UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Application of generative models based on normalizing flows for the simulation of events in LHC experiments as an alternative to Monte Carlo methods.

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Boix Ávalos, Joan

Tutor: Gómez Adrian, Jon Ander

External cotutor: SALT CAIROLS, JOSE FRANCISCO

Experimental director: RUIZ DE AUSTRI BAZAN, ROBERTO

ACADEMIC YEAR: 2022/2023

Agradecimientos

Quisiera aprovechar este espacio para expresar mi más sincero agradecimiento a todas las personas que han sido parte de esta aventura que ha sido obtención de mi grado universitario.

En primer lugar, quiero agradecer a mi tutor y profesor de la UPV, Jon Ander Gómez Adrián, quien no solo me ha ayudado en este proyecto tan importante con su amplio conocimiento, sino que también me ha transmitido valores fundamentales como la resiliencia y la paciencia.

En segundo lugar, deseo expresar mi agradecimiento a mis tutores externos del IFIC, Roberto Ruiz de Austri Bazan y José Salt Cairols, quienes han sido de gran ayuda al proporcionarme los conocimientos necesarios en el campo de la física para llevar a cabo este estudio.

En tercer lugar, me gustaría expresar mi profundo agradecimiento a mis padres, quienes han apoyado en todo momento mis decisiones y nunca me han dejado solo frente a mis errores.

Por último, quiero dar las gracias a la Universitat Politècnica de València, por brindarme la oportunidad de conocer a personas maravillosas que hoy en día puedo llamar amigos, y por facilitarme los conocimientos necesarios para poder seguir con mi carrera profesional.

Gracias.

Resum

Per a superar les limitacions dels models tradicionals de simulació de dades, que són computacionalment costosos i requereixen molt de còmput, proposem utilitzar models generatius del tipus normalitzadors de flux basats en aprenentatge profund, com MADE, MAF i IAF. Aquests models poden simular col·lisions de partícules en física d'altes energies de manera més eficient que els mètodes basats en Monte Carlo, per a detectar senyals de la física més enllà del model estàndard, la qual cosa és essencial en l'exploració dels aspectes desconeguts del comportament de la matèria.

MADE (Autoencoder Emmascarat per a Estimació de Distribucions), MAF (Flux Autoregressiu Emmascarat) i IAF (Flux Autoregressiu Invers) són exemples de models basats en Normalitzadors de Flux que han demostrat ser altament efectius en la simulació de distribucions de dades complexes. MADE utilitza una xarxa d'autoencoder amb connexions emmascarades per a aproximar qualsevol distribució de probabilitat, la qual cosa el fa adequat per a generar dades d'alta dimensionalitat amb una estructura dispersa. MAF és un model de Flux que modela una distribució objectiu transformant una distribució base simple a través d'una sèrie de transformacions invertibles, la qual cosa permet un mostreig i estimació de densitat eficient. IAF és un altre model del tipus Normalitzadors de Flux que transforma seqüencialment una distribució base amb funcions invertibles per a generar mostres d'una distribució objectiu.

Utilitzant aquests models generatius basats en Normalitzadors de Flux, podem simular dades amb alta qualitat i reduir el temps i els costos energètics dels mètodes tradicionals, proporcionant una forma més eficient i precisa de descobrir nous esdeveniments i dissenyar detectors en el context del LHC. A més, aquests models faciliten l'obtenció d'una estimació completa i detallada dels errors sistemàtics, la qual cosa és crucial per a validar amb precisió nous escenaris de física comparant les dades reals obtingudes dels experiments amb dades simulades de models teòrics.

Paraules clau: Aprenentatge profund, esdeveniments de Monte Carlo, models generatius basats en aprenentatge profund, models de flux, nova física, Gran Col·lisionador d'Hadrons

Resumen

Para superar las limitaciones de los modelos tradicionales de simulación de datos, que son computacionalmente costosos y requieren mucho cómputo, proponemos utilizar modelos generativos del tipo normalizadores de flujo basados en aprendizaje profundo, como MADE, MAF e IAF. Estos modelos pueden simular colisiones de partículas en física de altas energías, de manera más eficiente que los métodos basados en Monte Carlo, para detectar señales de la física más allá del modelo estándar, lo cual es esencial en la exploración de los aspectos desconocidos del comportamiento de la materia.

MADE (Autoencoder Enmascarado para Estimación de Distribuciones), MAF (Flujo Autoregresivo Enmascarado) e IAF (Flujo Autoregresivo Inverso) son ejemplos de modelos basados en Normalizadores de Flujos que han demostrado ser altamente efectivos en la simulación de distribuciones de datos complejas. MADE utiliza una red de autoencoder con conexiones enmascaradas para aproximar cualquier distribución de probabilidad, lo que lo hace adecuado para generar datos de alta dimensionalidad con una estructura dispersa. MAF es un modelo de Flujo que modela una distribución objetivo transformando una distribución base simple a través de una serie de transformaciones invertibles, lo que permite un muestreo y estimación de densidad eficiente. IAF es otro modelo del tipo Normalizadores de Flujo que transforma secuencialmente una distribución base con funciones invertibles para generar muestras de una distribución objetivo.

Utilizando estos modelos generativos basados en Normalizadores de Flujos, podemos simular datos con alta calidad y reducir el tiempo y los costos energéticos de los métodos tradicionales, proporcionando una forma más eficiente y precisa de descubrir nuevos eventos y diseñar detectores en el contexto del LHC. Además, estos modelos facilitan la obtención de una estimación completa y detallada de los errores sistemáticos, lo cual es crucial para validar con precisión nuevos escenarios de física comparando los datos reales obtenidos de los experimentos con datos simulados de modelos teóricos.

Palabras clave: Deep Learning, Eventos de Monte Carlo, Modelos generativos basados en Deep Learning, Modelos Flow, Nueva Física, Gran Colisionador de Hadrones

Abstract

To overcome the limitations of traditional data simulation models that are computationally expensive and time-consuming, we propose using generative models of the flow normalizers type based on deep learning, such as MADE, MAF, and IAF. These models can simulate particle collisions in high-energy physics more efficiently than Monte Carlo-based methods to detect signals of physics beyond the standard model, which is essential in exploring unknown aspects of the behavior of matter.

