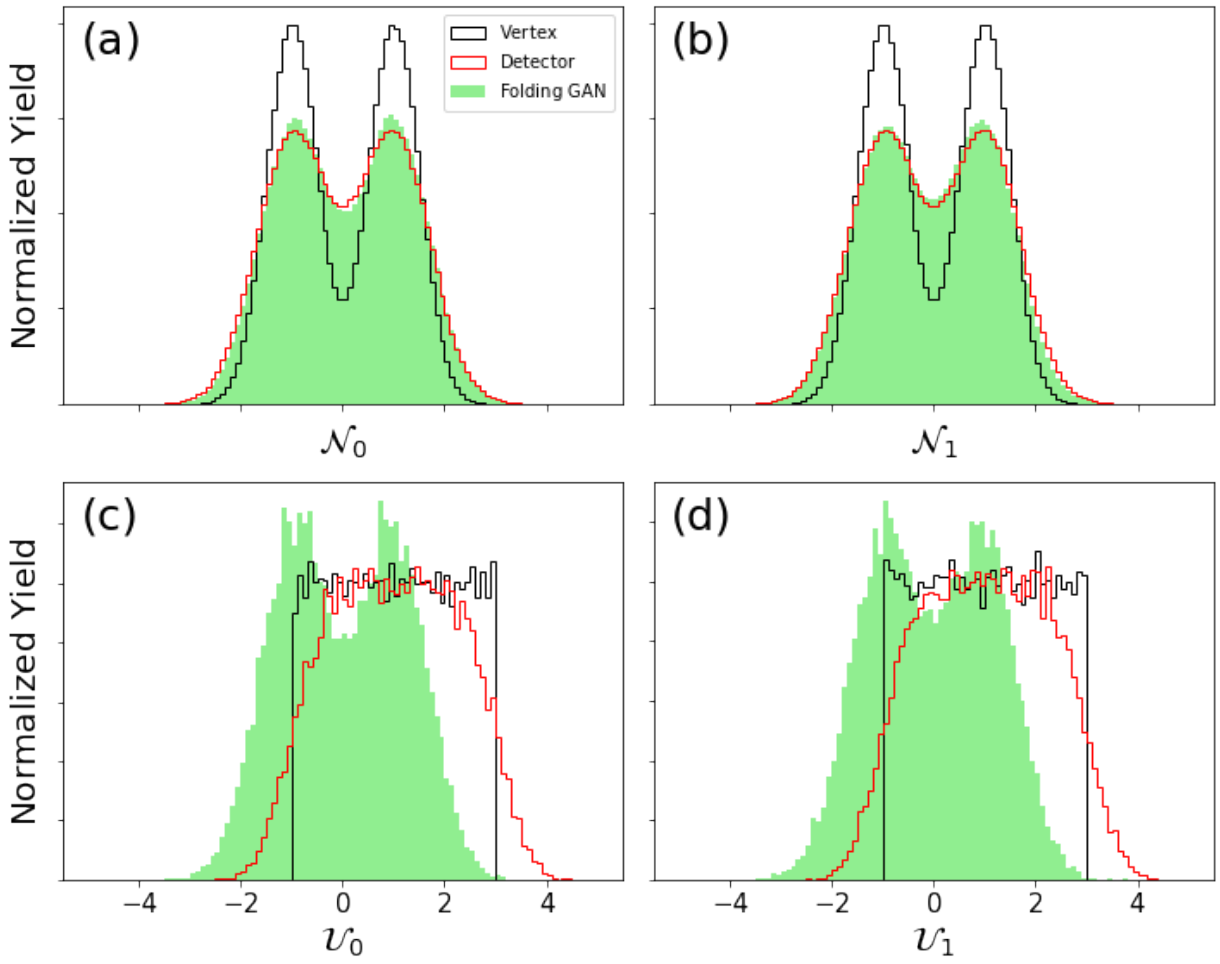


Fig. 24.: Smearing Effect on Toy Examples. The first row shows the agreement between the training detector-level data (red histograms) and the generated data by GAN (green shaded histograms). The bottom row shows how the GAN fails to agree with the validation detector data (red histograms) and the model always generates events similar to the training samples. Left and right columns are two different features.

complex scenarios. For this we use two statistically independent examples *toy*1 and *toy*2 that are used for training and validation respectively.

We first show the results of our *direct folding* approach by training on *toy*1 example and test on the same example to make sure this technique fold events coming from the same distribution. As shown in Fig. 24. (a) and (b), the *direct folding* results (green shaded histogram) are in a good agreement with detector-level data (red histogram) when we train and test on *toy*1 example, where (a) and (b) are two different features. In addition, we have also trained and tested the model on several other distributions and we can report that this simple approach with proper training and architecture designs can overall map any distribution to another as long as the test data is part of the training data.

To have a realistic and generic approach we need a framework that can learn the detector effects and is independent of the training data to satisfy the detector effects constraints. To test this, we feed *toy*2 example to the model at inference time and see how the model can fold events that did not see in the training. We can see in Fig. 24. (c) and (d), the model results (green shaded histograms) generated results in a way similar to the folding behaviors exist in the training samples, and the obtained results are more like a Gaussian than a uniform distribution.

6.4.2 DIRECT FOLDING LIMITATIONS

After the evaluation of our *direct folding* architecture, we can see that this approach learns a deterministic link between inputs (vertex-level) and outputs (detector-level), and works very well as a distribution mapper when we map from one distribution to another and test on samples coming from the same distributions; however, *direct folding* fails to fold events that are not part of the training data. This simple approach can perform very well if the training and testing data are statistically equivalent as shown in Fig. 24. (a) and (b) where the folding GAN results are in a good agreement with detector-level data; however, this direct mapping approach failed to fold events that are not similar to the training samples. As shown in Fig. 24. (c) and (d) when we feed uniform distributions to the model trained on Gaussians, the folding GAN will always generate results similar to the training data which is in this case Gaussians distributions.

6.5 CONDITIONAL FOLDING GAN

To address the limitations of the *direct folding* approach and satisfy the physics folding constraints discussed in 6.2, we implemented a generic *conditional folding* GAN architecture

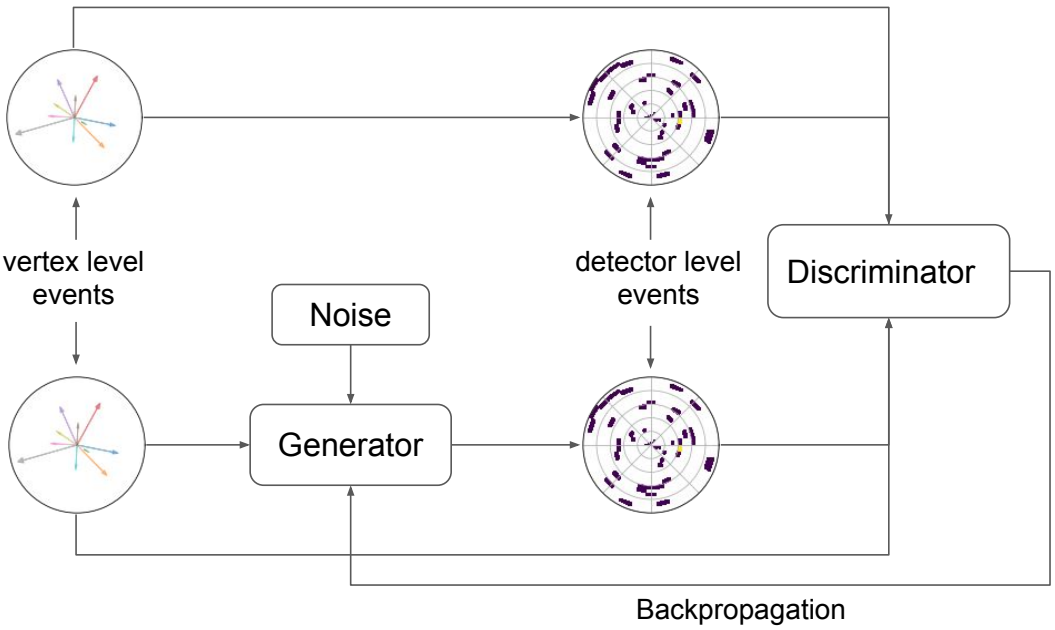


Fig. 25.: Conditional Folding GAN setup: The input to the generator is white noise and vertex-level events. The generator produces the detector-level events which is concatenated with the vertex-level events and passed to the discriminator.

that can fold events that are not part of the training samples. As shown in Fig. 25., we have a generator that receives vertex-level as input in addition to a 100-dimensional white noise centered at 0 with unit standard deviation. The generator will learn to fold the inputs and produce detector-level events that mimic the detector response. By conditioning the model on vertex-level event features we can enforce learning the correlations between vertex and detector level events as opposed to learning a deterministic mapping between inputs and outputs.

The idea behind the *conditional folding* model is not to learn a deterministic mapping between input and output samples as in the previous *direct folding* model, instead we will use a customized layer to concatenate predicted detector-level with vertex-level events to enforce learning the correlations and exploit the fact that vertex-level and detector-level events are paired.

If the discriminator D in a GAN model is defined as:

$$L_D = \langle -\log D(x) \rangle_{x \sim P_T} + \langle -\log(1 - D(x)) \rangle_{x \sim P_G} \tag{21}$$

where P_G is the generated distribution, and P_T is the true events distribution. Then we condition the discriminator on vertex-level events, and we modify the loss to,

$$L_D \rightarrow L_D^{(C)} = \langle -\log D(x, y) \rangle_{x \sim P_T, y \sim P_v} + \langle -\log(1 - D(x, y)) \rangle_{x \sim P_G, y \sim P_v} \tag{22}$$

where P_v denotes vertex-level events, hence, the generator G loss now will take the form,

$$L_G \rightarrow L_G^{(C)} = \langle -\log D(x, y) \rangle_{x \sim P_G, y \sim P_v} \tag{23}$$

6.5.1 RESULTS

For the *conditional folding* model we first test on toy example, then proceed to more realistic examples.

6.5.2 TOY EXAMPLE

To test the devolved *conditional folding*, we start with the same toy examples used to evaluate the *direct folding* approach. We train the model on *toy*1 example which is a Gaussian distribution, then evaluate the model at inference time with uniform distributions. After training of our folding GAN for 100,000 epochs, we can see that in Fig. 26. the folding model (red lines) can fold vertex-level data (black lines) and have a good agreement with

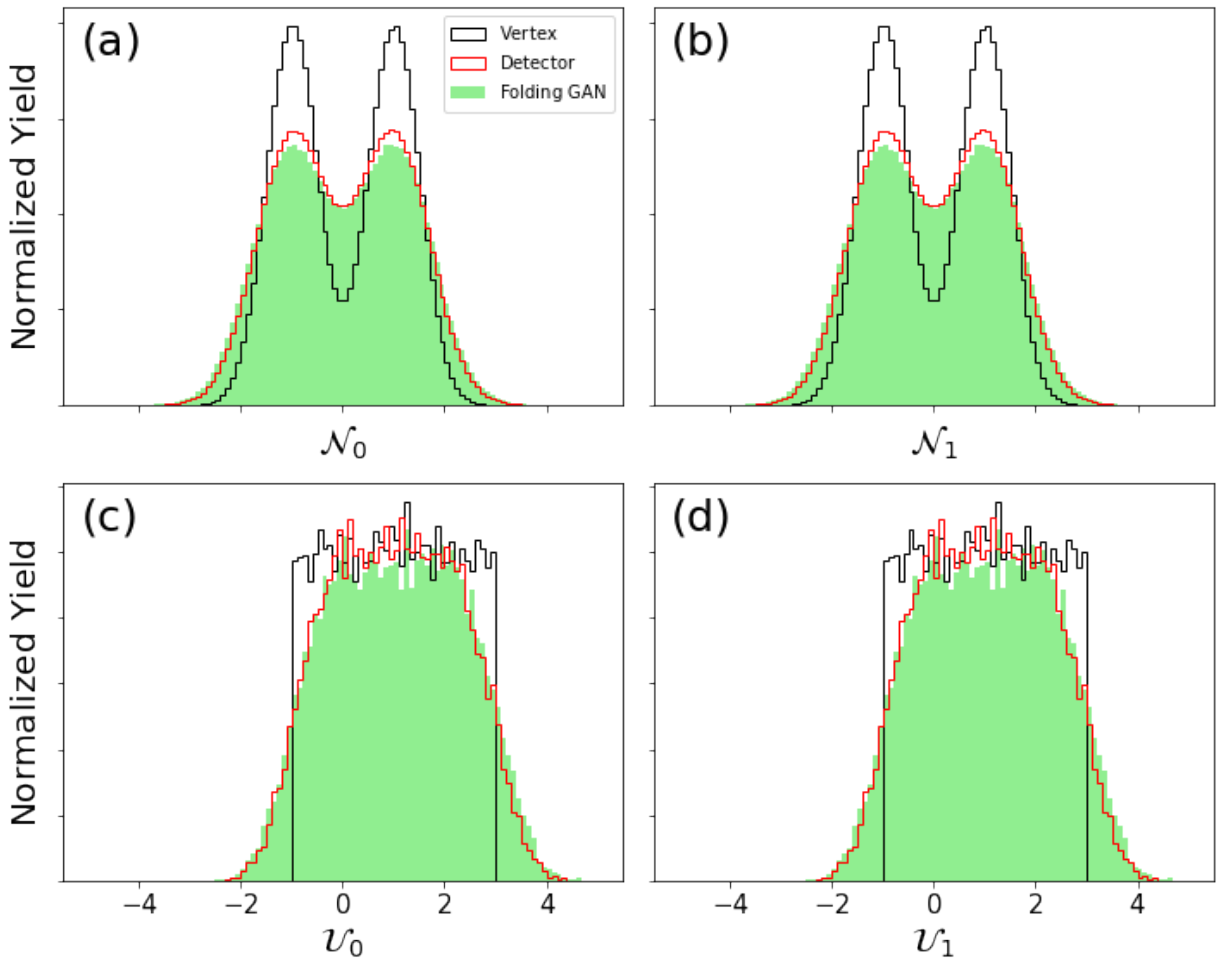


Fig. 26.: As in 24., but using the *conditional folding* model where the model can agree well with the training and validation samples.

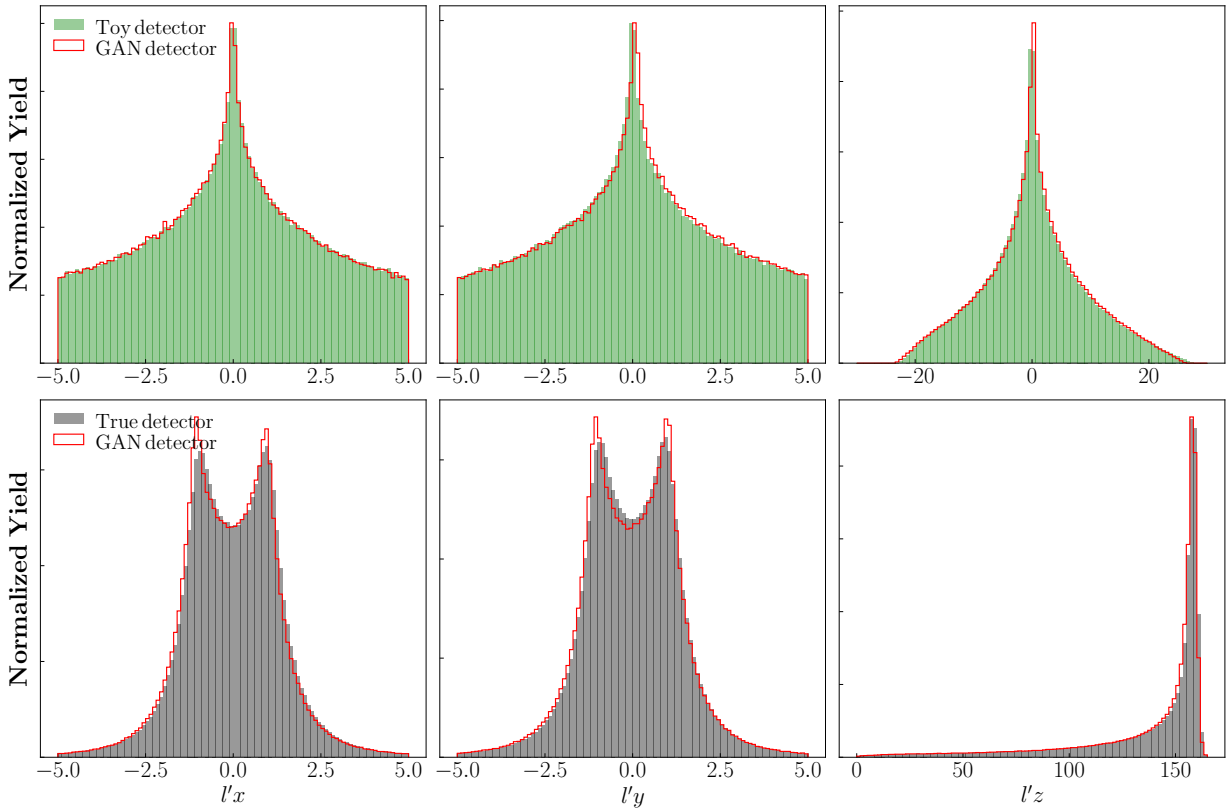


Fig. 27.: Top: 1D projections of toy PMDs generated using Gaussian distributions (training data, green) with corresponding synthetic events generated by GAN (red). Bottom: 1D projections of true detector events (validation data, gray) with synthetic events generated by the GAN (red).

detector-level data (green histograms). Similarly to what we did in acceptance effect, we also defined a uniformly distributed data that is not equivalent to the training data to test the model. The results in Fig. 26. (c) and (d) shows that the model can also fold the uniform vertex-level and match the detector-level data although the original training data was normally distributed, which indicates the model is learning that the inputs and outputs are paired.

6.5.3 INCLUSIVE DIS

We also apply our *conditional folding* methodology to the inclusive deep-inelastic scattering (DIS) reaction $l + N \to l' + X$, where l and l' are the incoming and outgoing lepton, N is the initial nucleon, and X are unobserved hadrons in the final state. So the training

data contains the three momentum of electron: $l'x$, $l'y$, and $l'z$. The purpose here is to test our *conditional folding* capability to fold events that are not part of the training data, to simulate their corresponding detector effects, and to verify that the model produces a faithful reconstruction of detector effect particle momentum distributions (PMDs). We simulate DIS events at HERA kinematics by MC sampling the theoretical cross section computed in collinear factorization at next-to-leading order in perturbative QCD. The theoretical cross section depends on nonperturbative parton distribution functions, which are extracted from the JAM global QCD analysis. The simulated DIS events are then passed to a detector simulator [84] that smears the events.

We train our model using toy vertex-level events and their associated detector-level events. In Fig. 27. (top row) we present 1D projections for the toy DIS PMDs showing the smearing effects induced by the detector simulator that distorts the vertex-level distributions. The folding GAN is able to reproduce correctly the smearing effects as expected, since the folding GAN was trained on these event. A more stringent test for the folding is to utilize different samples for the vertex-level events and demonstrate that the resulting detector-level distributions are correctly reproduced by the folding GAN. The resulting detector-level distributions in Fig 27. (bottom row) using the folding GAN agree with the true detector distributions, confirming that the folding GAN mimics accurately the detector simulator independently of the nature of the input vertex-level events.

6.5.4 REALISTIC EXAMPLE

This section will be updated upon receiving the permission from Jefferson Lab to publish the experimental data.

6.6 CONCLUSION

In this chapter we describe two techniques to simulate detector effects: *direct folding* and *conditional folding* GANs, and we discuss the limitations of the first model and how the second one can be more generic and overcome and satisfy some of the physics folding constraints. In the next chapter, we integrate the *conditional folding* procedure with a ML-based generator model to reconstruct vertex-level events (unfolding).

CHAPTER 7

VERTEX-LEVEL EVENTS CONSTRUCTION

In this chapter we integrate the *conditional folding* procedure developed in 6 with a GAN model to reconstruct vertex-level events. This work addresses **RQ3:** Can we build a ML-based event generator framework to reconstruct vertex-level events?, and will be published in Physical Review D [20]. We begin the discussion in Sec. 7.2 with a schematic overview of the MLEG training with our GAN-based model. This is followed in Sec. 7.3 by a description of the ML *conditional folding* that we use in order to simulate the effects of real particle detectors. The application to inclusive electron-proton DIS is discussed in Sec. 7.4, where we examine GAN training both with and without detector effects. We finally, summarize our findings in Sec. 7.5.

7.1 INTRODUCTION

In this work we integrate the *conditional folding* with a GAN model to present an ML-based event generator (MLEG) using generative adversarial networks, which have been increasingly utilized recently in high-energy physics applications as a tool for fast Monte Carlo simulations [68, 5, 6, 7, 8, 9, 69]. A crucial feature of GANs (as well as generative models in general) is their ability to generate synthetic data by learning from real samples without explicitly knowing the underlying physical laws of the original system. We present a case study for inclusive deep-inelastic scattering (DIS) with realistic pseudodata generated from phenomenological models. We first train the MLEG that can faithfully reproduce the phase space of inclusive DIS along with uncertainty quantification (UQ) stemming from finite statistics and model architectures. Subsequently, we implement detector effects using an effective parametrization of detectors and train the MLEG folding models to simulated detector-level DIS events. For the first time a closure test for reconstructing vertex-level DIS events, free of theoretical assumptions, is performed.

The results provide a new opportunity for experimental data analysis to use the GAN approach to build theory-free event generators which mitigate biases induced in reconstructing physical observables from experimental data.

7.2 GAN-BASED EVENT-LEVEL CONSTRUCTION

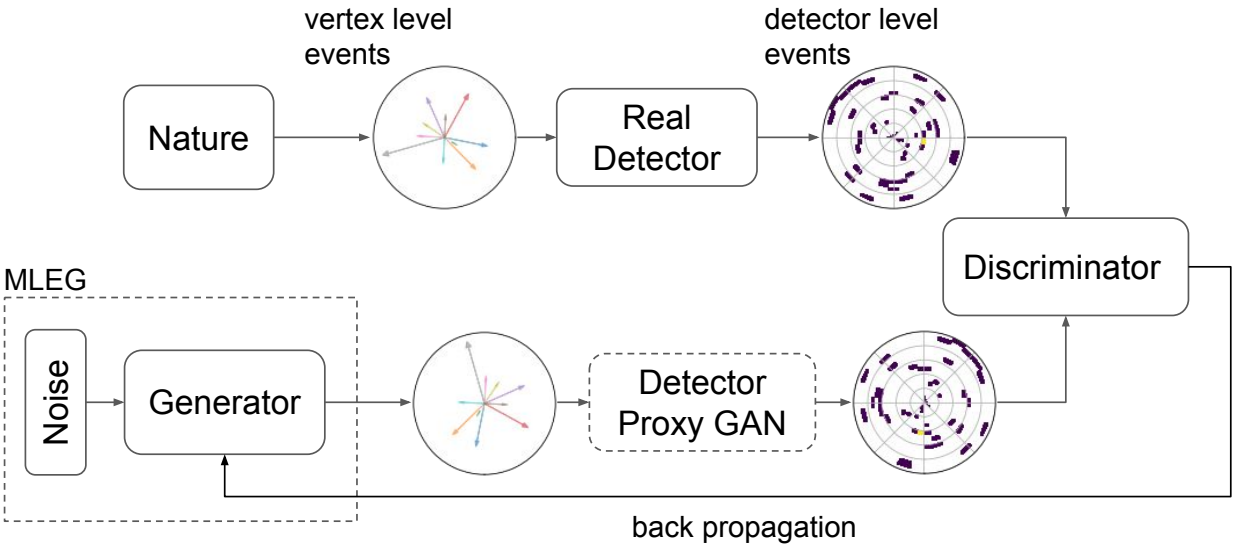


Fig. 28.: Schematic view of the MLEG GAN training framework. The MLEG (dashed box) uses a generator which transforms noise into event-level features. The generator is concatenated with a detector simulator to mimic synthetic detector-level event features. The deep neural network based discriminator compares detector-level event features in order to build gradients to update the generator of the MLEG.

A schematic view of the training workflow of our MLEG GAN is illustrated in 28., where, as usual, the GAN model is composed of a *generator* and a *discriminator*. The generator converts noise through a deep neural network into event-level features, which is customized by a given reaction. The generated event features are then passed into a detector simulator to convert them as "trial" detector-level events. The discriminator learns through another deep neural network to differentiate the true detector-level event samples from the ones produced by the generator and the detector simulator. The GAN training evolves as the generator and discriminator compete adversarially, each updating their parameters during the training process. Eventually, the generator is able to produce synthetic samples that the discriminator can no longer distinguish from the real samples, at which point the training of the MLEG is complete.

Although GANs have demonstrated impressive results in various applications, including generating near-realistic images [85], music [86], and videos [87], training a successful GAN model is known to be notoriously difficult. Many GAN models suffer from major problems, such as mode collapse, non-convergence, model parameter oscillation, destabilization,

vanishing gradient, and over-fitting due to unbalanced training of the generator and discriminator. Approaches and techniques to address these general problems have been proposed and discussed recently in the literature [45, 88, 89, 90, 91].

Unlike common GAN applications, such as the generation of realistic high resolution images, the success of our GAN application as nuclear and high-energy physics event generators relies on its ability to faithfully reproduce correlations among the particles' momenta, which are increasingly difficult in higher (greater than one or two) dimensions. At the same time, the corresponding multidimensional momentum distributions or histograms display rapid changes in the phase space that spans several orders of magnitude. The challenge is then to design suitable GAN architectures capable of reproducing all of the correlations among the particles, along with a faithful reproduction of the multidimensional histograms across the phase space. In Sec. 7.4 we will discuss in detail about how to customize this for our specific application of inclusive DIS.

7.3 INTEGRATION OF THE CONDITIONAL FOLDING PROCEDURE

Experimental data, provided in the form of final state particle momenta, are affected by distortions introduced by experimental detectors. A correction procedure is usually necessary to extract the true information from the measured cross sections and provide the vertex-level distributions used in physics analysis. Such detector effects have multiple causes, including limited acceptance, finite resolution, efficiency distortion, and bin migrations due to radiation and rescattering. Corrections are commonly taken into account using unfolding procedures that attempt to correct for the detector effects at the histogram level, requiring *ad hoc* corrections for each type of observable.

In order to demonstrate that our framework is realizable in a real experimental analysis, such detector effects must be taken into consideration. For this purpose, we use the *"eic-smear"* software package [84], which was developed at Brookhaven National Laboratory as a fast simulation tool for the future Electron-Ion Collider [92], and provides a simplified parametrization of the response of the detectors. The *eic-smear* is an open source package providing smearing capability in quantities like momentum, energy, polar and azimuthal angles. Initially, to enhance the smearing effects so they can be readily perceptible, the parameters were considerably increased. This resulted in unsatisfactory results. Reducing the smear parameters to their nominal values proved consequential in converging to the desired result.

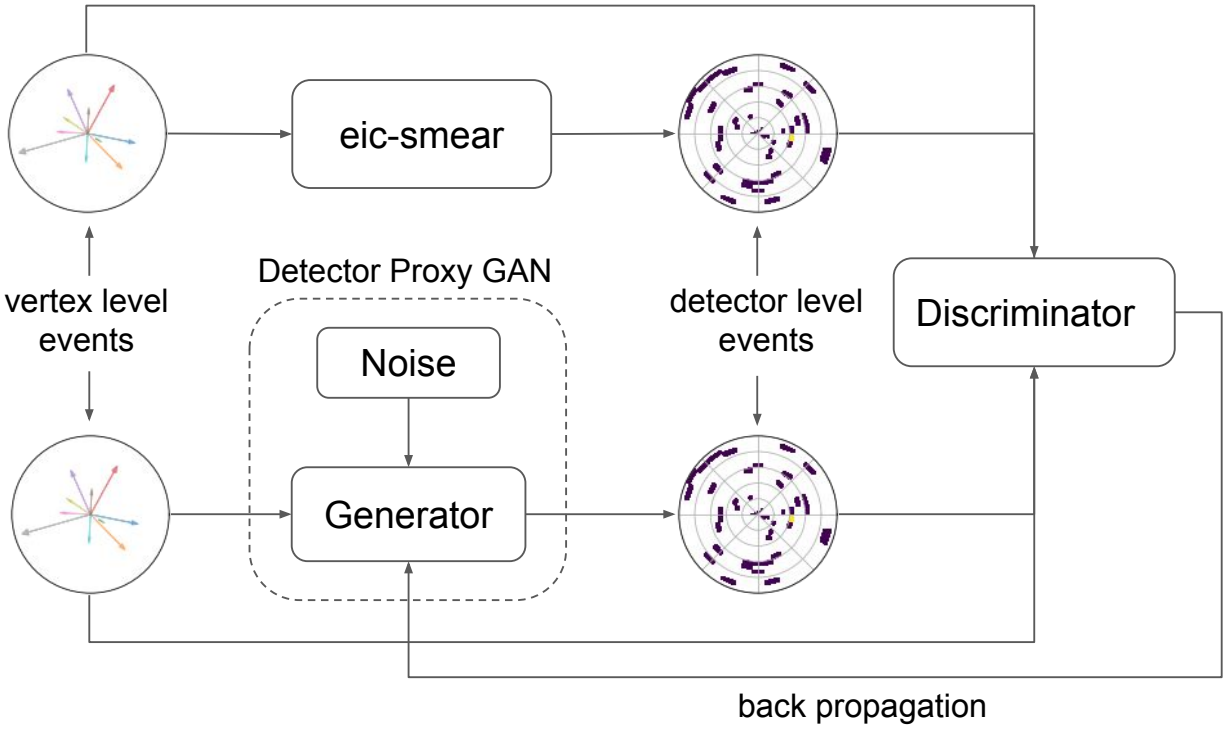


Fig. 29.: Schematic view of the ML detector surrogate, where a generator converts input vertex-level event features and noise to detector-level event features. The training samples are obtained from guess vertex-level samples and the corresponding detector-level samples using a detector simulator. The discriminator (right hand side of the figure) is trained simultaneously with vertex-level and detector-level event features in order to minimize the dependence of the generator on the input vertex-level guess samples.

We combine the *conditional folding* model, as illustrated in Fig. 29.. The idea is to train a conditional generator simulating the smearing effect of the detector by converting input vertex-level event features and noise into detector-level event features, as dictated by *eic-smear*. To do this we build training samples using trial vertex-level guess event samples and the associated *eic-smear* detector-level samples to train the *conditional folding*. Once the *conditional folding* is trained, the ML detector surrogate (represented by the dashed box in 29.) can be integrated as the detector simulator in 28.. It is worth noting that for a more realistic description of detector effects, the *eic-smear* parametrization should be replaced by a full GEANT-based [57] detector model. However, its integration within our

MLEG models using standard ML libraries is beyond the scope of the present analysis, and will be the subject of future work.

7.4 APPLICATION TO INCLUSIVE ELECTRON-PROTON SCATTERING

In this section we describe the application of our MLEG strategy to the inclusive unpolarized DIS of electrons (four-momentum k) from protons (four-momentum P). Our goal is solely to produce the scattered electron phase space, labeled by the four-momentum k'. As a surrogate for real experimental data, we use pseudodata generated from the Jefferson Lab Angular Momentum (JAM) Collaboration's QCD global analysis framework [93] that has been tuned to describe world data on inclusive DIS and other high-energy scattering processes.

The inclusive electron DIS samples are generated at a center of mass energy of 318.2 GeV, compatible with HERA kinematics, by integrating the 2-dimensional differential cross section $\mathrm{d}\sigma \,/\, \mathrm{d}x\,\mathrm{d}Q^2$, computed at next-to-leading order in perturbative QCD using importance sampling, and unweighting events over a very dense binning in (x, Q^2)-space. Each event is transformed into an outgoing electron momentum in the HERA laboratory frame by generating an azimuthal angle relative to the beam axis sampled from a uniform distribution. While our ultimate goal is to apply this approach to real data, this case study provides unique insights of our ML workflow and allows us to identify challenges in formulating a suitable feature space to be learned by the model.

When training the GAN solely using the electron momentum in the laboratory frame as event features, the generator was found to create electron samples that violate momentum conservation near the edge of the phase space, and the model was not sensitive enough to prevent the production of these samples [10]. To alleviate this problem and aid the training, we use a change of variables that enhances the discriminator awareness in these difficult regions. Specifically, we define the scaled variables

$$\nu_1 = \ln\big((k'_0 - k'_z)/1\,\mathrm{GeV}\big), \tag{24a}$$

$$\nu_2 = \ln\big((2E_e - k'_0 - k'_z)/1\,\mathrm{GeV}\big), \tag{24b}$$

where E_e is the incident electron energy, k'_0 and k'_z are scattered electron energy and longitudinal momentum, respectively. In Eqs. (24) the energies and momenta in the arguments of the log are implicitly in units of GeV. These variables can be easily inverted into the original momentum space. In particular, the variable ν_2 changes rapidly as the energy of

the outgoing electron approaches its limit, allowing the discriminator to be aware of such region.

In the following, we present details of our chosen ML architecture used for the event-level construction and the ML detector surrogate.

- **MLEG**: The input to the generator in 28. is a 100-dimensional white noise array centered at 0 with unit standard deviation. The generator network consists of 5 hidden dense layers, with 512 neurons per layer, activated by a leaky Rectified Linear Unit (ReLU) function. The number of layers and neurons are optimized to balance execution time and convergence. The last hidden layer is fully connected to a 2-neuron output corresponding to the variables ν_1 and ν_2, activated by a linear function representing the generated features. The corresponding discriminator also consists of 5 hidden dense layers with 512 neurons per layer, optimized as for the generator, and activated by a leaky ReLU function. To avoid overfitting, a 10% dropout rate is applied to each hidden layer. The last hidden layer is fully connected to a single-neuron output, where "1" indicates a true event and "0" a fake event. The discriminator D is trained to give $D(\boldsymbol{F}) = 1$ for each training sample $\boldsymbol{F}$, and $D(\widetilde{\boldsymbol{F}}) = 0$ for each sample $\widetilde{\boldsymbol{F}}$ produced by the generator.

- *conditional folding*: This model is described in chapter 6. As shown in 29. we have a generator that receives vertex-level as input in addition to a 100-dimensional white noise centered at 0 with unit standard deviation. The generator will learn to fold the inputs and produce detector-level events that mimic the detector response dictated by *eic-smear*. By conditioning the model on vertex-level event features we can enforce learning the correlations between vertex- and detector-level events as opposed to learning a deterministic mapping between inputs and outputs. As for the MLEG, the generator will produce a 2-neuron output corresponding to the detector-level variables ν_1 and ν_2, activated by a linear function representing the generated features, and the discriminator will similarly produce "0" or "1" for training and generated samples, respectively. In both the generator and discriminator architectures of the ML detector surrogate, we use the same number of hidden layers, neurons, dropout rates, and activation functions as in our MLEG. A similar idea of using GAN for detector effects has been proposed by [60], where in contrast to our folding procedure, parton-level data is mapped to detector-level data using a conditional GAN model.