MADE (Masked Autoencoder for Distribution Estimation), MAF (Masked Autoregressive Flow), and IAF (Inverse Autoregressive Flow) are examples of flow normalizer-based models that have been highly effective in simulating complex data distributions. MADE uses a masked autoencoder network to approximate any probability distribution, making it suitable for generating high-dimensional data with a sparse structure. MAF is a flow model that models a target distribution by transforming a simple base distribution through a series of invertible transformations, allowing for efficient sampling and density estimation. IAF is another flow normalizer-based model that sequentially transforms a base distribution with invertible functions to generate samples from a target distribution.

By using these generative models based on flow normalizers, we can simulate high-quality data and reduce the time and energy costs of traditional methods, providing a more efficient and accurate way to discover new events and design detectors in the context of the LHC. Additionally, these models facilitate obtaining a complete and detailed estimate of systematic errors, which is crucial for accurately validating new physics scenarios by comparing real data obtained from experiments with simulated data from theoretical models.

Key words: Deep Learning, Monte Carlo events, Generative models based on Deep Learning, Flow models, New Physics, Large Hadron Collider

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

At present, the Standard Model serves as a theoretical framework that explains the composition of matter and the forces that regulate the physical processes of its constituents. The most recent addition to this model, the Higgs boson[1], is included. Nevertheless, certain observables do not align with theoretical predictions, and there are several fundamental aspects of physics that the model fails to explain, such as the neutrino masses, gravity, and dark matter. The Standard Model also features unexplained parameters, such as the Higgs mass, which is relatively low despite its considerable quantum corrections, indicating a lack of understanding.

To shed light on the fundamental laws of nature, the European Organization for Nuclear Research (CERN) operates the Large Hadron Collider (LHC), the most powerful and extensive collider worldwide. The LHC accelerates particles to high velocities and collides them, studying their interactions.

At the LHC, a program obtains actual data by recording a small fraction of proton-proton collisions. A trigger selects events online, and the recorded events are processed and reconstructed. Using seven detectors, four collision points collect data from the 27-kilometer circular ring of the LHC[2]. Simulated data is also generated during an experiment's lifespan, both quickly and in detail, to compare models with actual events. Results from analyses of the collected data are discussed.

In addition to measuring Standard Model observables, experiments also search for new Physics Beyond the Standard Model that could explain the current model's limitations. The LHCb, ATLAS, ALICE, and CMS experiments have already collected a substantial amount of data. ATLAS and CMS are general-purpose experiments, aiming to research the origin of masses and Physics Beyond the Standard Model. While there is no confirmed new model of physics, promising results could emerge with further experimentation. Conversely, LHCb concentrates on B physics, matter-antimatter asymmetry, and CP violation, and ALICE is an experiment that investigates Heavy Ion physics.

This work focuses on the ATLAS experiment (Figure 1.1). By applying Deep Generative Models[1], in our case, Normalizing Flows[2], we provide a faster way to simulate physical events.

[1]Class of unsupervised learning algorithms that utilize deep neural networks to learn and generate new data that resembles the original dataset.

[2]Type of probabilistic models that transform a simple distribution into a complex target distribution through a series of invertible transformations.

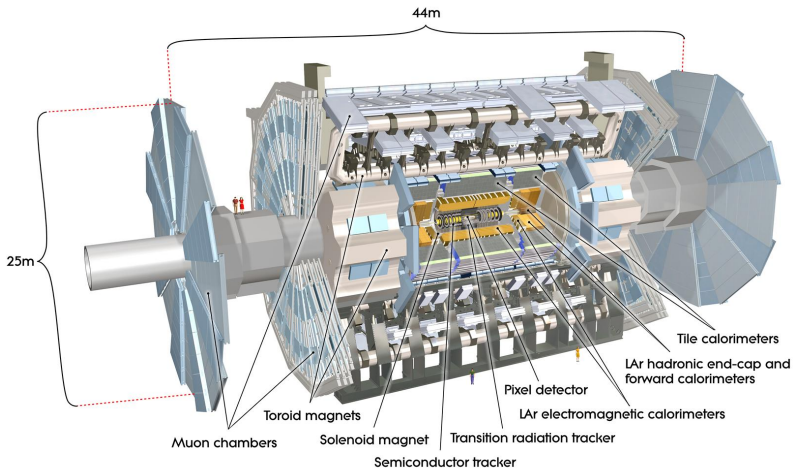


Figure 1.1: Computer-generated image of the ATLAS detector located at CERN.

1.1 Motivation and problem description

The current methods used for simulating physical processes using the Markov Chain Monte Carlo methods (MCMC)[3] face a significant challenge due to their high computational requirements. This approach involves three steps: sampling pseudo-random numbers and transforming them into simulated physical events, such as particle collisions, to then select the desired ones and discard the rest.

As pointed out, the extensive computational resources needed by these simulations put a hard limit on scientific progress in the field. Algorithms like VEGAS[4] can take up to 10 minutes[5] to generate particle physics experiments of an MC event, including the detector response.

We believe this event generation pipeline can be accelerated using Deep Generative Models, which may have the ability to produce more comprehensive and precise results for signals in the Standard Model and Beyond the Standard Model of Physics.

It is essential to find an alternative for event generation, as the time and energy consumption required by current methods will prove infeasible and unsustainable in the future, when billions of events will need to be produced due to the buildup of data obtained in the collider.

1.2 Objectives

In order to establish the scope of this study, we define the objectives we aim to accomplish as the result of this work.

As pointed out previously, one goal of this study is to provide new frameworks for efficient event generators as an alternative to the current methods based on Monte Carlo.

The results of the experiments will be compared based on standard metrics for their use case, which provide insight on the similarities and dissimilarities of our estimated distributions and the *ground truth*[3] distributions.

[3] Accurate reference data for evaluating or comparing other information or models.

Finally, we analyze the strengths and weaknesses of each approach, as we aim to provide different ways to generate and construct events. We will focus on obtaining distributions that adjust to the data provided generated with Monte Carlo methods

1.3 Structure of this work

The work is divided into seven parts, each addressing specific aspects and contributing to the overall organization and coherence of the study:

First, we introduce the problem and its related ongoing CERN experiments, briefly describing the current shortcomings followed by the objectives of this study.

The second chapter is dedicated to the analysis of the state-of-the-art. Here we describe the present cutting-edge advancements, providing a brief overview of classical and similar approaches along with machine learning solutions that closely align with out intended objectives.

Chapter three of our study centers on the dataset utilized. We provide a brief overview of its generation process, outlining the constraints involved, and discussing the file formatting of the provided data.