For both of our GAN architectures we adopt the Least Squares GAN (LSGAN) [46],

which replaces the cross entropy loss function in the discriminator of a regular GAN by a least squares term,

$$\begin{aligned}\min_D V(D) = &\frac{1}{2}\langle (D(x)-b)^2\rangle_{x\sim P_T} \\ &+ \frac{1}{2}\langle (D(G(\tilde{x}))-a)^2\rangle_{\tilde{x}\sim P_G}\,,\end{aligned} \tag{25}$$

$$\min_G V(G) = \frac{1}{2}\langle (D(G(x))-c)^2\rangle_{x\sim P_G}\,, \tag{26}$$

where P_G denotes the distribution of the generated samples and P_T the distribution of the training samples. As a result, by setting $b-a=2$ and $b-c=1$, minimizing the loss function of LSGAN implies minimizing the Pearson χ^2 divergence. For the conditional model, the objective functions can be defined as

$$\begin{aligned}\min_D V(D) = &\frac{1}{2}\big\langle (D(x|y)-b)^2\big\rangle_{x\sim P_T,\, y\sim P_v} \\ &+ \frac{1}{2}\big\langle (D(G(\tilde{x}|y))-a)^2\big\rangle_{\tilde{x}\sim P_G,\, y\sim P_v}\,,\end{aligned} \tag{27}$$

$$\min_G V(G) = \frac{1}{2}\big\langle (D(G(x|y))-c)^2\big\rangle_{x\sim P_G,\, y\sim P_v}\,, \tag{28}$$

where P_v denotes the conditioned vertex-level samples that are fed as inputs to the ML detector surrogate. The main advantage of the LSGAN is that by penalizing the samples that are far from the decision boundary, the generator is prompted to generate samples closer to the manifold of the true samples.

Our networks are trained adversarially for 100,000 epochs, where an epoch is defined as one pass through the training data set. For the optimizer, in both cases we use Adam [94] with a 10^{-4} learning rate, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. To balance the generator and discriminator training, the training ratio is set to 5.

7.4.1 GAN TRAINING WITHOUT DETECTOR EFFECTS

As a first step in our numerical analysis, we train the MLEG using the DIS pseudo-data samples without detector effects in order to establish the baseline agreement between training and synthetic data, without the complications introduced by the detector folding. In Fig. 30. we compare the training and synthetic normalized inclusive *ep* phase space distributions for the scattered electron in the variables ν_1 and ν_2.

The uncertainty bands are generated by taking the standard deviation of 10 independently trained GANs, where for each case the training samples are prepared using the bootstrapping procedure (taking random samples with replacement). It is useful here to

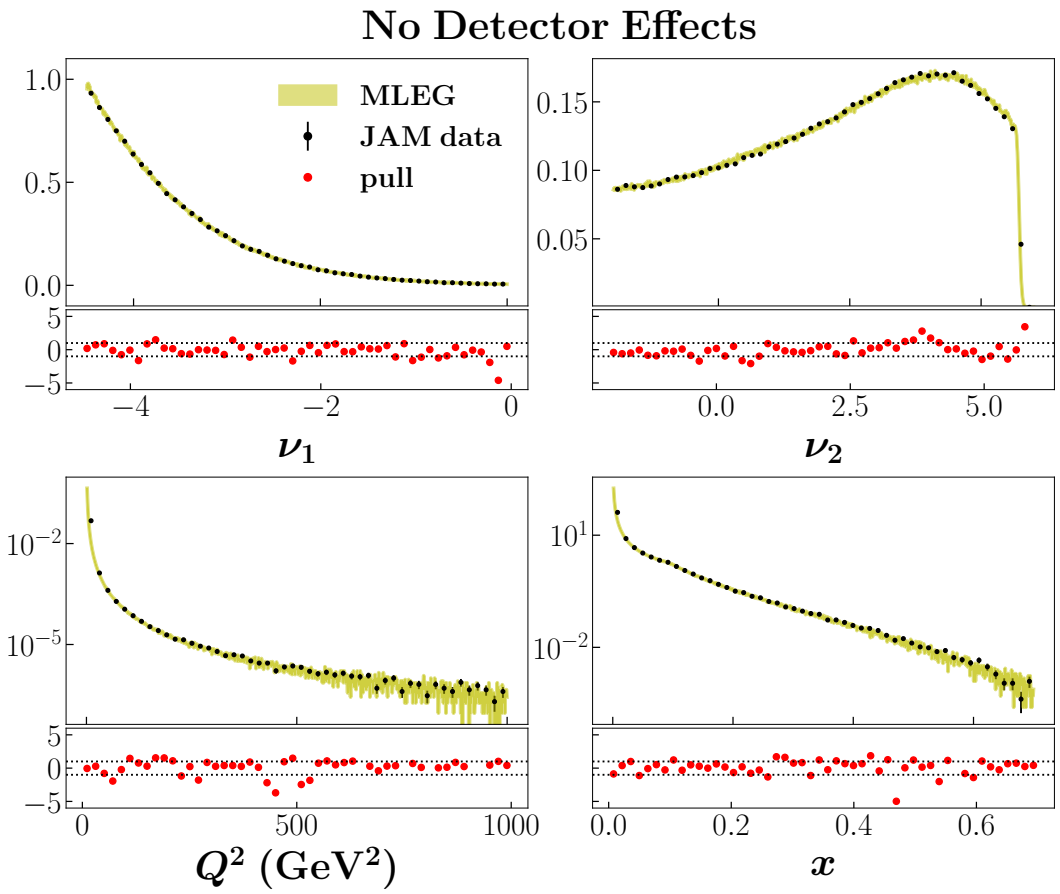


Fig. 30.: Comparison of distributions of training and derived variables from JAM training samples (black circles) and GAN-generated synthetic data (yellow bands) for the case of no detector effects; the band size reflects the uncertainty evaluated using the bootstrap procedure (see text). The bottom of each panel shows the pull distributions (red circles) defined in 29, with the two horizontal dotted lines corresponding to $\pm 1\sigma$.

define the "pull" metric between the training (JAM) and synthetic (GAN) data by

$$\text{pull} \;=\; \frac{\mathrm{E}\big[\mathcal{P}(\mathcal{O}|\text{bin})\big]_{\text{GAN}} - \mathrm{E}\big[\mathcal{P}(\mathcal{O}|\text{bin})\big]_{\text{JAM}}}{\sqrt{\mathrm{V}\big[\mathcal{P}(\mathcal{O}|\text{bin})\big]_{\text{GAN}} + \mathrm{V}\big[\mathcal{P}(\mathcal{O}|\text{bin})\big]_{\text{JAM}}}}\,, \tag{29}$$

where $\mathrm{E}[\mathcal{P}(\mathcal{O}|\text{bin})]$ and $\mathrm{V}[\mathcal{P}(\mathcal{O}|\text{bin})]$ are the expectation values and variances of the discrete probability density $\mathcal{P}$ of an observable $\mathcal{O}$. As expected, the synthetic distributions for ν_1 and ν_2 match well with the distributions from the training samples, within the statistical uncertainties, since for these variables the deviation from the training set is explicitly disfavored by the discriminator. Also shown in Fig. 30. are distributions of derived quantities that

are physically relevant for the DIS process, namely, the four-momentum transfer squared, $Q^2 = -(k - k')^2$, and the Bjorken scaling variable $x = Q^2/2P\cdot(k - k')$. While these observables are obtained by nonlinear transformations of the original variables ν_1 and ν_2, the result accurately reconstructs the matching, within uncertainties, with the corresponding spectra from the training data.

In Fig. 31. we illustrate the reduced inclusive *ep* DIS cross section, σ_r^{ep} (in practice the reaction involved positrons scattering from protons), as a function of Q^2 in multiple bins of x for the HERA data [95] and for the parametrization of the data from the JAM global QCD analysis [93]. These are compared with the reduced cross sections reconstructed by the GAN. Within the statistical uncertainties, the empirical results are well reproduced by the MLEG simulation in most of the regions of the phase space. Note that the agreement between the JAM fit and the HERA data deteriorates at the largest Q^2 values for each fixed-x spectrum due to the vanishing of the phase space.

Nonetheless, the GAN is able to follow the pattern of the phase space distribution, such as the approximate scaling behavior, as well as the drop around the edges of the phase space. This result is quite nontrivial since the relationship between the variables that are learned (ν_1, ν_2) and the DIS variables (x, Q^2) is nonlinear, and demonstrates the ability of the GAN to learn accurately the underlying probability distribution of the phase space.

7.4.2 GAN TRAINING WITH DETECTOR EFFECTS

Having established a baseline agreement for our MLEG framework, we proceed to include detector effects, as would be in actual experimental situations, which inevitably increases the complexity of the analysis. As discussed above, we train separately an ML detector surrogate using a detector parametrization provided by the *eic-smear* software. For the trial vertex-level event samples we use directly the samples from the JAM global QCD analysis instead of the flat phase space so as to optimize the GAN training. However, we stress that in principle the model architecture for the detector surrogate can be trained with any samples.

In Fig. 32. we show the vertex- and detector-level distributions for ν_1 and ν_2, where significant distortions are observed for the latter. An issue regarding the change of variables in Eqs. (24) is that after smearing the detector-level k'_z variable can exceed the physical limit given by the incident beam energy E_e, rendering the transformation singular for those unphysical cases. However, since the change of variables, in particular for ν_2, is solely designed to increase the detector awareness in the difficult regions, we can replace E_e in

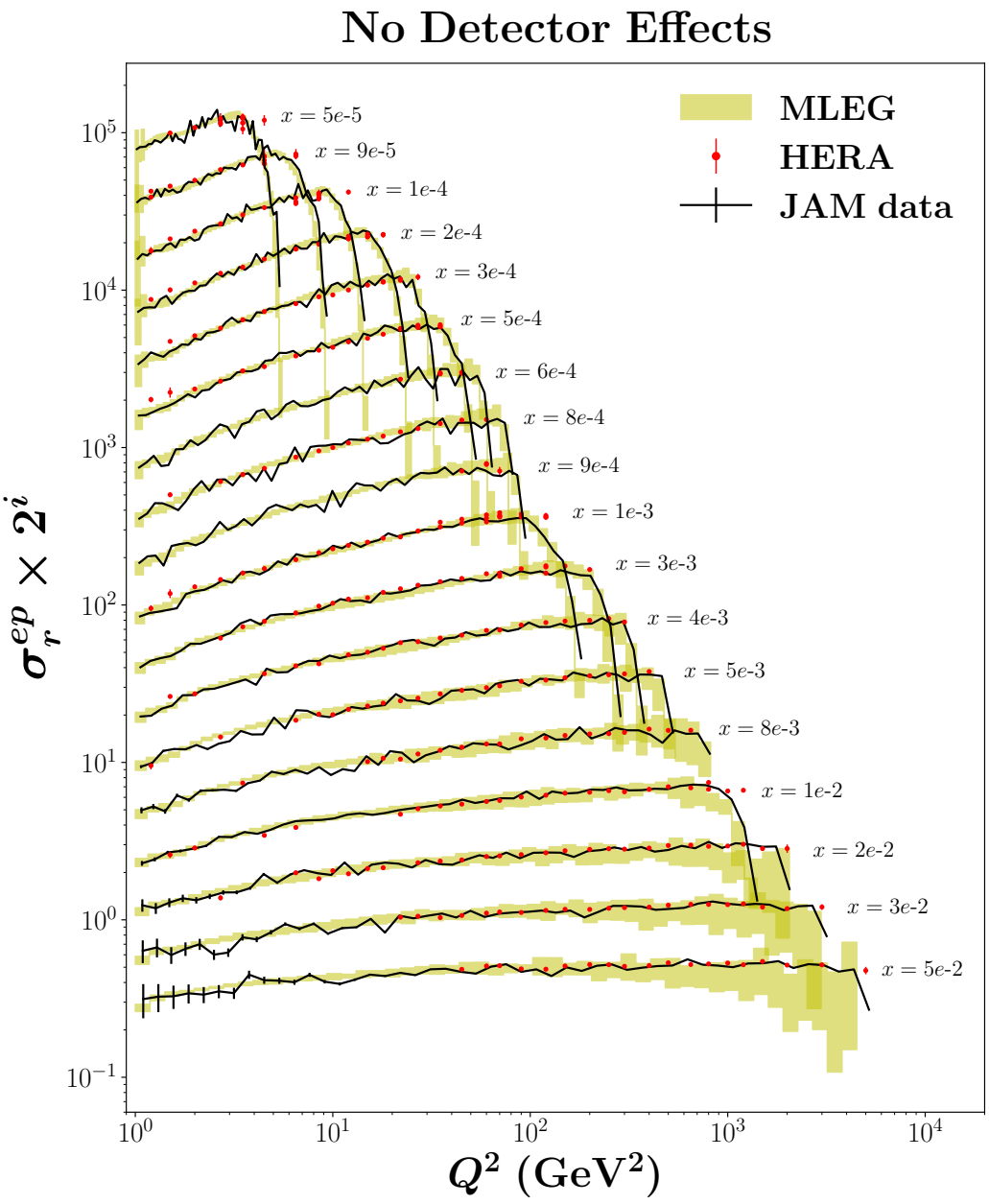


Fig. 31.: Comparison of the reduced inclusive ep cross section σ_r^{ep} versus Q^2 at fixed values of Bjorken-x from the HERA collider [95] (red circles) with data generated from the JAM global QCD analysis (black solid lines) and the trained GAN (yellow bands). No detector effects are included, and for clarity the cross sections are scaled by a factor 2^i, with i ranging from $i = 0$ for the highest-x value to $i = 17$ for the lowest-x value.

Eqs. (24) by the maximum energy found for the detector-level samples to achieve the same goal, and avoid the singularity of the variable transform. This disparity, however, creates an impression of higher levels of distortion in the ν_2 variable compared to ν_1. We also illustrate the smearing effects by focusing on small intervals in ν_1 and ν_2, as shown in the Fig. 32. insets, to indicate the nontrivial distortion that is taking place across the phase space. Included in Fig. 32. are the corresponding predictions from the detector-level GAN output, which shows very good agreement with the training samples. Note that there are regions where GANs do not match precisely with *eic-smear*, namely, the tail regions at small and large ν_2, which correspond to the edges of the reaction phase space. For the scope of this study, the GAN output represents a reasonable true detector proxy, allowing us to carry out the vertex-level learning closure test and validate the proof of principle of our MLEG framework.

With the ML detector surrogate we proceed with training the MLEG with detector effects. In Fig. 33. we show similar results as in Fig. 30., but this time with detector effects included. As expected, the variables ν_1 and ν_2 are well reproduced, since the discriminator supervises on these variables during the training. Similarly, the predicted DIS variables x and Q^2 at the detector level are well reproduced within the uncertainties. We also inverte ν_1 and ν_2 into the original momentum space and plot the 2D correlation plot between k'_x and k'_y. As shown in Fig. 35. these two inverted variables are well reproduced since they are inverted directly from the training variables.

As the final step, we examine the quality of the MLEG at the vertex level by analysing the direct output of its generator, and plot in Fig. 34. the corresponding vertex-level distributions. Relative to the detector level, the vertex-level distributions are observed to have, on average, larger values for the pull than those in Fig. 33.. This is expected since we do not directly supervise at the vertex level, but instead these are inferred quantities. A more detailed examination of this is shown in Fig. 38., where we plot the reduced cross sections as in Fig. 31., but in the presence of detector effects. As expected, the uncertainties increase due to the detector effects. We also invert the corresponding vertex-level distributions of ν_1 and ν_2 into the original momentum space and plot the 2D correlation plot between k'_x and k'_y.

One particular region where the deviations from the true result becomes larger compared to Fig. 31. are those at low Q^2 and high x (see, for instance, the low-Q^2 region for the $x = 0.01$ and 0.02 bins). The difficulty in accurately inferring this region can be traced back to the accuracy in reproducing ν_2 in Fig. 34. around $\nu_2 \sim 0$. To see this more clearly,

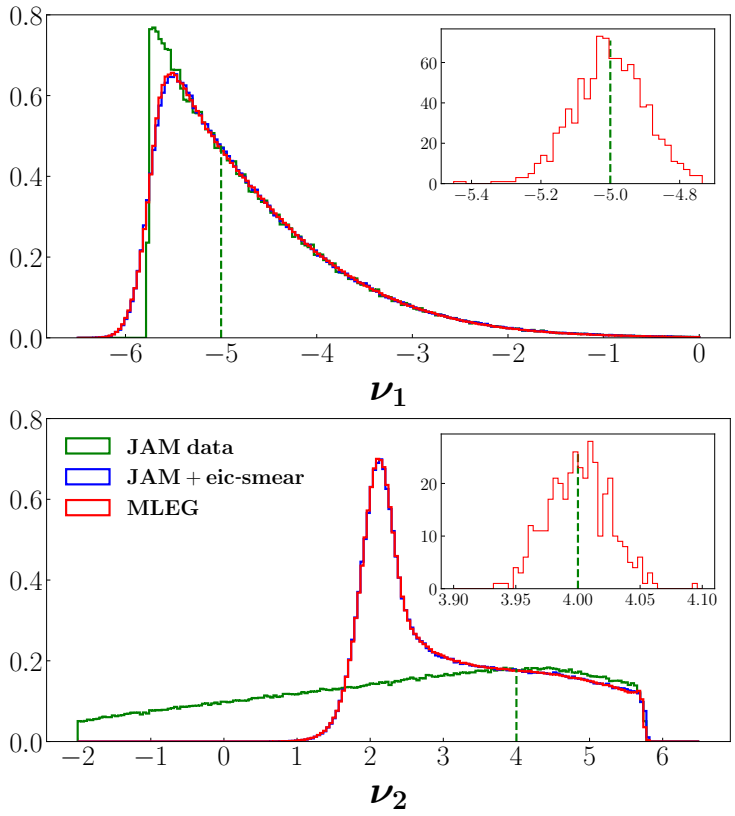


Fig. 32.: Comparison of training features at the vertex-level (generated, blue histograms) and detector level (smeared, green histograms) with the MLEG generated synthetic data (red histograms). The insets illustrate the local smearing effect at the points indicated by the green vertical dashed lines.

in Fig. 37. we plot ν_2 as a function of Q^2 at various fixed values of x. As x increases, the range of ν_2 increases covering the negative ν_2 regions for lower values of Q^2. The observed deviations in Fig. 34. around $\nu_2 \sim 0$ can be understood by observing the corresponding regions in Fig. 33. where the detector-level distribution is compatible with zero. In this case, the inference of the underlying vertex-level distribution is ill-defined.

We can understand such effect by considering the extreme scenario where the detector does not observe a particle at all, or converts the vertex-level samples into flat noise. Clearly, in such a situation the vertex-level distribution is not recoverable. In the present situation, some regions of the phase space are subjected to some degree to such extreme effects, while

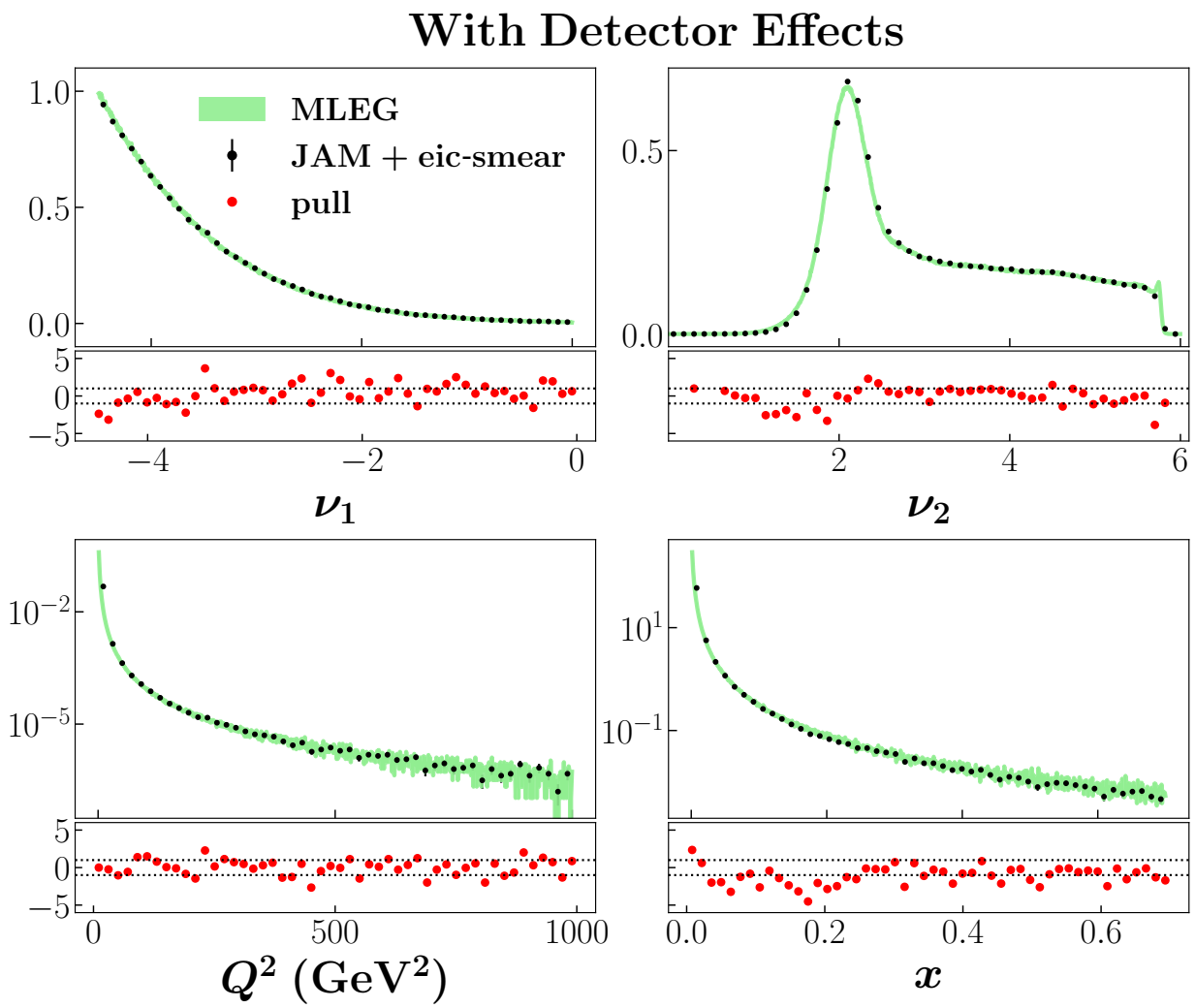


Fig. 33.: As in Fig. 30., but with Detector Effects Present

other regions are not. Other than those ill-defined regions, one can see that within the uncertainties the synthetic reduced cross sections are in agreement with the true vertex-level cross sections. This can be seen as confirmation that our MLEG training passes the closure test in the presence of detector effects.

7.5 CONCLUSION

We have presented a new approach based on generative adversarial networks to extract physics observables from pseudodata in a physics agnostic manner. To illustrate the strategy, we developed a GAN-based MLEG capable of generating synthetic data that mimic inclusive

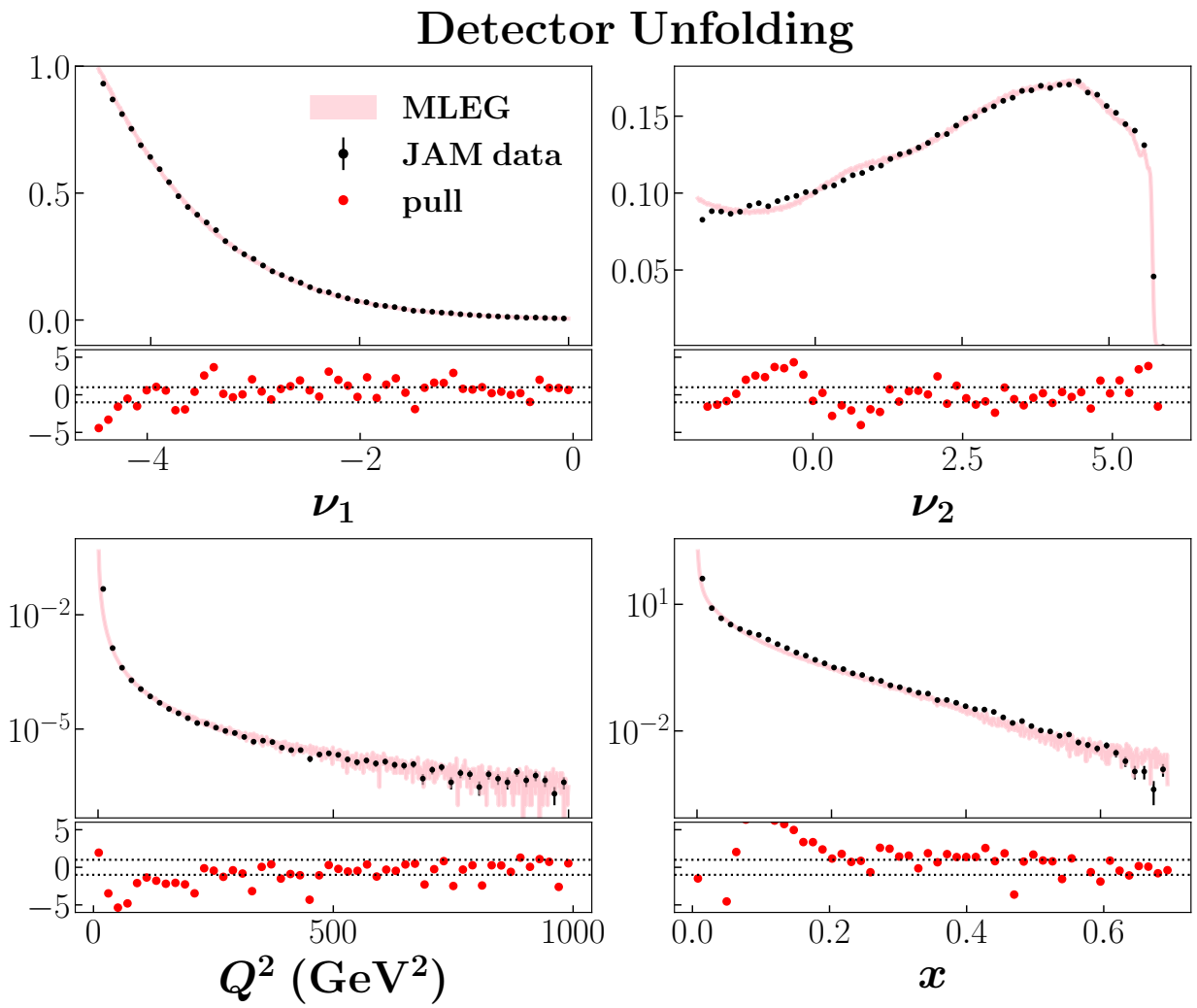


Fig. 34.: As in Fig. 30., but with All the Variables Inferred by the Unfolding procedure

deep-inelastic *ep* scattering pseudodata generated in the kinematics of the ZEUS and H1 experiments at HERA. To demonstrate the veracity of our approach we performed a closure test, extracting the original phase space distributions from synthetic particle four-momenta.

To simulate real experimental scenarios, we introduced distortions into the analysis that would be induced by a real detector, implementing a resolution smearing function, and after repeating the test obtained good agreement between original and extracted phase space distributions. Pulls quantified the uncertainty associated with the unfolding procedure, showing not only that we were able to extract the desired physics observables, but providing an uncertainty quantification for the unfolding procedure. To our knowledge this is the first time that detector effects were unfolded from pseudodata on an event basis.

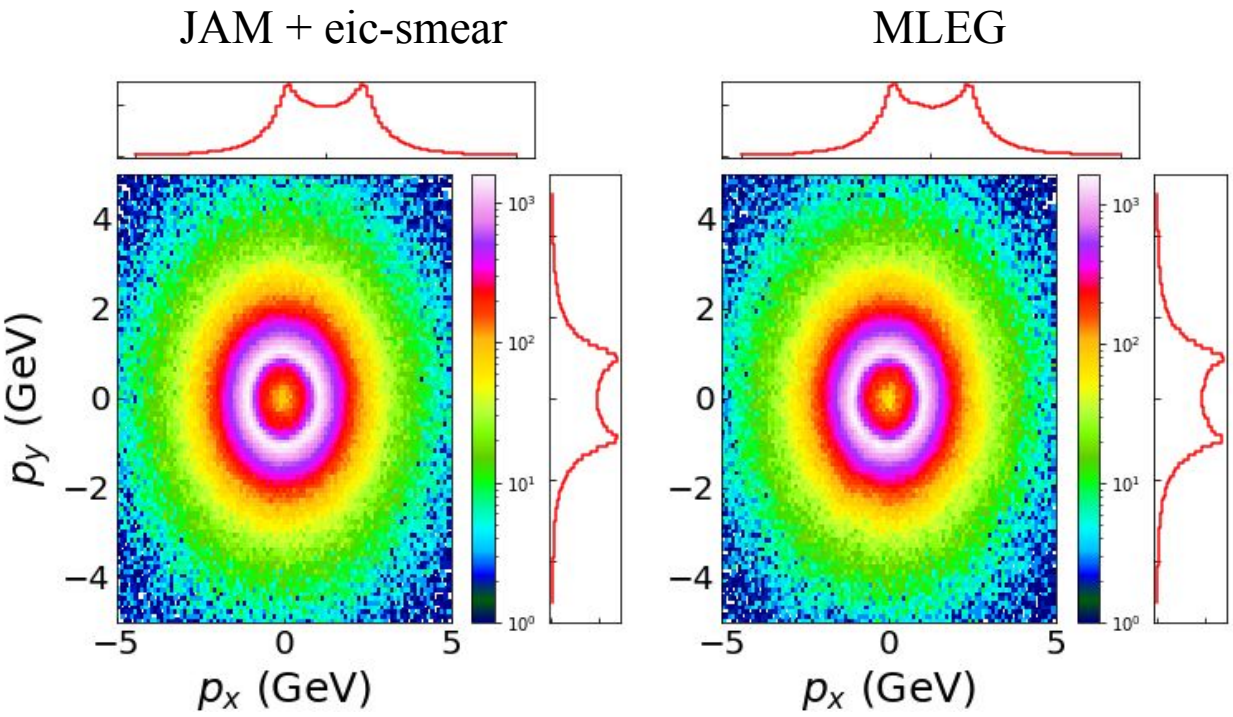


Fig. 35.: Comparison of distributions of inverted variables from JAM training samples and GAN-generated synthetic data.

The present analysis is a necessary and important proof of concept that demonstrates the viability of applying ML techniques to mitigate theoretical bias in experimental data analysis. Despite the fact that in our analysis we have effectively utilized only two-dimensional degrees of freedom to be reproduced by the MLEG, our main result is that it is possible to unfold detector effects at the event level. From the ML point of view, a larger number of particles in the final state amounts to a larger feature space. It is expected, therefore, that an extension of our proposed idea to include additional particles in the final state is feasible, provided that the number of particles in the final state remains moderate. This is case, for example, in semi-inclusive and exclusive electron-nucleon scattering.

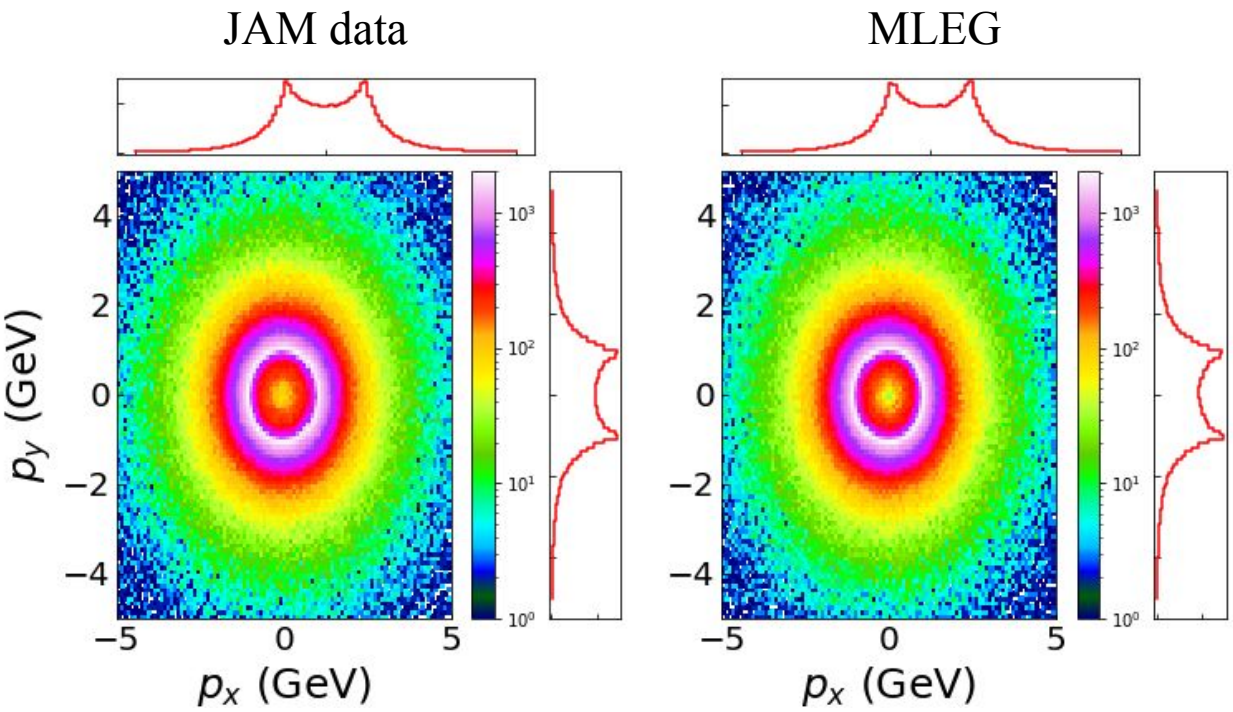


Fig. 36.: As in Fig. 35., but with All the Variables Inferred by the Unfolding Procedure

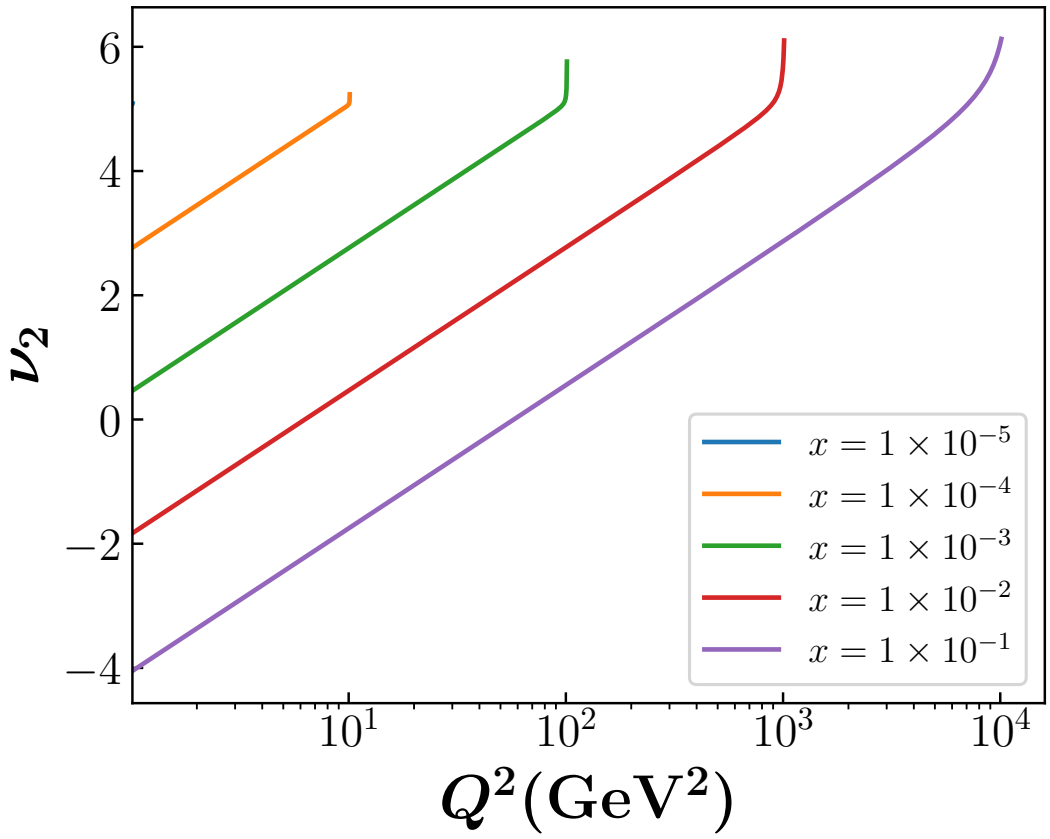


Fig. 37.: Kinematics Relation between the ν_2 Variable and Q^2 at Different Values of x.