In chapter four, we propose our own solution to improve upon the models mentioned in the previous references. Our objective is to develop a valid model that effectively meets the specified goals. Through careful analysis and consideration of the existing literature, we crafted an approach that addresses the limitations of prior work.

Chapter five is dedicated to the technologies employed to make this study possible, from the software tools to the hardware architecture and infrastructure. Followed by chapter six, where we present the metrics by which our experiments' success were measured. We give a brief explanation on what they indicate and how we calculate them.

In the seventh chapter we present and describe the results of our experimentation, together with the framework for each experiment and a brief comparison between our results.

Finally in chapter eight we conclude by offering an assessment of the chosen approach's suitability for the intended experiment, and propose future improvements on this model and other techniques that may be worth exploring.

CHAPTER 2
State of the art

The LHC has always prioritized the detection and generation of signals related to new physics, which is evident from the numerous experiments conducted using both supervised and unsupervised machine learning techniques. However, despite these efforts, no evidence of physics beyond the Standard Model has been discovered so far.

As indicated previously, efforts from CERN, in particular from the ATLAS project team, tried to define a strategy for detection of new physics [6]. The cited study presents their strategy for a comprehensive search to identify potential new physics signals. Events were classified into multiple classes based on their final state, and an automated search algorithm tested data compatibility with Monte Carlo simulations in distributions sensitive to new physics effects. Significance of deviations was quantified using pseudo-experiments. Analysis of data-derived signal regions allows for statistical interpretation, with sensitivity evaluated using Standard Model processes and benchmark signals of new physics. No significant deviations were found, leading to the absence of data-derived signal regions for further analysis.

There has also been a community effort to find BSM signals, namely *The Dark Machines Anomaly Score Challenge*[7]. The challenge focused on detecting signals of new physics at the LHC using unsupervised machine learning algorithms. An anomaly score was proposed to define model-independent signal regions in LHC searches. A benchmark dataset consisting of over 1 billion simulated LHC events was provided, which we employ in this study and describe in Chapter 3. Various anomaly detection and density estimation algorithms developed for the challenge were reviewed, and their performance was assessed in realistic analysis environments. The valuable insights gained contribute to the advancement of unsupervised methods in the search for new physics at the LHC.

Similar to the objective of our study, we find [8]. The aforementioned study focuses on Drell-Yan processes at the LHC, and it describes a novel approach which employs normalizing flows as an integrator to enhance the unweighting efficiency of Monte-Carlo event generators in collider physics simulations.

To conclude, it is worth pointing out some of the different approaches to the subject of this study, **event generation**. Efforts have been made using Monte Carlo methods coupled with regression [9, 10]. Other Deep Learning Generative Methods such as GANs [11, 12, 13] and VAEs [14] have also been used in this context. As of Normalizing Flows, we observe that they have been mainly used as a piece of the VAE generation process[15], and rarely as a standalone model.

2.1 Analysis of the current situation

After carefully reviewing the latest contributions in the field of efficient event simulation using Deep Learning Generative Models, we have discovered several key findings that warrant further exploration.

One important observation is that most research in this area has predominantly focused on supervised learning methods. This has sparked our curiosity to dive into the less explored realm of unsupervised learning, which presents exciting opportunities for advancement in event simulation.

While Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have received considerable attention in event simulation studies (as evident from relevant research), the utilization of Normalizing Flows, a type of generative model, has been relatively limited. Typically, Normalizing Flows have been used as auxiliary tools, assisting in the sampling process from the latent space of VAEs and GANs. However, their independent potential and effectiveness in event generation remain largely unexplored.

Therefore, the primary aim of our study is to thoroughly investigate and analyze the behavior of various Normalizing Flow models within the context of event generation, encompassing both Standard Model (SM) and Beyond the Standard Model (BSM) processes. By diving into this uncharted territory and carefully examining the performance and capabilities of Normalizing Flows in event simulation, we hope to uncover new insights and possibilities that could drive progress in this field of research.

CHAPTER 3
The dataset

The datased used for the experimentation is one of many generated to be used in **The Dark Machines Anomaly Score Challenge**[16], whose objective was to give an event-by-event classification between SM events and events produced by BSM processes via an anomaly score.

Each event represents a simulated proton-proton collision similar to those ocurring at the LHC at a center-of-mass energy of 13 TeV[1], and is comprised of a series of final-state objects (Table 3.1).

Symbol ID	Object
j	jet
b	b-jet
e^-	electron (e^-)
e^+	positron (e^+)
m^-	muon (μ^-)
m^+	antimuon (μ^+)
g	photon (γ)

Table 3.1: Symbol ID and corresponding object type in each event.

3.1 Data generation

The creation of the dataset for the SM and BSM processes was performed applying diverse technologies which served to generate events, handle parton-level matrix elements and parton distribution functions, incorporate parton showering, simulate the detector response, and perform jet clustering. A detailed explanation of the data generation procedure is described in [17].

The number of objects resulting from a collision event can vary, and for each of them the full energy (E), traverse momentum (p_T), pseudo-rapidity (η) and azimuthal angle (ϕ) are recorded.

An event was stored in the dataset and is stored in this experiment if it meets at least one of the following requirements:

- At least one jet or b-jet with $p_T > 60$ GeV and $|\eta| < 2.8$

[1]Teraelectronvolt. Ultra-high energy unit in particle physics, representing trillions of electron volts. Each electronvolt represents the energy acquired by an electron in passing through a potential difference of 1 volt.

- At least one electron with $p_T > 25$ GeV and $|\eta| < 2.47$, except for $1.37 < |\eta| < 1.52$
- At least one muon with $p_T > 25$ GeV and $|\eta| < 2.7$
- At least one photon with $p_T > 25$ GeV and $|\eta| < 2.37$

Some additional requirements regarding final objects were applied on the dataset and are also applied in this experiment. The requirements on the final state objects are:

- Jet or b-jet: $p_T > 20$ GeV and $|\eta| < 2.8$
- Electron or muon: $p_T > 15$ GeV and $|\eta| < 2.7$
- Photon: $p_T > 20$ GeV and $|\eta| < 2.37$

3.2 Data format

Each file corresponds to a process. The total number of generated events per file is at least the number that is needed for obtaining 10 fb^{-1} of data[2] ($N_{10fb^{-1}}$). When $N_{10fb^{-1}} <$ 20000, a second file with 20000 lines is given.