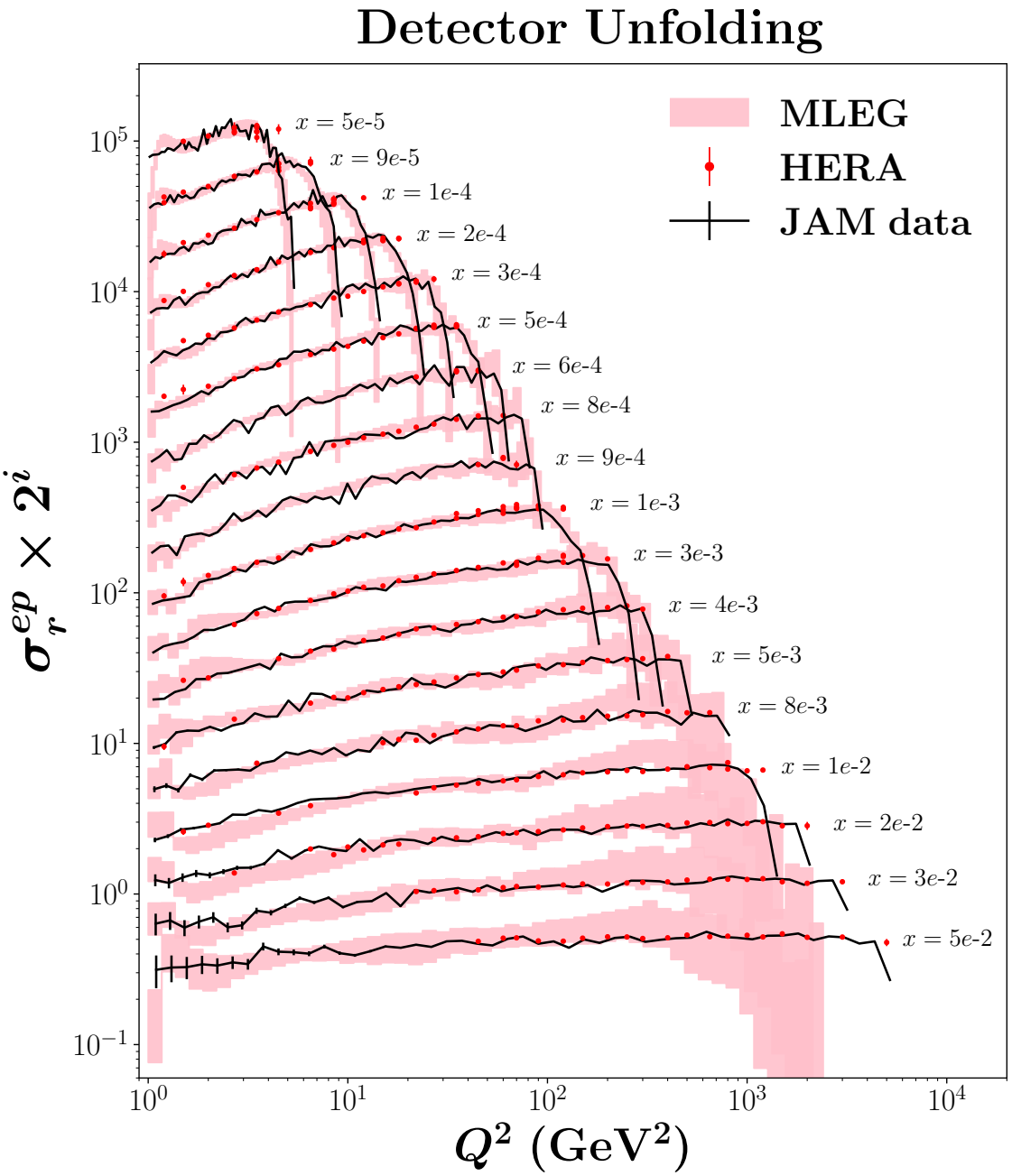


Fig. 38.: As in 31., but with the synthetic reduced cross sections generated by the GAN including detector effects and unfolding.

CHAPTER 8

EVALUATION AND INTERPRETATION

In this chapter we apply some machine learning techniques, include hidden layers analysis, dimensionality reduction, and loss landscape visualization to interpret our devolved tools. We start by exploring a non-linear dimensionality reduction transformation to understand how the hidden layers of our NNs can learn different physics reaction mechanisms. We also explore loss landscape visualization technique on toy and realistic examples and see how the loss landscape of given NNs react to different hyper-parameters and use this tool to aid in model selection and have a stable framework that generalizes well to several applications.

8.1 INTRODUCTION

ML algorithms have been defined for many years as black boxes for their complexity, and not straightforwardly interpretable to humans. In recent years, there have been efforts to produce explainable machine learning techniques that can provide more disruptive approach to ML algorithms. Several techniques are used for such interpretations, such as Local Interpretable Model-agnostic Explanations (LIME) [96], which is an explainable technique that explains the predictions of any ML classifiers. LIME tests what happens to the predictions when we provide variations of the data into the ML algorithm, it supports several data categories, including tabular and image data sets. There is also SHapley Additive exPlanations (SHAP) [97], which is another explainable technique that can explain the predictions of given ML algorithm by calculating the contribution of each feature to the prediction. SHAP can also determine the importance of of all features and their impact on the model prediction that has better consistency with human intuition than previous approaches according to the authors.

Hidden layers visualizations is another technique used to interpret NNs by visualizing the weights and output of the hidden layers and understand what information and features are being captured in each layer. Because the hidden layers outputs live in a high-dimensional space and we can only visualize 2D or 3D dimensional space, several non-linear dimensionality reduction techniques exist to bridge the gab, such as t-distributed stochastic neighbor embedding (tSNE) [76] and Uniform Manifold Approximation and Projection (UMAP) [98].

Loss visualizations is another tool used to interpret the training of different NNs. Many argue that visualizing the (1D line) loss that computes the loss value at each step (epoch) is not enough to understand how the loss surface looks like during the training and to have side-by-side comparisons with several NNs architectures; therefore, efforts have been made to visualize loss landscape of NNs by reducing the high-dimensional space of loss surface into 2D, and explain why certain NNs architectures perform better than others, such as [12].

In this work, we explore the hidden layers of our MLEG framework using t-SNE algorithm applied on the discriminator, and we also explore several NNs architectures and hyper-parameters and visualize the loss landscape using the technique developed in [12].

8.2 LAYER ANALYSIS USING T-SNE

Hidden variables in ML algorithms, such as NNs are usually not interpretable, but they interact with the observed variables. This makes it challenging for hyper-parameter tunning and model selection, and leads to incomprehension of the domain. In this section we explore our NNs hidden variables using t-distributed stochastic neighbor embedding (t-SNE) [76]. We first give a brief background of t-SNE then we show results when apply t-SNE directly on the training set and our MLEG framework.

8.2.1 T-SNE BACKGROUND

In 2008, Geoffrey Hinton and Laurens van der Maaten developed one of the well-known dimensionality reduction algorithm called t-distributed stochastic neighbor embedding (t-SNE). This algorithm is an extension of stochastic neighbor embedding (SNE) [99], where the later solves the crowd points problem by presenting the student's t-distribution instead of Gaussians which makes it easier to optimize, and to produce better visualizations. t-SNE is a non-linear dimensionality reduction technique that visualizes high-dimensional space by giving each data point a location in a lower dimensional space.

The algorithm calculates a similarity measure between pairs of points in the high and low dimensional spaces. Then t-SNE uses a cost function to optimize the similarity. As shown in Fig. 39., the left plot represents a high-dimensional space which has scattered points. For each data point, for example x_i, we center a Gaussian distribution over that point, then we measure the density of all other points under that Gaussian x_j. The circle or Gaussian distribution width can be adjusted using a hyper-parameter called perplexity. This will produce a set of probabilities $Pj|i$ for all points that represent the similarities between points. Similarly, in the lower space we measure the similarities between data points, but

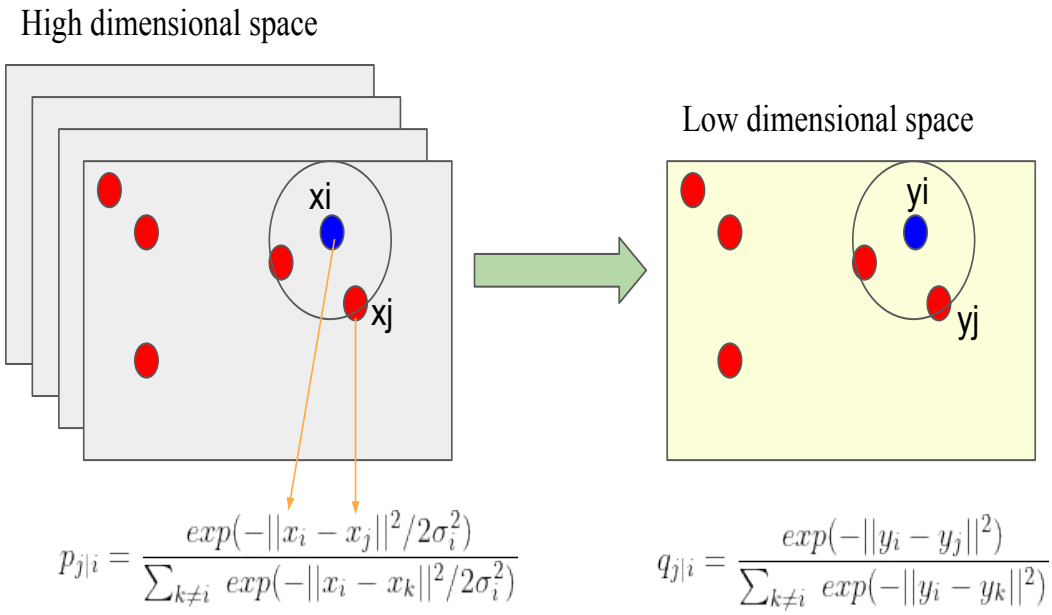


Fig. 39.: t-distributed Stochastic Neighbor Embedding (t-SNE)

instead of using a Gaussian distribution, we use a Student t-distribution that will give us another set of probabilities $qj|i$. The idea of incorporating t-distribution is because it has heavier and longer tails than Gaussian distribution which can solve the crowd problems exist in [99]. The t-SNE paper suggested a perplexity values between 5 and 30. After we compute the probabilities of low- and high-dimensional space, we want them to be as close as possible and we measure the difference between the two sets of probabilities using Kullback-Liebler divergence (KLD). Then we use gradient decent algorithm to minimize the KLD cost function, as smaller loss values generally means more preservation of the global and local structures of the data points.

8.2.2 T-SNE ON HIDDEN LAYERS

Latent variables in several ML applications show that they carry useful and interesting properties of the training data. By projecting the data into a high-dimensional space in a given NN, one might find correlations and more separations that are not observed in the actual data. Exploring the hidden layers can assist in model selection, and building a

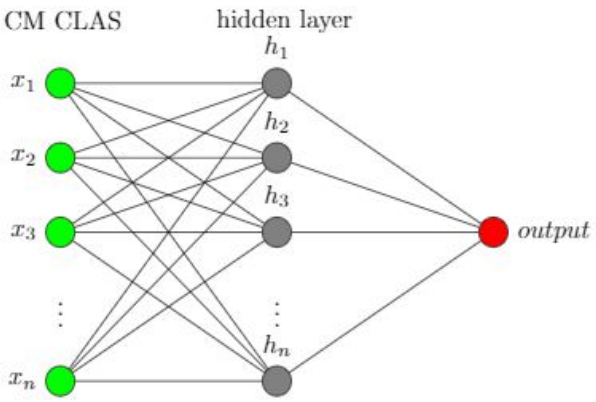


Fig. 40.: Overview of NN Hidden lLayer

successful ML event generator, which will improve our understanding of the domain.

We use MC and center-of-mass (CM) CLAS data [66] to perform t-SNE analysis on our model's hidden layers to explore different physics mechanisms. The idea of using t-SNE on the NNs hidden layers is straightforward. As shown in Fig. 40., an NN contains multiple inputs (green circles) $x_1, x_2, x_3, ...x_n$, corresponding to CM CLASS data, then the inputs go through the first hidden layer (grey circles) that contains nodes or unites $h_1, h_2, h_3, ...h_n$. The weights of this hidden layer is multiplied by the inputs, then activated by an activation function, such as LReLU before it moves froward to the next layer. For simplicity we only show one hidden layer in this figure. Finally, the last hidden layer is connected to the output layer (red circle) that produces the estimated outputs of the original inputs. In this work, we extract the hidden layers outputs of our discriminator that was trained on CM CLASS data. Each layer has 512, so we apply t-SNE to project this high-dimensional space into 2D, and perform the analysis. Projecting the last hidden layer of the discriminator into 2D We are specifically interested in the (resonance regions) that are not known and cannot not be separated from the experimental CLAS data, but we can see an approximation of those regions by visualizing 2D of $\pi^+ p$ *mass* and $\pi^+\pi^-$ *mass*, where these two variables can be calculated by taking the square root of the dot product between the momentum of proton and pion particles. In particle physics, a resonance refers to the peak located around a certain energy found in differential cross sections of a given scattering experiment. Particle physicists often use the appearance of those regions to detect new particles in scattering data. The approximated resonance regions of the CLAS data include three dominant reaction channels: $\pi^-\Delta^{++}$, ρ $(770)p$, and a diffuse background indicated by $p\pi^+\pi^-$ that effectively

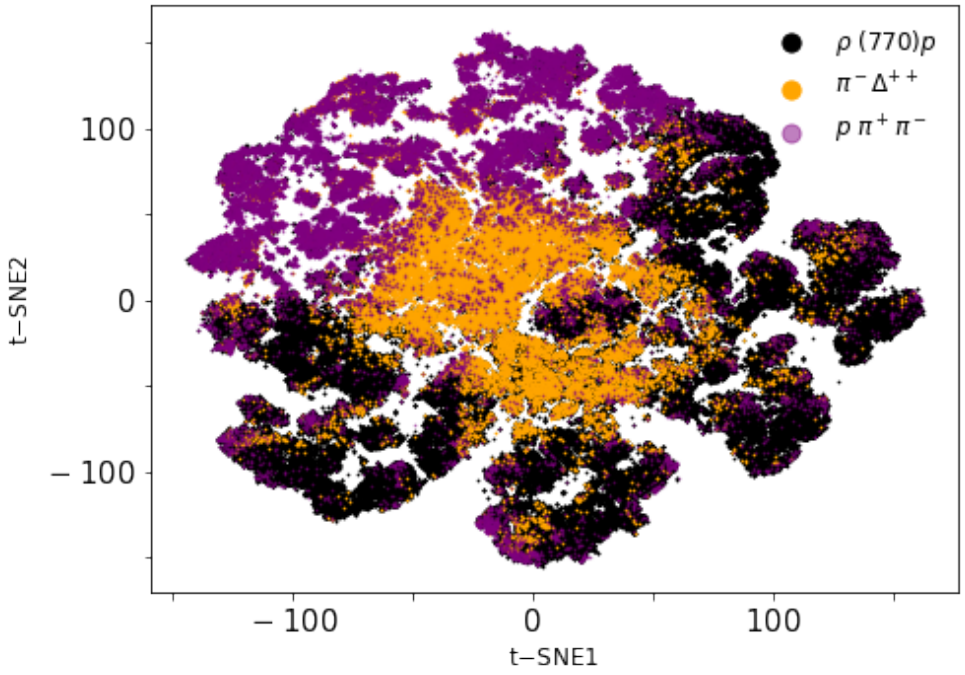


Fig. 41.: Projecting the Last Hidden Layer of the Discriminator into 2D using t-SNE

accounts for other sub-leading contributions to the reaction. Because this data was generated based on theories and assumptions, we have full control of the data and those regions are accessible and can be isolated into three different regions. Having this MC data with the corresponding labels for the resonance regions can help to tune the t-SNE algorithm and have a validation set to test our methodology.

In Fig. 41., we show the results we obtained after projecting the last hidden layer of the discriminator (512 nodes) of our MLEG into 2D using t-SNE, and color the map by the resonance labels. The hidden layer was extracted after training the model for 100,000 epochs on realistic MC data. We can see $p\pi^+\pi^-$ (purple color) is clustered on the top left of the scatter plot which indicates this region can be separated by extracting those t-SNE islands. For the $\rho\,(770)p$ region (black color) there is also clusterization where some islands are dominated by black with minor contamination from other regions, and similarly with $\pi^-\Delta^{++}$ (orange color) which has more contamination as we expect because this region does overlap with other regions.