The data is formatted as a CSV file with one line per event, and each line can vary in length. An event consists of three event-specifiers followed by the kinematic features corresponding to each object within the event. The line format is:

```
event ID; process ID; event weight; MET; METphi; obj1, E1, pt1, eta1, phi1;
                    obj2, E2, pt2, eta2, phi2; ...
```

Each event has three specifiers. The `event ID` is a unique integer that serves to identify a particular event for debugging purposes. The `process ID` is a string referring to the process that generated the event, as seen in Table 3.2. The `event weight` depends on the cross section for a particular process and the number of events in a single file, and is ignored in our experiments.

Every event also has two kinematic features, the `MET` and `METphi` entries are the magnitude E_T^{miss} and the azimuthal angle $\phi_{E_T^{miss}}$ of the missing traverse energy vector of the event, which represent the traverse energy and azimuthal angle of the objects that escape detection.

The object identifiers (`obj1`,`obj2`,...) are strings that follow Table 3.1, which identify each object in the event. Four values follow each object identifier, which specify its features: E_n (E), pt_n (p_T), eta_n (η), phi_n (ϕ). These features correspond to those mentioned previously in 3.1.

The particles are shown in a determined order: b-jets, jets, leptons and photons. And inside each particle type, they are sorted in descending order based on their p_T. The particle with the highest p_T inside its class will be referred as the *leading* particle.

As previosuly indicated, all of the events are separated by process, generating 65 files with a total size of 66.5GB. For experimentation, only the largest files for each type of process were chosen (for processes that reached $10fb^{-1}$ with less than 20K events, the file with 20K events was chosen).

[2]The inverse femtobarn indicates the number of particle collisions that have occurred, on average, per femtobarn of integrated luminosity ($1fb^{-1} \approx 10^{12}$ proton-proton collisions).

SM processes	
Physics process	*Process ID*
pp → jj	njets
pp → W ± (+2j)	w_jets
pp → γ (+2j)	gam_jets
pp → Z (+2j)	z_jets
pp → tt + jets (+2j)	ttbar
pp → W ± t (+2j)	wtop
pp → W ± $\bar{t}$ (+2j)	wtopbar
pp → W W (+2j)	ww
pp → t + jets (+2j)	single_top
pp → $\bar{t}$ + jets (+2j)	single_topbar
pp → $\gamma\gamma$ (+2j)	2gam
pp → W γ (+2j)	Wgam
pp → ZW ± (+2j)	zw
pp → Zγ (+2j)	Zgam
pp → ZZ (+2j)	zz
pp → h (+2j)	single_higgs
pp → ttγ (+1j)	ttbarGam
pp → tt$\bar{Z}$	ttbarZ
pp → tt$\bar{h}$ (+1j)	ttbarHiggs
pp → γt (+2j)	atop
pp → tt$\bar{W}$	ttbarW
pp → $\gamma\bar{t}$ (+2j)	atopbar
pp → Zt (+2j)	ztop
pp → Z$\bar{t}$ (+2j)	ztopbar
pp → tt$\bar{t}t$	4top
pp → tt$\bar{W}$ + $\bar{W}$	ttbarWW

BSM processes	
Physics process	*Process ID*
pp → $\tilde{g}\tilde{g}$ (1 TeV)	gluino_01
pp → $\tilde{g}\tilde{g}$ (1.2 TeV)	gluino_02
pp → $\tilde{g}\tilde{g}$ (1.4 TeV)	gluino_03
pp → $\tilde{g}\tilde{g}$ (1.6 TeV)	gluino_04
pp → $\tilde{g}\tilde{g}$ (1.8 TeV)	gluino_05
pp → $\tilde{g}\tilde{g}$ (2 TeV)	gluino_06
pp → $\tilde{g}\tilde{g}$ (2.2 TeV)	gluino_07
pp → $\tilde{t}_1\tilde{t}_1$ (220 GeV), $m_{\tilde{x}_0} = 20$ GeV	stop_01
pp → $\tilde{t}_1\tilde{t}_1$ (300 GeV), $m_{\tilde{x}_0} = 100$ GeV	stop_02
pp → $\tilde{t}_1\tilde{t}_1$ (400 GeV), $m_{\tilde{x}_0} = 100$ GeV	stop_03
pp → $\tilde{t}_1\tilde{t}_1$ (800 GeV), $m_{\tilde{x}_0} = 100$ GeV	stop_04
pp → Z^0 (2 TeV)	Zp_01
pp → Z^0 (2.5 TeV)	Zp_02
pp → Z^0 (3 TeV)	Zp_03
pp → Z^0 (3.5 TeV)	Zp_04
pp → Z^0 (4 TeV)	Zp_05

Table 3.2: Generated SM and BSM signal processes with their identification.

CHAPTER 4
Proposed solution

Considering the current state of the art discussed in Chapter 2 and the dataset analysis in Chapter 3, a solution is proposed utilizing Normalizing Flows to generate relevant results in event generation. The choice of using Normalizing Flows is based on the relatively few studies employing this model compared to the more popular VAEs and GANs. VAEs, along with GANs and Flow models, are widely used in the AI field, and we expect promising results with this technique.

Once a well-fitting model capturing the data distribution of the dataset is obtained, it will facilitate extensive searches for signals of Physics Beyond the Standard Model, enabling the validation of new theoretical models.

A concise explanation of the theoretical concepts underpinning Normalizing Flows and some of its variants' architectures follows, coupled with custom data structures that facilitate the flexibility of the events generated.

4.1 Alias Method

The Alias method[18, 19] is a technique used for efficient random sampling from discrete probability distributions. It is based on the idea of using a mixture of uniform random variables and a lookup table to generate random values.

To construct the Alias table, we start with a probability distribution where each outcome is associated with a certain probability. We calculate the average probability and divide the outcomes into two groups: *alias* and *prob*. The *alias* group contains the outcomes with probabilities greater than or equal to the average, while the *prob* group contains the outcomes with probabilities less than the average.

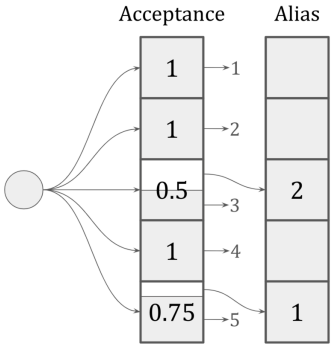


Figure 4.1: A diagram of an alias table that represents the probability distribution ⟨0.25, 0.3, 0.1, 0.2, 0.15⟩. [20]

For each outcome in the *prob* group, we pair it with an outcome from the *alias* group. The paired outcome is selected randomly, and its probability is adjusted to be the difference between the average probability and the probability of the *prob* outcome.