Because t-SNE is stochastic and the embedding is initialized randomly, the layout in the projected lower dimensional space is likely to vary from one run to another and produces

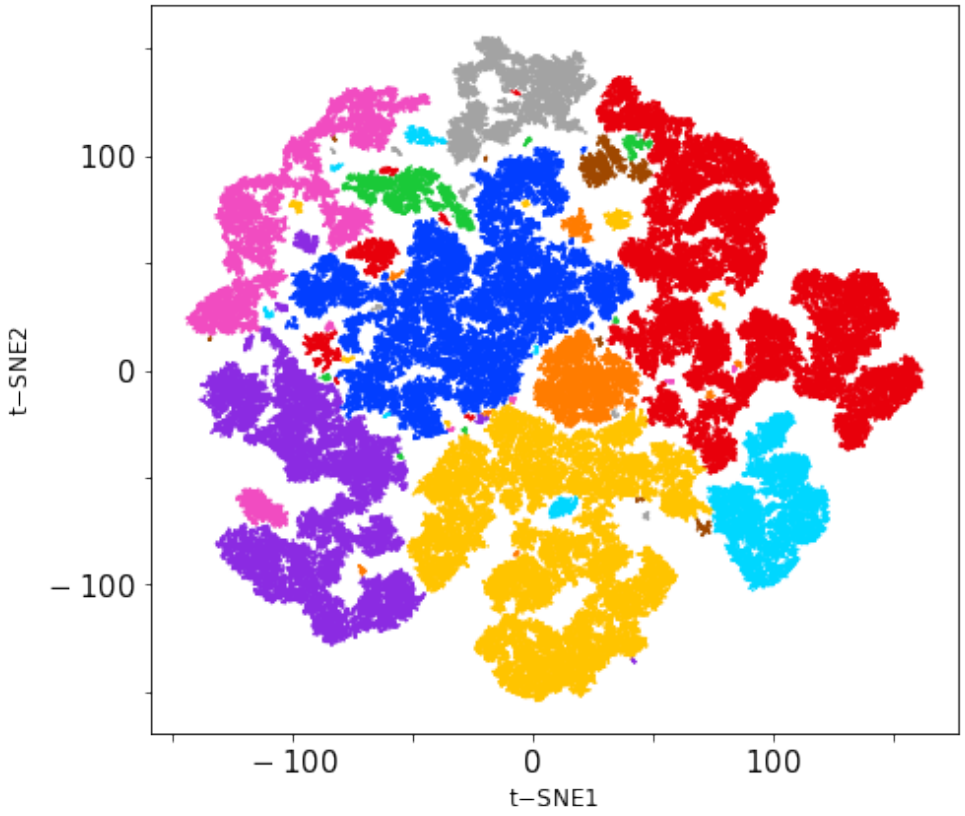


Fig. 42.: DBSCAN Clusters Correspond to Different Regions in t-SNE Map

different results, but local structures and correlations tend to be robust across multiple runs. After extensive studies and exploring several hyper-parameters, we found the algorithm to produce the similar global shapes and patterns using simulated and CLAS data for several runs with the following configurations: perplexity = 100, learning rate = 200, number of iterations = 5000, indicating the stability of the generated results. The hidden layers analysis was investigated at the discriminator level because in our case the generator hidden layers do not carry the information of the augmented features as the FAT layer, discussed in 4, is concatenated with the outputs layer of the generator. Therefore, the discriminator is more sensitive to these features. The analysis and hyper-parameters tuning were based on GPU-Accelerated t-SNE [100] for faster computational time.

t-SNE on Non-Labeled Data

In the previous section, we provided an example of performing t-SNE on the discriminator's hidden layers and we were able to validate the results. This is because MC data was used and we know the exact locations of the resonances were known. In real experimental data like CLAS the resonance regions that are not known so we will not be able to use labels to color the regions. Ideally we want to take the t-SNE map to isolate the islands on the map and then check the corresponding events. To do this we use Density Based Spatial Clustering of Applications with Noise (DBSCAN) [101], which is a ML clustering algorithm that does not require domain knowledge to determine the input parameters. The DBSCAN algorithm groups data points that are close to each other based on Euclidean distance and a minimum number of points to form a cluster. There are two main parameters need to be tuned: the first is ϵ which is the distance between two data points to be considered as neighbors, and the second is $min{-}sample$ which is the minimum number of samples to form a dense region. In Fig. 42. we show the results from DBSCAN clustering applied on the t-SNE map in Fig. 41.. For this we use $\epsilon = 0.5$, and $min{-}samples = 25$ that produced 70 clusters. We can see the regions (islands) are now colored by their corresponding clusters so we can easily extract each region and perform the analysis. To verify if the regions correspond to different physics mechanisms, we extracted the largest two clusters from Fig. 42. which are the blue and red islands located in the middle of the figure, and extracted the corresponding events to see where those events are located. We find that the events correspond to the (red) cluster map to ρ $(770)p$ resonance and the events correspond to the (blue) cluster map to $\pi^{-}\Delta^{++}$ resonance with some contaminations from other events. This concludes that the hidden layers of our GAN model carry interesting information and can separate between resonance regions when they are projected into high-dimensional space. Studying the locations of resonance regions and the ability to extract those events is a valuable contribution in particle physics as physicists often use the appearance of the resonances to detect new particles in scattering data. This also improves our understanding of the trained model and we can use this analysis to help in hyper-parameters tuning and model selection.

8.3 LOSS LANDSCAPE VISUALIZATION

The performance of NNs can be impacted by several factors, such as variable initialization, optimizer, network architecture, batch size, and other hyper-parameters. Studying the

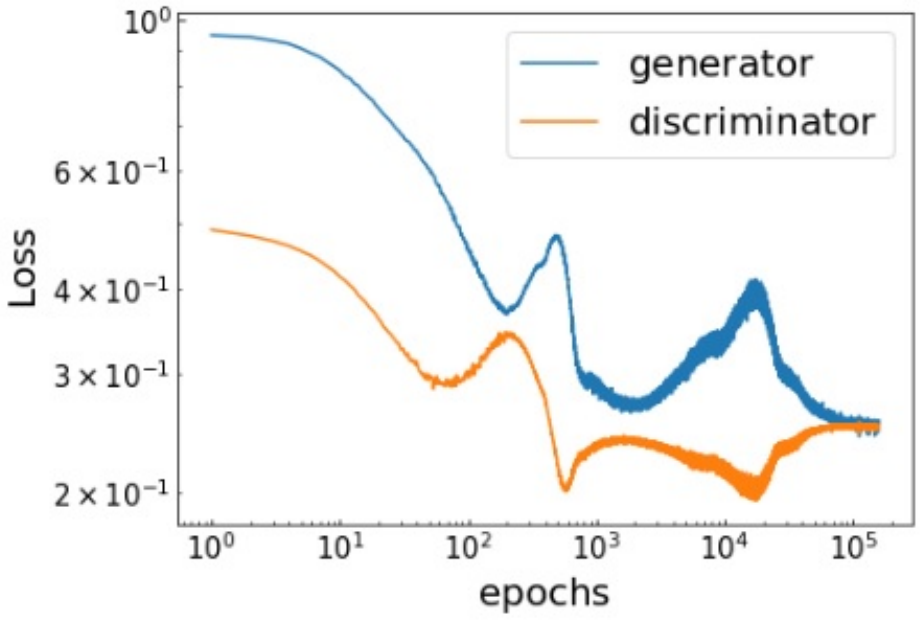


Fig. 43.: 1D Loss Curve Generated from the MLEG Trained on CLAS Data

effects of various hyper-parameters is challenging because their loss values live in a high-dimensional space. Several scientific applications rely on a simple (1D line) loss curve which is basically computing the mean/sum of the loss value for each epoch which produces a scalar for each iteration, and then plot the loss values as a function of epochs as shown in Fig. 43.. While this method is beneficial to give an overview of the model performance, it only shows a small range of gradients of the parameters, and it does not show the convexity of the function, and why certain NNs architectures generalize better than others. Recently, [12] devolved *filter normalization* technique to visualize the loss landscape of NNs that can show how convex/non-convex an NN function is, and explain why certain NN architectures generalize well while others suffer from high generalization errors. As part of hyper-parameter tuning and model selection, we applied *filter normalization* technique, devolved in [12] to visualize the loss landscape and have side-by-side comparisons for different NNs architectures and parameters.

8.3.1 FILTER NORMALIZATION

In a simple ML problem, NNs are trained on feature vectors x_i, and corresponding labels y_i, and number of data samples m, with model parameters/weights θ, and the objective

function can be defined as,

$$\mathcal{L}_\theta = \frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, \theta) \tag{30}$$

where the function $l(x_i, y_i, \theta)$ measures how well the NN can predict y_i given x_i with the model weights θ. NNs mostly contain large number of parameters and their loss functions live in a high-dimensional space. Several techniques have been proposed to reduce the loss space into 1D line or 2D surface plots, such as [102].

In this work, we use *filter normalization* technique to visualize the loss landscape of GAN NNs that can show how convex/non-convex a NN function is, and explain why certain NNs architectures with a set of parameters perform better than others. To use this approach, we choose a center point in the loss surface θ^* and choose two random direction vectors γ and ν and plot 2D surface of the form $f(\alpha, \beta) = L(\theta^* + \alpha\gamma + \beta\nu)$. Because the plots are sensitive to the scale of the model weights, as suggested in [12], we rescale the random directions using the *filter normalization* to have the same Frobenius norm of the corresponding filter (or weights in Dense layers) in θ. This technique has been used to study the effects of different NNs architectures (e.g skip connections) and to investigate sharp verses flat minimizers and how they correlate with generalization error.

Loss Surface Using Several Architectures for GANs

We apply *filter normalization* technique on several GANs architectures to understand the effects of different parameters and help in tunning the model. We are mainly interested in the GANs loss landscapes when trained using residual blocks as in Fig. 48. verses feedforward NNs (no residual blocks) as shown in Fig. 47.. One problem with GANs in general is that there is no clear correlation between the loss values and the performance of the model. Because the generator and discriminator play a min-max game and compete with each other, the loss values can have different non-intuitive patterns for multiple runs. For example, in Fig. 44. we see three different GANs model trained on CLAS data. The first model (a) was trained using 10 residual blocks layers in the generator and discriminator, and 15 residual blocks layers in the second model (b), and 15 feed-forward layers (no residual blocks) in the last model (c). An example of a residual block is shown in Fig. 46.. We can see by looking at these three models (a), (b), and (c), one cannot choose which model will perform better and generate more stable results. However, when we plot the corresponding loss surface in Fig. 45., we clearly can see different behaviours where (a) and (b) that use residual blocks seem to have very smooth loss surface, while (c) that does not have residual blocks has very

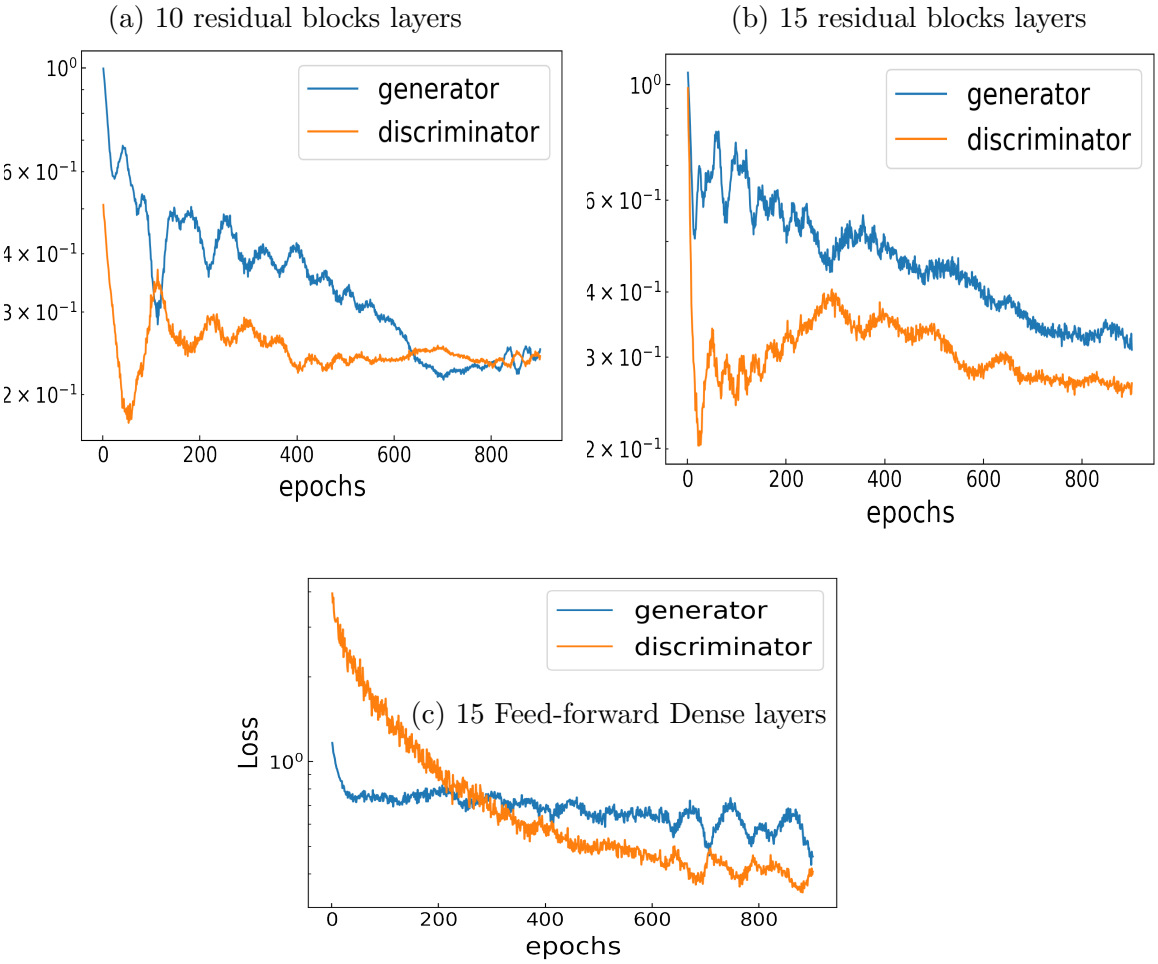
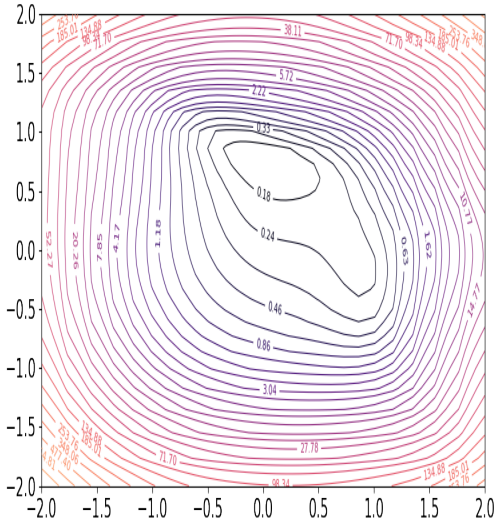


Fig. 44.: 1D Loss from Generator and Discriminator using Several NNs Architectures

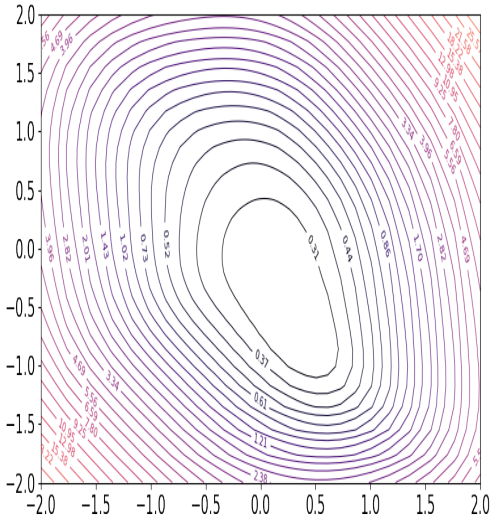
chaotic behaviour which can explain why deep NN layers with residual blocks are easier to train.

8.4 CONCLUSION

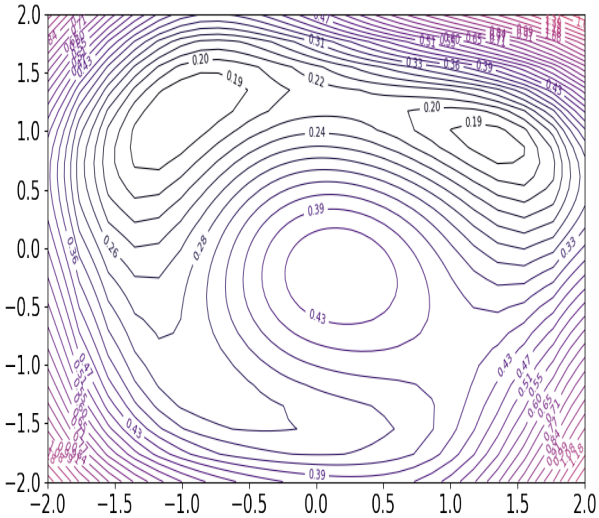
In this chapter we apply t-SNE with DBSCAN clustering algorithm to analyze the hidden layers of our GAN models to explore the possibility of extracting physics mechanisms. The results on MC and CLAS data show that when projecting the training data into high-dimensional space in the hidden layers, we can find more correlations and separations that allow for extracting the resonance regions. We also use loss landscape visualization to investigate why certain architectures and optimization algorithms perform and generalize better than others. This helps to fine-tune the model and to have a better interpretation of the devolved tools.