To generate a random value from the distribution, we first select a row from the Alias table uniformly at random. Then, we toss a fair coin to decide whether to choose the original outcome or its alias. This process allows us to sample from the desired distribution efficiently.

The Alias method provides a constant-time complexity $O(1)$ for generating random values from discrete probability distributions, regardless of the number of outcomes. We will employ a variant of this method to represent the distribution of appearances of particles in the observed events.

4.1.1. Counter

In order to guarantee the appearance of even the most rare event configurations, we decided to encode the particle presence on an event as a 7-dimensional integer vector. In this vector, as can be seen on Table 4.1, each dimension corresponds to the amount of objects of a certain particle type observed on an event.

Particle Type	j	b	e^+	e^-	m^+	m^-	g
Counter Vector	j_n	b_n	e_n^+	e_n^-	m_n^+	m_n^-	g_n

Table 4.1: Counter Vector reference to Particle types present in an event.

Applying the technique described above, we constructed a *Counter Generator* (see Figure 4.2), which models the discrete distribution of the number of particles per event using two levels. Level one contains up to 80% of occurrences, and level two the 20% remaining.

More concretely, it is capable of sampling from the distribution of 7-dimensional vectors that describe particle amounts in a collision event; one dimension per particle.

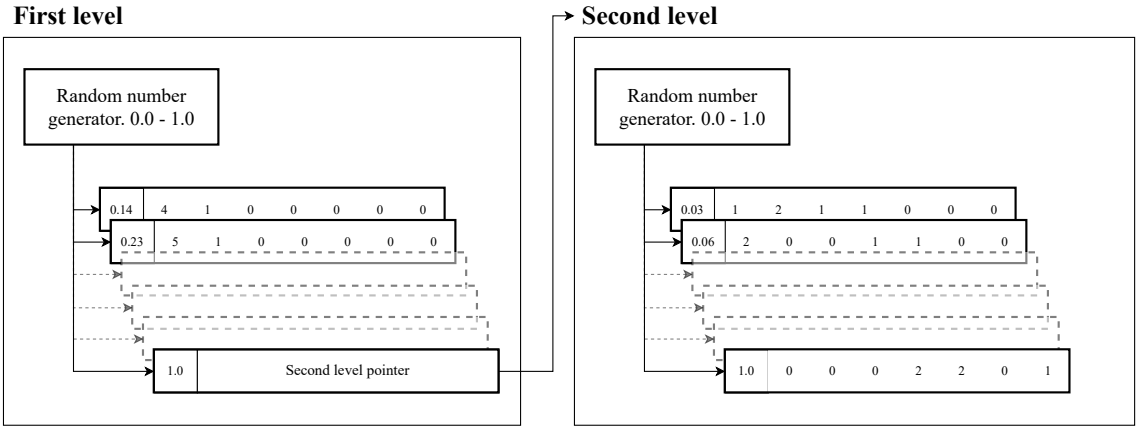


Figure 4.2: Two-level counter generator.

As we can see in Figure 4.3, this technique allows us to correctly sample from the distribution of particle amounts per event.

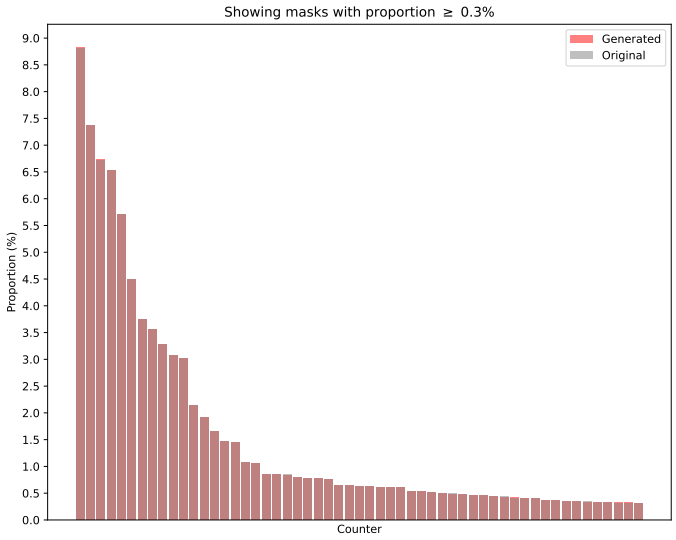


Figure 4.3: Count array distribution based on % over 5 million samples. It can be seen both have the same distribution, with no relevant differences.

4.2 Density Estimation

Density estimation is a fundamental task in statistical modeling and machine learning that aims to learn the underlying probability distribution of a given dataset. The goal is to estimate the density function, which describes the likelihood of observing a particular value of the random variable that the data points represent. Density estimation has applications in various domains, such as anomaly detection, signal processing, image and speech recognition, and natural language processing.

There are several methods for density estimation, such as Masked Autoencoder for Distribution Estimation (MADE)[21], which is a parametric modeling approach to this problem and the building block of Masked Autoregressive Flows (MAF)[22].

4.2.1. Normalizing Flows

Normalizing flows[23] is a class of generative models that learn to transform a simple probability distribution into a complex target distribution. The idea is to apply a sequence of invertible and differentiable transformations (Figure 4.4), called flow layers, to a base distribution to obtain the target distribution. The transformations can be parameterized by neural networks, allowing for flexible and expressive mappings.

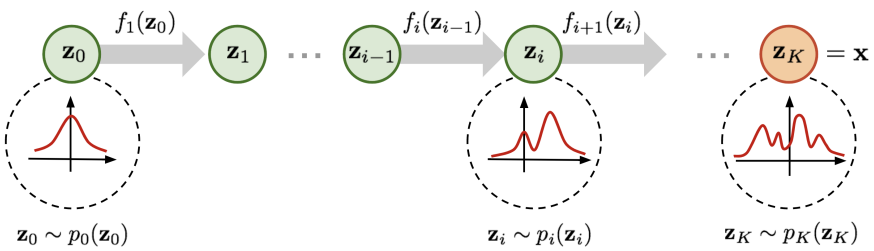


Figure 4.4: Illustration of a normalizing flow model, transforming a simple distribution to a complex one step by step.