(a) Loss Surface, 10 Residual Blocks



(b) Loss Surface, 15 Residual Blocks



(c) Loss Surface, 15 Feed-forward Layers

Fig. 45.: Discriminator Loss Landscape using Several NNs Architectures

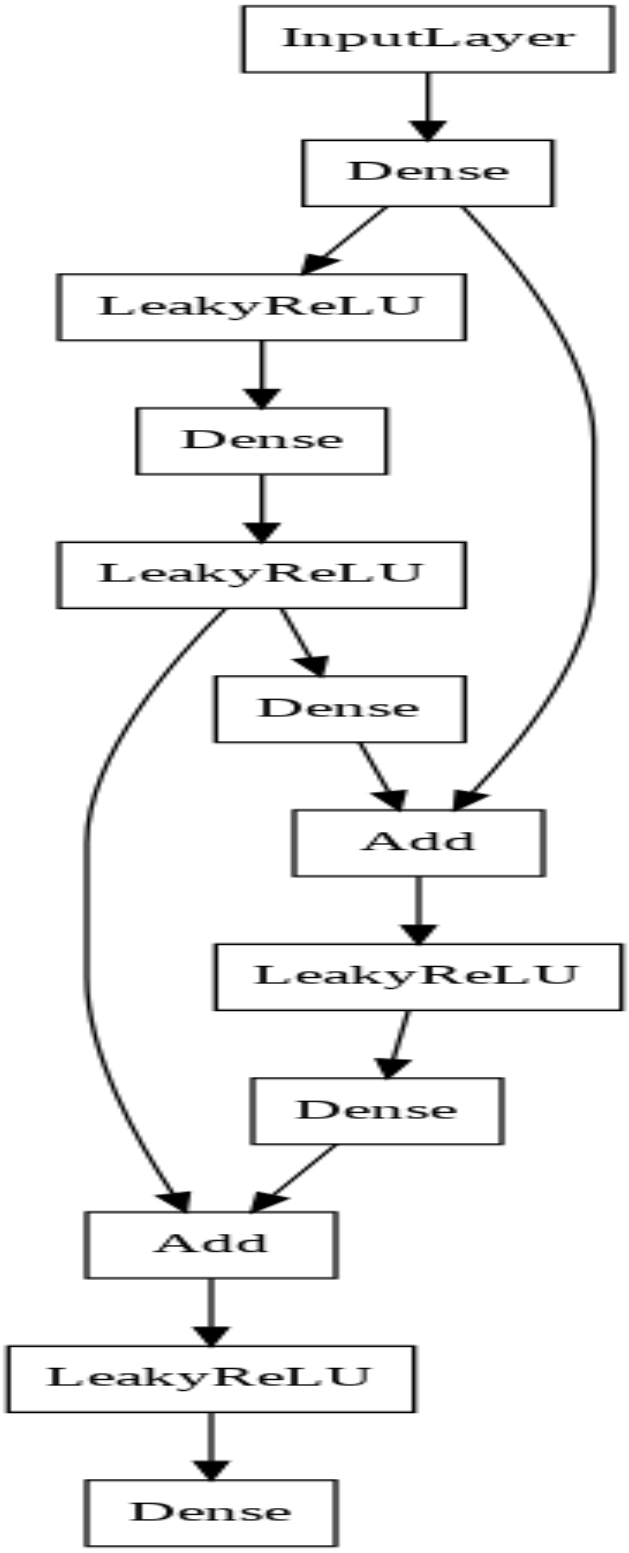


Fig. 46.: Two Residual Blocks Example

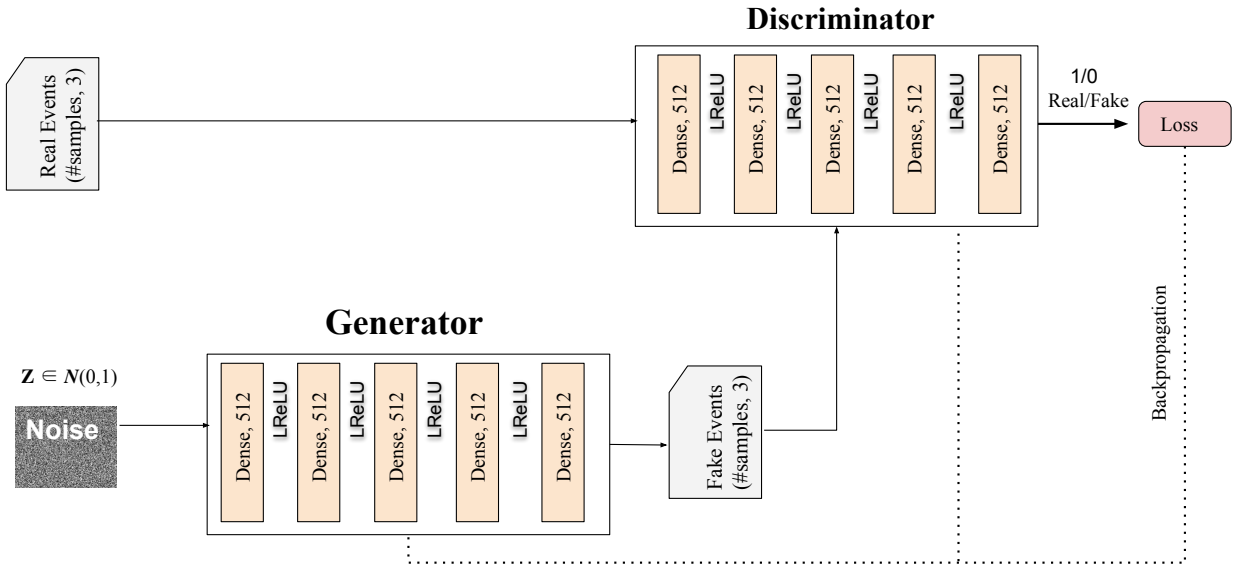


Fig. 47.: Simple GAN diagram that uses feed-forward NNs in both: the generator and discriminator.

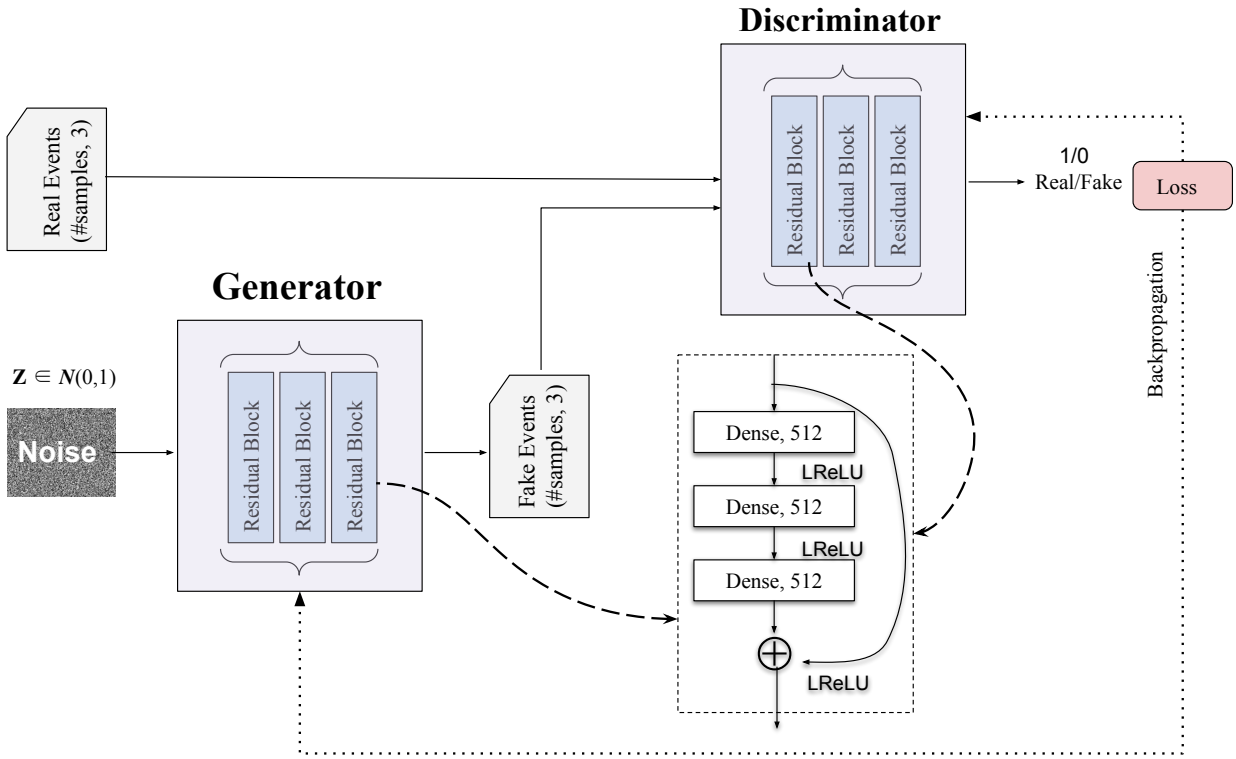


Fig. 48.: As in 47., but Using Residual Blocks in the Generator and Discriminator

CHAPTER 9

CONTRIBUTIONS, CONCLUSIONS, AND FUTURE WORK

In this chapter, we summarize the major contributions of several components for the development of a ML-based event generator framework. We expect some of our developed software to be further utilized in the future while some tools have been already utilized by Jefferson Lab experimental groups.

9.1 CONTRIBUTIONS

Our contributions include several components discussed in chapters: 4, 5, 6, 7, and 8 that are essential for devolving a successful ML-based event generator framework. The major contributions for each chapter will be summarized in the following subsections.

9.1.1 FAT-GAN

We develop a Feature-Augmented and Transformed GAN (FAT-GAN) that is able to faithfully reproduce the distribution of final state electron momenta in inclusive electron scattering. The major contribution of FAT-GAN is to incorporate physical laws into the model architecture. For this purpose, instead of directly simulating the event features, we derive a set of transformed variables from particle momenta that significantly improves the agreement of the generated results and avoids the production of unphysical events. We also modify the GAN architecture to satisfy physics constraints, including energy conservation. The FAT-GAN work was published in [10] and addresses **RQ1:** Can we build a ML-based generator to faithfully reproduce particle collision events?.

9.1.2 CFAT-GAN

We extend the FAT-GAN model to develop a Conditional Feature-Augmented and Transformed Generative Adversarial Network (cFAT-GAN) that can generate events with respect to different energy beams, even events of the energy beam are not presented in the training data. This gives the flexibility of allowing users to specify the reaction energy beam as an input to produce the corresponding synthetic events without the need to retrain the model for multiple scenarios. The cFAT-GAN not only demonstrates faithful reproduction

of events at the trained beam energies, but also correctly generates events for interpolated energies. We also analyse the generator and discriminator hidden layers to examine the sensitivity of the trained architectures with different input energies. The cFAT-GAN work was published in [11], and related to **RQ1**.

9.1.3 SIMULATION DETECTOR EFFECTS

The Major contribution of this chapter is to develop a *conditional folding* algorithm that can learn the smearing detector effects and converts vertex-level events to the corresponding detector-level events. The idea of the *conditional folding* is to concatenate vertex- and detector-level events through a customized layer to enforce learning the correlations and exploit the fact that two sets of events are paired instead of learning a deterministic mapping between vertex- and detector-level events. The folding procedure can be trained on events covering the full phase space and then tested on events covering a subset of the phase space indicating that the folding procedure generalizes better than using a direct distribution mapper. This *conditional folding* algorithm addresses **RQ2:** Can we simulate the smearing detector effects using ML tools?.

9.1.4 VERTEX-LEVEL EVENTS CONSTRUCTION

In this work, we build a new ML-based event generator framework to construct vertex-level events. The novel idea is to combine a *conditional folding* algorithm with a generator model that takes white noise as inputs and generate for the first time a closure test for reconstructing vertex-level events, free of theoretical assumptions. The framework is validated using simulated inclusive deep-inelastic scattering data, and we estimate the uncertainty of the generated results using a statistical bootstrapping technique. This work was published in [20], add addresses **RQ3:** Can we build a ML-based event generator framework to reconstruct vertex-level events?.

9.1.5 EVALUATION AND INTERPRETATION

In this work, we evaluate several architectures of the GAN models and explain why certain choice of NNs designs are easier to train and can generalize better. For this purpose, we apply *filter normalization* technique to visualize the GANs loss landscapes that can present some of the properties of the ML models in understandable and interpretable forms. This method can explore the loss surface of different models and help to justify our choice

of NNs parameters and architectures. We also analyze the GANs hidden layers by reducing the dimensionality of the layers outputs using t-SNE and use DBSCAN to cluster the t-SNE map space into several regions that we find to correspond to different physics resonances illustrating the ability of our model to differentiate between the underlying physics mechanisms.

9.2 CONCLUSIONS

Along with advances in ML methods, MLEGs are emerging as an alternative approach to MCEGs for generating simulated physical events that mimic those produced in high-energy physics accelerators. Compared to MCEGs, MLEGs demonstrate clear advantages, including fast event generation and being agnostic of theoretical assumptions. The development of MLEGs is still in its relatively early stages and most of the current efforts focus on using MLEGs as fast simulation tools with less attention to other properties, such as folding and unfolding and faithful reproductions. This motivates us to investigate several aspects of the MLEGs and examine their accurate simulations as well as examine their ability to fold and unfold particle events.

In this work, we present FAT-GAN model that incorporates physics into the GAN model. This contribution can significantly improve the ability of the model to accurately mimic simulated and experimental data and capture comblex correlations in the training features. We then extend the model to cFAT-GAN that can generate events at unrelated collision energies, and we examine the ability of the model to interpolate and extrapolate between different energy beams. We also develop a conditional folding procedure that takes vertex-level as inputs and correctly produces the corresponding detector-level events. The folding model can be trained on the full phase space and tested on any part of the phase space. The model is validated on several datasets including toy and realistic examples. After that, we integrate the folding procedure with a ML generator model based on GAN to reconstruct theory-free vertex-level events eliminating the biases introduced using existing event generator tools.

The results show a realistic proof of concept to use MLEG to simulate particle collision events. We expect that the use of our framework in data scattering will be a valuable complementary tool for nuclear and particle physics programs at current and planned facilities, such as Jefferson Lab [14] and the Electron-Ion Collider [103]. It is important to note that MLEGs are not likely to replace MCEGs, but can be a complementary tool to verify the underlying theory when compared with experimental data.

9.3 FUTURE WORK

One of the very attractive properties of generative models is super-resolution [104, 105, 106], or generating samples going beyond the resolution of its training samples. Correspondingly, in physics event generation a question that has been debated in literature is whether the generated events can add statistics beyond that of the training sample or not. That is, can the MLEG generate data that does not only reproduce the examples seen in the training data, but produces additional, diverse, and realistic samples that are more useful for downstream tasks than the original training data?

It has been claimed in [107] that, since the network does not add any physics knowledge, one can only achieve as much statistical precision as in the training sample. The main reason for this statement was that an MLEG does not learn to mimic the true event generator of the training sample, but rather the data of the sample it was trained on. This would imply that one cannot improve the model or parameter discrimination by increasing statistics with the MLEG. In addition, the statistical uncertainty of the training samples would enter the MLEG as systematic uncertainty, thus creating an even more stringent overall uncertainty than in the original data. It is stated, nevertheless, that the MLEG could still offer a better relation between accuracy and computational and storage resources than MCEGs or real experimental data.

The findings of [108] show empirically that the claim of [107] is neither well-founded nor fulfilled, thus declaring MLEG as a promising venue for the amplification of training statistics. Moreover, this work quantifies the extent to which the events can be amplified before being limited by the statistics of the training samples. The argument in favor of augmentation feasibility relies on the fact that MLEGs are powerful interpolation tools even in high-dimensional spaces. Despite being model-agnostic interpolators, the interpolation functions do fulfill basic properties such as smoothness, and therefore can add to the discrete data sets in a reliable fashion, by enabling denser binning, or higher resolution. In fact, it was shown that an MLEG can achieve the same precision as the minimal precision of a fit with a known functional form, saturating when the number of generated events becomes orders of magnitude larger than the size of the original training set. While this means that the MLEG cannot outperform the precision of a functional fit, it reassures us that in those cases where the functional form of the fitting curve is unknown (which represents most realistic physics scenarios), the MLEG becomes a powerful tool for precision augmentation.

We plan to further investigate the capability of GANs and other ML generative models to produce Super-Resolution in physics events.

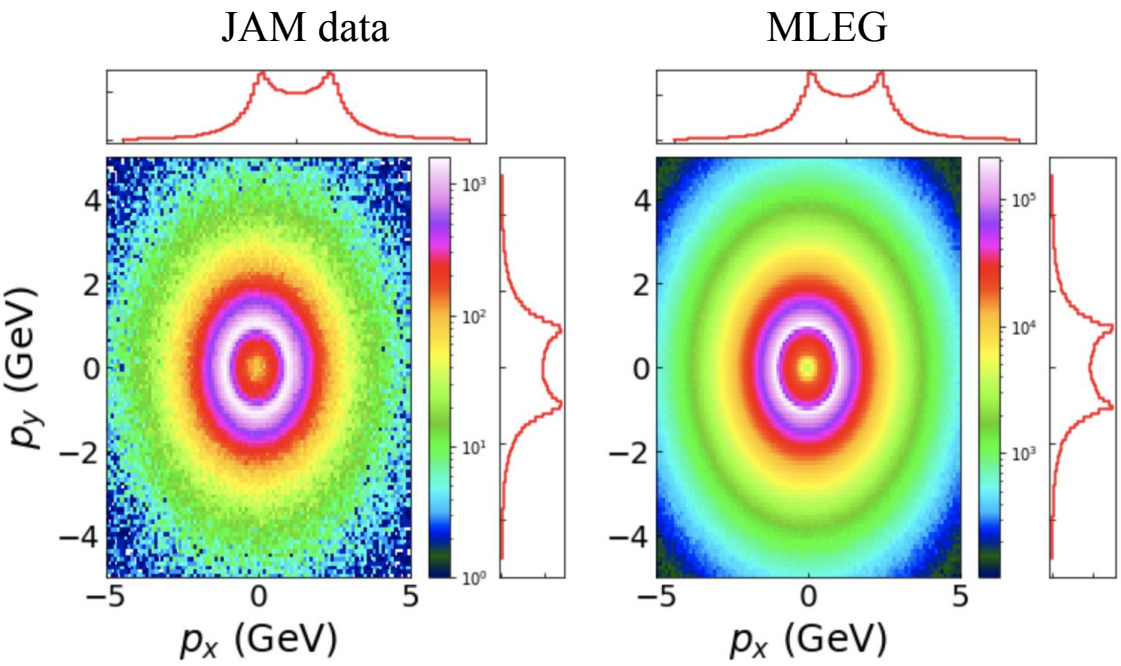


Fig. 49.: As in Fig. 36., but with ten times more samples generated by our folding procedure. Is this super-resolution?