Normalizing flows offer several advantages over other generative models, such as the ability to model complex and multimodal distributions, exact likelihood computation, efficient sampling and inference, and controllable data generation.

4.2.2. Autoregressive Flows

The **autoregressive constraint** is a method for modeling sequential data where each output is influenced only by previously observed data. The likelihood of observing a specific element is determined by its conditional probability given the preceding elements, and the overall probability of observing the entire sequence is obtained by multiplying these conditional probabilities together.

$$p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i|x_1, \ldots, x_{i-1}) = \prod_{i=1}^{D} p(x_i|x_{1:i-1})$$

In order to obtain a normalizing flow with the autoregressive property, we have to frame its flow transformations as autoregressive models, where each dimension in a vector variable is conditioned only on the previous dimensions. This is called an **autoregressive flow**.

Typically, the parameters of the conditional densities can be learned. The autoregressive density $p(x_{1:D})$, for instance, is a popular option.

$$p(x_i|x_{1:i-1}) = N(x_i|\mu_i, (\exp(\alpha_i))^2)$$

$$\begin{aligned} \mu_i &= f_{\mu_i}(x_{1:i-1}) \\ \alpha_i &= f_{\alpha_i}(x_{1:i-1}) \end{aligned}$$

Its conditional density is a univariate Gaussian, and neural networks that rely on the prior $x_{1:i-1}$ compute its mean and standard deviation.

In order to sample from said distribution, we generate vectors u from a standard Normal $N(0, 1)$, and apply a simple scale-and-shift to obtain x:

$$x_i = u_i \cdot exp(\alpha_i) + \mu_i$$

$$u_i \approx N(0, 1)$$

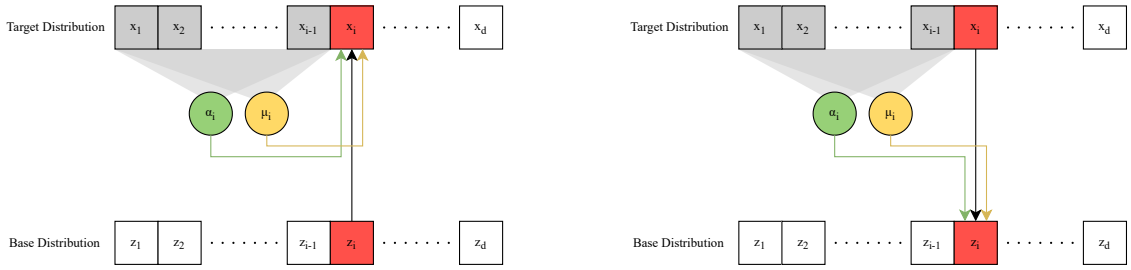
4.2.3. MAF

The Masked Autoregressive Flow (MAF)[22] is a Normalizing Flow model which implements the conditional-Gaussian autorregressive property described before.

MAF models can learn quickly thanks to the simultaneous evaluation of all the conditional likelihoods $p(x_1), p(x_2|x_1), ...p(x_D|x_{1:D-1})$ can be evaluated simultaneously by taking advantage of the batch parallelism of machine-learning-ready GPUs.

This model works as a **bijector**, and as such, it can perform forward and backward passes on data.

In order to obtain x_i, MAF uses α_i and μ_i, which are scalars computed by passing $x_{1:i-1}$ through a network, as depicted by Figure 4.5a. The transformation applied is a simple scale-and-shift, as described in the section before.



(a) MAF forward pass. **(b)** MAF inverse pass.

Figure 4.5: MAF scale-and-shift operations.

This transformation was made to be easily invertible, to computing $u = T^{-1}(x)$ does not require to invert f_α and f_μ. Therefore, we can easily recover the original sample u from the base distribution by reversing the scale-and-shift:

$$u_i = (x_i - \mu_i) \cdot exp(-\alpha_i)$$

Sampling can be slow, since in order to compute a dimension x_i, the model has to wait for all previous $x_{1:i-1}$ to be computed, therefore being unable to exploit the previously mentioned processor parallelism.

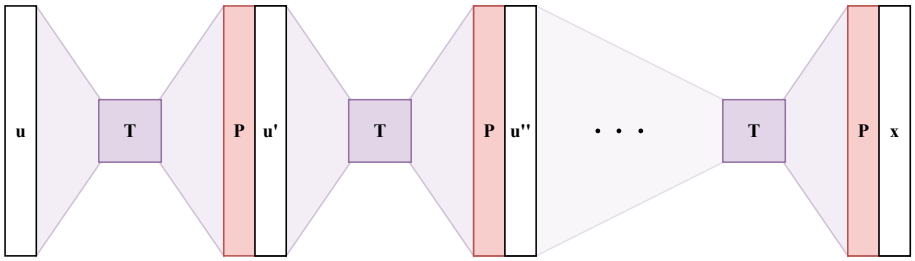


Figure 4.6: MAF Architecture with multiple layers.

This flow model is usually implemented as depicted on Figure 4.6. We start from a distribution U, which we use to obtain the base samples u, and we chain the scale-and-shift transformations, introducing a permutation layer P to change the ordering of each dimension so the model can learn the inter-dependencies between all the variables.

MAF uses the Masked Autoencoder for Distribution Estimation (MADE)[21] in order to compute the nonlinear parameters of the shift-and-scale transformations described earlier simultaneously and in one pass, incorporating these efficient autoregressive models within the normalizing flows framework.

MADE

Masked Autoencoder for Distribution Estimation (MADE)[21] represents a specifically crafted structure that effectively enforces the **autoregressive characteristic** in an autoencoder

Instead of supplying the autoencoder with diverse observation window inputs, MADE employs **binary mask matrices** to eliminate the influence of specific hidden units. This process ensures that each input dimension is reconstructed solely from preceding dimensions, following a predetermined order, within a single iteration when predicting the conditional probabilities using the autoencoder.

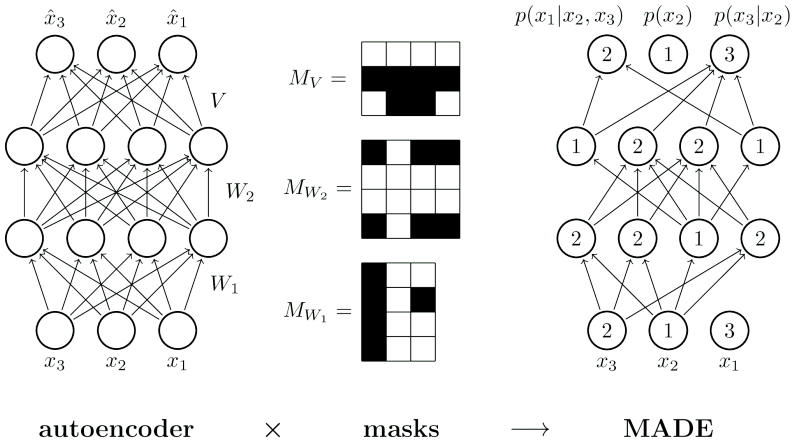


Figure 4.7: Masking of a conventional three hidden layer autoencoder into a MADE.

The numbers are assigned to all the units and layers, fixing the ordering of the input dimensions, and calculating the conditional probability with respect to it.

As seen in Figure 4.7, a unit in a layer can only be connected to other units with equal or smaller numbers in the previous layer, and this type of dependency propagates through the network up to the output layer.

4.2.4. RQS

The Rational Quadratic Spline (RQS) is a type of normalizing flow that represents a monotonically increasing piecewise rational quadratic function.

The RQS bijector is constructed using bins, where each bin defines a monotonically increasing rational-quadratic function within a specified interval. Outside of this interval, the function is set to the identity transformation.

A-RQS

The Autoregressive Rational Quadratic Spline (A-RQS) corresponds to the autoregressive variant of the Rational Quadratic Spline. In the context of flow models, this bijector provides enhanced flexibility than the MAF presented before, being able to perform better on datasets of higher dimensionality [24].

The transformation that the A-RQS bijector performs can be represented as follows:

$$x_i = \frac{(\alpha_i + \beta_i(u_i - u_k))}{(1 + \gamma_i(u_i - u_k))}$$

where u_k is the position of the k-th knot, α_i, β_i, and γ_i are the parameters of the rational quadratic function in the i-th interval, and x_i is the output value of the transformation at input u_i.

Therefore, this model follows the same architecture as the one presented in Figure 4.6, but instead of the scale-and-shift operation as the transform function, we have a much more flexible T [25].

CHAPTER 5
Used technologies

Throughout the history of AI research, numerous libraries and frameworks have emerged and expanded, offering new layers of abstraction to facilitate the implementation process.

In the development of this project, a variety of software and hardware tools of this nature have been utilized to design, train and test the proposed models.

5.1 Software environment

All the experiments were carried out in a Linux-based machine, running Ubuntu 20.04.6 LTS. Due to its standardized status in the field of artificial intelligence, Python 3[26] was chosen for the development of the codebase.

5.1.1. Model definition

For this study, Tensorflow, Tensorflow Probability and Keras, were used as the libraries for model definition and training.

Tensorflow[27] is an open-source deep learning framework. It utilizes a computational graph paradigm, where mathematical operations are represented as nodes connected by edges that flow data between them. Tensors, multidimensional arrays, are the fundamental data units in TensorFlow. The framework offers flexibility with both static and dynamic graph modes, enabling efficient memory allocation and accommodating varying input sizes or control flow. TensorFlow provides high-level APIs like Keras for easy model development, as well as lower-level APIs for advanced customization. Integration with GPUs and TPUs[1] enhances computation speed, making TensorFlow a powerful tool in the field of machine learning.

Tensorflow Probability[27] is a powerful library built on top of TensorFlow that integrates probabilistic modeling and machine learning. It enables researchers and practitioners to model uncertainty and make probabilistic predictions in their machine learning pipelines. TensorFlow Probability offers a wide range of probabilistic distributions, including Gaussian processes, Bayesian neural networks, and hidden Markov models, allowing for flexible modeling of complex and uncertain real-world phenomena.

Keras[28] is a high-level neural networks API built on TensorFlow which offers a user-friendly and modular approach for designing, training, and evaluating deep learning models. With its intuitive design, pre-defined layers, and built-in optimization al-

[1] Tensor Processing Units. Specialized, highly efficient and optimized hardware accelerator for performing tensor operations.

gorithms, Keras simplifies the process of model development and training. It promotes code reusability and transferability, allowing models to be easily shared and reused.

As in our experiments GPUs were employed for training and sampling generation, the libraries described above leveraged cuDNN and CUDA underneath.

cuDNN[29] or CUDA Deep Neural Network library, is a high-performance GPU-accelerated library developed by NVIDIA. It provides optimized implementations of key deep learning primitives, such as convolutional and recurrent operations, for training and inference on NVIDIA GPUs.

CUDA[30] is a parallel computing platform and programming model developed by NVIDIA that enables developers to harness the computational power of GPUs for general-purpose computing.

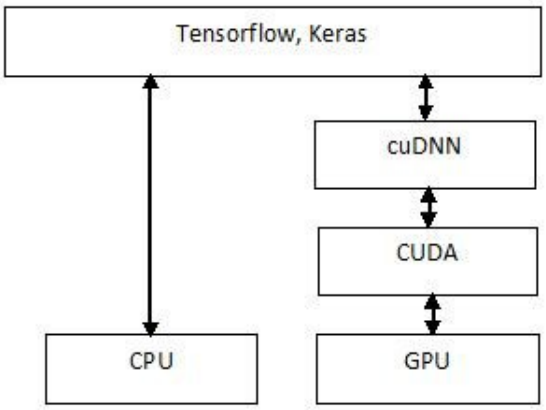


Figure 5.1: Interaction of Tensorflow and Keras with GPU, with the help of CUDA and cuDNN.[31]

5.1.2. Data representation

In order to observe the quality of the obtained results, a visual representation of the generated data must be used. In our case, we used Matplotlib to display our generated results' distribution and compare them to the expected distribution.

Matplotlib[32] is a data visualization library for creating static, animated, and interactive visualizations in Python. It provides a range of 2D and 3D plotting functionalities for generating publication-quality figures across a wide range of formats, including line plots, scatter plots, bar plots, histograms, pie charts, and many more. Matplotlib is widely used in the scientific and engineering communities for data analysis and communication of research findings, and it offers a high degree of customization and flexibility to suit various needs and preferences. It is an open-source library and is compatible with a wide range of platforms, including Windows, macOS, and Linux.

5.1.3. Model evaluation

Certain metrics had to be calculated to ensure the quality of our trained event generator models. For this, Scikit-learn was employed as it provides all the methods needed to obtain these metrics.