	Publications	Status
1	Y. Alanazi, N. Sato, et al. “Simulation of electron-proton scattering events by a feature-augmented and transformed generative adversarial network (fat-gan)”. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21	Published
2	L. Velasco, Y. Alanazi, et al. “cfat-gan” in 19th IEEE International Conference on Machine Learning and Applications ICMLA-2020	Published
3	Y. Alanazi, N. Sato, et al. “A Survey of Machine Learning based Physics Event Generation”. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21	Published
4	L. Velasco, Y. Alanazi, et al. “cfat-gan”. Deep Learning Applications, Volume 3, 2021	Published
5	M. Almaeen, Y. Alanazi, et al. ”Variational Autoencoder Inverse Mapper: Deep Learning Framework for Inverse Problems”. International Joint Conference of Neural Networks IJCNN-2021	Published
6	M. Almaeen, Y. Alanazi, et al. ”Point Cloud-based Variational Autoencoder Inverse Mappers (PC-VAIM) - An Application on Quantum Chromodynamics Global Analysis”	Accepted ICMLA-2022
7	Y. Alanazi, P. Ambrozewicz, et al. Machine Learning-based event generator for electron-proton scattering. Physical Review D (2022)	Published
8	Y. Alanazi, M. Battaglieri, Y. Li, et al. A novel Artificial Intelligence for reduction and interpretation of subatomic particle production data.	To be resubmitted
9	H. Aldabagh, S. Wijerathna, Y. Alanazi, et al. Translation between Tunneling Current and Forces Images.	To be resubmitted.

TABLE 5.: Publications List

REFERENCES

[1] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[2] M. L. Mangano and T. J. Stelzer, "Tools for the simulation of hard hadronic collisions," *Annual Review of Nuclear and Particle Science*, no. 1, pp. 555–588, 2005.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, p. 2672–2680, 2014.

[4] M. Paganini, L. de Oliveira, and B. Nachman, "Accelerating science with generative adversarial networks: An application to 3d particle showers in multilayer calorimeters," *Phys. Rev. Lett.*, vol. 120, p. 042003, Jan 2018.

[5] M. Paganini, L. de Oliveira, and B. Nachman, "Calogan: Simulating 3d high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks," *Phys. Rev. D*, vol. 97, p. 014021, Jan 2018.

[6] L. de Oliveira, M. Paganini, and B. Nachman, "Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis," *Computing and Software for Big Science*, vol. 1, sep 2017.

[7] P. Musella and F. Pandolfi, "Fast and accurate simulation of particle detectors using generative adversarial networks," *Computing and Software for Big Science*, vol. 2, nov 2018.

[8] B. Hashemi, N. Amin, K. Datta, D. Olivito, and M. Pierini, "Lhc analysis-specific datasets with generative adversarial networks," *arXiv*, vol. 1901.05282, 2019.

[9] A. Butter, T. Plehn, and R. Winterhalder, "How to gan lhc events," *arXiv*, vol. 1907.03764, 2019.

[10] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, P. Ambrozewicz, F. Hauenstein, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco, and Y. Li, "Simulation

of electron-proton scattering events by a feature-augmented and transformed generative adversarial network (fat-gan)," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (Z.-H. Zhou, ed.), pp. 2126–2132, International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[11] L. Velasco, E. McClellan, N. Sato, P. Ambrozewicz, T. Liu, W. Melnitchouk, M. P. Kuchera, Y. Alanazi, and Y. Li, "cFAT-GAN: Conditional Simulation of Electron-Proton Scattering Events with Variate Beam Energies by a Feature Augmented and Transformed Generative Adversarial Network," in *19th IEEE International Conference on Machine Learning and Applications*, pp. 372–375, 2020.

[12] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[13] A. Buckley, "Computational challenges for MC event generation," *J. Phys. Conf. Ser.*, vol. 1525, no. 1, p. 012023, 2020.

[14] V. D. Burkert, "Jefferson lab at 12 gev: The science program," *Annual Review of Nuclear and Particle Science*, vol. 68, no. 1, pp. 405–428, 2018.

[15] T. Sjostrand, S. Mrenna, and P. Z. Skands, "A Brief Introduction to PYTHIA 8.1," *Comput. Phys. Commun.*, vol. 178, pp. 852–867, 2008.

[16] F. Chollet *et al.*, "Keras," 2015.

[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[18] J. Albrecht, A. A. Alves, G. Amadio, G. Andronico, N. Anh-Ky, L. Aphecetche, J. Apostolakis, M. Asai, L. Atzori, and et al., "A roadmap for hep software and

computing r and d for the 2020s," *Computing and Software for Big Science*, vol. 3, Mar 2019.

[19] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat, and S. Palazzo, "DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC," *JHEP*, vol. 08, p. 110, 2019.

[20] Y. Alanazi, P. Ambrozewicz, M. P. Kuchera, Y. Li, T. Liu, R. E. McClellan, W. Melnitchouk, E. Pritchard, M. Robertson, N. Sato, R. Strauss, and L. Velasco, "AI-based Monte Carlo event generator for electron-proton scattering," *arXiv*, vol. 2008.03151, 2020.

[21] O. M. Oderinde, S. M. Shirvani, P. D. Olcott, G. Kuduvalli, S. Mazin, and D. Larkin, "The technical design and concept of a pet/ct linac for biology-guided radiotherapy," *Clinical and Translational Radiation Oncology*, vol. 29, pp. 106–112, 2021.

[22] U. Amaldi, "The importance of particle accelerators," *Europhysics News*, vol. 31, pp. 5–9, 11 2000.

[23] M. Bahr et al., "Herwig++ Physics and Manual," *Eur. Phys. J.*, vol. C58, pp. 639–707, 2008.

[24] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert, and J. Winter, "Event generation with SHERPA 1.1," *JHEP*, vol. 02, p. 007, 2009.

[25] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, and T. Stelzer, "Madgraph 5: going beyond," *Journal of High Energy Physics*, Jun 2011.

[26] W. Kilian, T. Ohl, and J. Reuter, "Whizard—simulating multi-particle processes at lhc and ilc," *The European Physical Journal C*, vol. 71, Sep 2011.

[27] J. Weil, H. van Hees, and U. Mosel, "Dilepton production in proton-induced reactions at SIS energies with the GiBUU transport model," *Eur. Phys. J. A*, vol. 48, p. 111, 2012. [Erratum: Eur.Phys.J.A 48, 150 (2012)].

[28] G. Bíró, G. G. Barnaföldi, G. Papp, M. Gyulassy, P. Lévai, X.-N. Wang, and B.-W. Zhang, "Introducing hijing++: the heavy ion monte carlo generator for the high-luminosity lhc era," *arXiv*, vol. 1901.04220, 2019.

[29] C. Andreopoulos et al., "The GENIE Neutrino Monte Carlo Generator," *Nucl. Instrum. Meth. A*, vol. 614, pp. 87–104, 2010.

[30] C. Juszczak, "Running nuwro," *arXiv*, vol. 0909.1492, 2009.

[31] B. P. Kersevan and E. Richter, "The monte carlo event generator acermc versions 2.0 to 3.8 with interfaces to pythia 6.4, herwig 6.5 and ariadne 4.1," *Computer Physics Communications*, vol. 184, p. 919–985, Mar 2013.

[32] M. L. Mangano, F. Piccinini, A. D. Polosa, M. Moretti, and R. Pittau, "Alpgen, a generator for hard multiparton processes in hadronic collisions," *Journal of High Energy Physics*, vol. 2003, p. 001–001, Jul 2003.

[33] D. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[34] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2020.

[35] V. Blobel, "Unfolding Methods in Particle Physics," pp. 240–251. 12 p, Jan 2011.

[36] Hunt, "Bayesian methods in nonlinear digital image restoration," *IEEE Transactions on Computers*, vol. C-26, no. 3, pp. 219–229, 1977.

[37] M. Almaeen, Y. Alanazi, N. Sato, W. Melnitchouk, M. P. Kuchera, and Y. Li, "Variational autoencoder inverse mapper: An end-to-end deep learning framework for inverse problems," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

[38] Y. Alanazi, N. Sato, P. Ambrozewicz, A. Hiller-Blin, W. Melnitchouk, M. Battaglieri, T. Liu, and Y. Li, "A survey of machine learning-based physics event generation," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (Z.-H. Zhou, ed.), pp. 4286–4293, International Joint Conferences on Artificial Intelligence Organization, 8 2021. Survey Track.

[39] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. R. de Austri, and R. Verheyen, "Event generation and statistical

sampling for physics with deep generative models and a density information buffer," *arXiv*, vol. 1901.00875, 2019.

[40] C. Ahdida, R. Albanese, A. Alexandrov, A. Anokhina, S. Aoki, G. Arduini, E. Atkin, N. Azorskiy, J. Back, A. Bagulya, and et al., "Fast simulation of muons produced at the ship experiment using generative adversarial networks," *arXiv*, vol. 1909.04451, 2019.

[41] J. A. Martínez, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. V, "Particle generative adversarial networks for full-event simulation at the lhc and their application to pileup description," *Journal of Physics: Conference Series*, vol. 1525, p. 012081, Apr 2020.

[42] C. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz, "Event Generation with Normalizing Flows," *Phys. Rev. D*, vol. 101, no. 7, p. 076002, 2020.

[43] J. N. Howard, S. Mandt, D. Whiteson, and Y. Yang, "Foundations of a fast, data-driven, machine-learned simulator," *arXiv*, vol. 2101.08944, 2021.

[44] S. Choi and J. H. Lim, "A data-driven event generator for hadron colliders using wasserstein generative adversarial network," *Journal of the Korean Physical Society*, Feb 2021.

[45] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv*, vol. 1606.03498, 2016.

[46] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," *arXiv*, vol. 1611.04076, 2017.

[47] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv*, vol. 1701.07875, 2017.

[48] C. Villani, *Topics in optimal transportation*. American mathematical Society, 2016.

[49] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *ArXiv*, vol. 1704.00028, 2017.

[50] C. Li, W. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," *arXiv*, vol. 1705.08584, 2017.

[51] A. Ramdas, S. J. Reddi, B. Póczos, A. Singh, and L. Wasserman, "On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, AAAI'15, 2015.

[52] D. Blei, A. Kucukelbir, and J. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[53] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," in *International Conference on Learning Representations (ICLR 2018)*, 2018.

[54] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde, "Sliced-wasserstein autoencoder: An embarrassingly simple generative model," *arXiv*, vol. 1804.01947, 2018.

[55] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[56] A. Butter and T. Plehn, "Generative Networks for LHC events," *arXiv*, vol. 2008.08558, 2020.

[57] S. Agostinelli et al., "Geant4-a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A 506(3), pp 250*, 2003.

[58] S. Ovyn, X. Rouby, and V. Lemaître, "Delphes, a framework for fast simulation of a generic collider experiment," *arXiv*, vol. 1307.6346, 2009.

[59] T. T. Böhlen, F. Cerutti, M. P. W. Chin, A. Fassò, A. Ferrari, P. G. Ortega, A. Mairani, P. R. Sala, G. Smirnov, and V. Vlachoudis, "The FLUKA Code: Developments and Challenges for High Energy and Medical Applications," *Nucl. Data Sheets*, vol. 120, pp. 211–214, 2014.

[60] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, "How to gan away detector effects," *arXiv*, vol. 1912.00477, 2020.

[61] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, "Generating and refining particle detector simulations using the wasserstein distance in adversarial networks," *arXiv*, vol. 1802.03325, 2018.

[62] G. Adams *et al.*, "The CLAS Cherenkov detector," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 465, no. 2, pp. 414–427, 2001.

[63] F. Scarselli and A. C. Tsoi, "Universal Approximation Using Feedforward Neural Networks: A Survey of Some Existing Methods, and Some New Results," *Neural Networks*, vol. 11, no. 1, pp. 15–37, 1998.

[64] Q. Liu, H. Lv, and R. Jiang, "hicGAN infers super resolution Hi-C data with generative adversarial networks," *Bioinformatics*, vol. 35, no. 14, pp. i99–i107, 2019.

[65] Y. Li, Y. Ni, R. A. C. Croft, T. Di Matteo, S. Bird, and Y. Feng, "Ai-assisted superresolution cosmological simulations," *Proceedings of the National Academy of Sciences*, vol. 118, no. 19, p. e2022038118, 2021.

[66] B. A. Mecking, "The cebaf large acceptance spectrometer (clas)," 2003.

[67] H. Ji and Y. Li, *Monte Carlo Methods and Their Applications in Big Data Analysis*, pp. 125–139. 2016.

[68] M. Paganini, L. de Oliveira, and B. Nachman *Physical Review Letters*, vol. **120**, 4, (2018).

[69] A. Gretton, K. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola *Advances in Neural Information Processing Systems 19*, (2007).

[70] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[71] J. Gauthier, "Conditional generative adversarial nets for convolutional face generation," 2015.

[72] A. Clark, J. Donahue, and K. Simonyan vol. arXiv:1907.06571 [cs.CV], (2019).

[73] D. Michelsanti and Z.-H. Tan, "Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification," *Interspeech 2017*, Aug 2017.

[74] D. Donahue and A. Rumshisky, "Adversarial text generation without reinforcement learning," *CoRR*, vol. abs/1810.06640, 2018.

[75] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[76] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[77] M. E. Christy and P. E. Bosted, "Empirical fit to precision inclusive electron-proton cross sections in the resonance region," *Physical Review C*, vol. 81, may 2010.

[78] A. N. H. Blin, V. Mokeev, M. Albaladejo, C. Ferná ndez-Ramírez, V. Mathieu, A. Pilloni, A. Szczepaniak, V. D. Burkert, V. V. Chesnokov, A. A. Golubenko, and M. Vanderhaeghen, "Nucleon resonance contributions to unpolarized inclusive electron scattering," *Physical Review C*, vol. 100, sep 2019.

[79] A. N. H. Blin, W. Melnitchouk, V. I. Mokeev, V. D. Burkert, V. V. Chesnokov, A. Pilloni, and A. P. Szczepaniak, "Resonant contributions to inclusive nucleon structure functions from exclusive meson electroproduction data," *Physical Review C*, vol. 104, aug 2021.

[80] E. Golovatch *et al.*, "First results on nucleon resonance photocouplings from the $\gamma p \to \pi^+\pi^- p$ reaction," *Phys. Lett. B*, vol. 788, pp. 371–379, 2019.

[81] M. Battaglieri *et al.*, "Photoproduction of pi+ pi- meson pairs on the proton," *Phys. Rev. D*, vol. 80, p. 072005, 2009.

[82] M. Battaglieri *et al.*, "First measurement of direct f0(980) photoproduction on the proton," *Phys. Rev. Lett.*, vol. 102, p. 102001, 2009.

[83] J. Ballam *et al.*, "Bubble Chamber Study of Photoproduction by 2.8-GeV and 4.7-GeV Polarized Photons. 1. Cross-Section Determinations and Production of rho0 and Delta++ in the Reaction gamma p –> p pi+ pi-," *Phys. Rev. D*, vol. 5, p. 545, 1972.

[84] D. Bang and H. Shim, "Software package for *eic-smear*."

[85] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," 2018.

[86] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," 2016.

[87] A. Clark, J. Donahue, and K. Simonyan, "Adversarial video generation on complex datasets," 2019.

[88] S. Arora and Y. Zhang, "Do gans actually learn the distribution? an empirical study," 2017.

[89] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017.

[90] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, "Veegan: Reducing mode collapse in gans using implicit variational learning," 2017.

[91] D. Bang and H. Shim, "Mggan: Solving mode collapse using manifold guided training," 2018.

[92] R. A. Khalek, A. Accardi, J. Adam, D. Adamiak, W. Akers, M. Albaladejo, A. Albataineh, M. G. Alexeev, and et al, "Science requirements and detector concepts for the electron-ion collider: Eic yellow report," 2021.

[93] C. Cocuzza, C. E. Keppel, H. Liu, W. Melnitchouk, A. Metz, N. Sato, and A. W. Thomas, "Isovector emc effect from global qcd analysis with marathon data," *Phys. Rev. Lett.*, vol. 127, p. 242001, Dec 2021.

[94] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[95] H. Abramowicz, I. Abt, and L. Adamczyk, "Combination of measurements of inclusive deep inelastic e±p scattering cross sections and qcd analysis of hera data," 2015.

[96] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," 2016.

[97] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

[98] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2018.

[99] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Advances in Neural Information Processing Systems* (S. Becker, S. Thrun, and K. Obermayer, eds.), vol. 15, MIT Press, 2002.

[100] D. M. Chan, R. Rao, F. Huang, and J. F. Canny, "t-sne-cuda: Gpu-accelerated t-sne and its applications to modern data," 2018.

[101] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, p. 226–231, AAAI Press, 1996.

[102] I. J. Goodfellow, O. Vinyals, and A. M. Saxe, "Qualitatively characterizing neural network optimization problems," 2014.

[103] A. Accardi, L. Albacete, M. Anselmino, C. Aschenauer, and et al, "Electron-ion collider: The next qcd frontier," 2016.

[104] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, 2017.

[105] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," 2016.

[106] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015.

[107] K. T. Matchev and P. Shyamsundar, "Uncertainties associated with gan-generated datasets in high energy physics," *arXiv*, vol. 2002.06307v2, 2020.

[108] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, and T. Plehn, "Ganplifying event samples," *arXiv*, vol. 2008.06545, 2020.

VITA

Yasir Alanazi
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

EDUCATION

Bachelor of Science in Computer Science (2010)
Al-Jouf University, Sakaka, Saudia Arabia

Master of engineering in Software Engineering (2017)
University of St. Thomas, St. Paul, Minnesota, USA

PhD Candidate in Computer Science (2022)
Old Dominion University, Norfolk, Virginia, USA

PUBLICATIONS

Google Scholar: `https://scholar.google.com/citations?user=el7kpxkAAAAJ&hl`

CONTACT

Email: alanaziyasir@gmail.com

Typeset using LaTeX.