Scikit-learn[33] is a versatile machine learning library in Python, providing a user-friendly interface for data preprocessing, model selection, and evaluation. With its extensive collection of algorithms, evaluation metrics, and preprocessing techniques, scikit-

learn simplifies the development and deployment of machine learning models. In this work, it was used to compute validation metrics for the trained models.

5.2 Hardware environment

In order to tackle the complexity and scale of the machine learning problems at hand, it was essential to leverage a powerful hardware architecture for training our models.

The intricacy of deep learning algorithms, coupled with the vast amounts of data involved, demanded significant computational resources. By harnessing the capabilities of GPUs, which for this task achieve a higher performance when compared to CPUs, we were able to expedite the training process and achieve faster convergence.

Therefore, the hardware infrastructure used in this project consisted of a server with 32 GB of DDR4 RAM memory and a high-performance *NVIDIA GeForce RTX 3060 graphics card*, powered by the GA106-300-A1 chip (Figure 5.2) with 3584 NVIDIA CUDA cores.

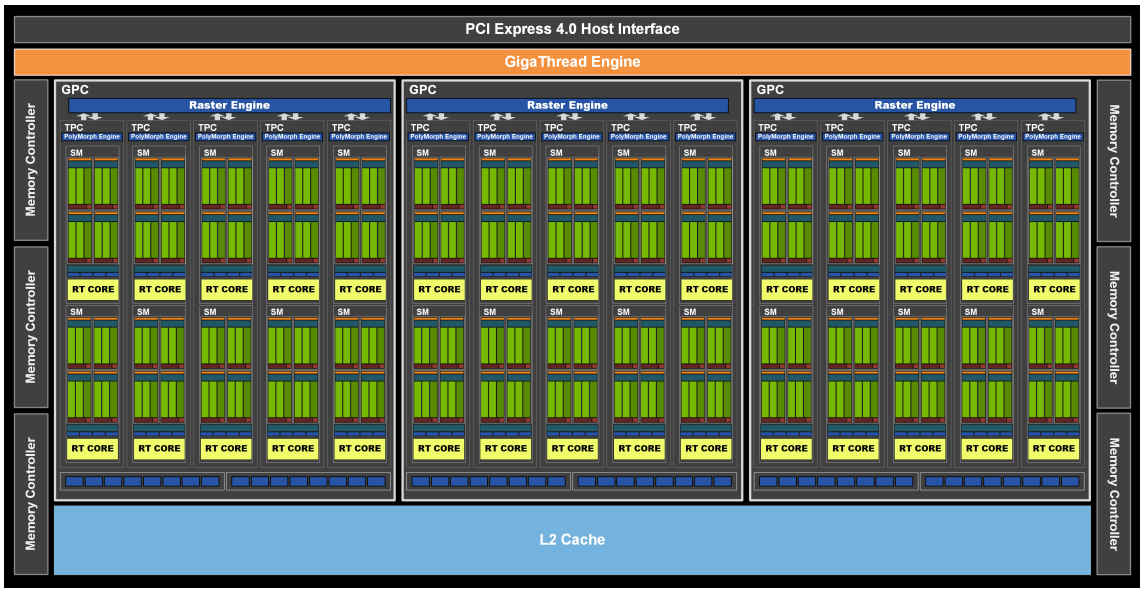


Figure 5.2: NVIDIA GA106 graphics chip scheme.[31]

CHAPTER 6
Metrics

In our experimental study, we explore and utilize three key concepts to analyze and compare probability distributions: the Wasserstein distance, the Kullback-Leibler divergence, and the Jensen-Shannon divergence. These concepts enable us to quantify the dissimilarity or similarity between our learnt distributions and the dataset distributions.

6.1 Wasserstein Distance

The Wasserstein distance, also known as the Earth Mover's distance[34], is a metric used to quantify the dissimilarity between probability distributions. It provides a way to compare the structural differences between distributions by measuring the minimum "cost" required to transform one distribution into another.

Given two probability distributions P and Q defined on a metric space $\mathcal{X}$, the Wasserstein distance $W(P, Q)$ is calculated as the minimum cost of transporting mass from P to Q, where the cost is determined by a ground metric $c(x, y)$ between elements x and y in $\mathcal{X}$. Mathematically, the Wasserstein distance is expressed as:

$$W(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \int_{\mathcal{X} \times \mathcal{X}} c(x, y)\, d\gamma(x, y)$$

Here, $\Gamma(P, Q)$ represents the set of all joint distributions (couplings) γ with marginals P and Q. The ground metric $c(x, y)$ reflects the cost of transporting a unit of mass from x to y. The Wasserstein distance provides a meaningful comparison of distributions in terms of their structure.

6.2 Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence[35], also known as relative entropy, is a measure of the difference between two probability distributions P and Q. It quantifies the amount of information lost when Q is used to approximate P. The KL divergence is defined as:

$$D_{KL}(P \parallel Q) = \int P(x) \log \left(\frac{P(x)}{Q(x)} \right) dx$$

where $P(x)$ and $Q(x)$ are the probability density functions of P and Q respectively. The KL divergence is asymmetric and not a true metric since it does not satisfy the triangle inequality, but we will use it as a point of reference.

6.3 Jensen-Shannon Divergence and Distance

The Jensen-Shannon divergence[36] is a symmetric and smoothed version of the KL divergence previously mentioned in 6.2. It measures the similarity or dissimilarity between two probability distributions P and Q. The Jensen-Shannon divergence is defined as the average of the KL divergence between P and the average distribution M, and the KL divergence between Q and M. Mathematically, it can be expressed as:

$$JSD(P,Q) = \frac{1}{2}D_{KL}(P \parallel M) + \frac{1}{2}D_{KL}(Q \parallel M)$$

where $M = \frac{1}{2}(P+Q)$ is the average distribution. The Jensen-Shannon divergence ranges between 0 and 1, where 0 indicates identical distributions and 1 represents completely dissimilar distributions.

The Jensen-Shannon distance is simply the square root of the Jensen-Shannon divergence, providing a metric for comparing the dissimilarity between probability distributions.

CHAPTER 7
Experimentation

In this section, we focus on the implementation and experimentation that we carried out based on the solution proposed in Chapter 4, using the technologies presented in Chapter 5 and the metrics described in Chapter 6.

Before checking out the results of the experiments, we first specify some details about the partitioning, analysis and preprocessing of the dataset presented in Chapter 3 and the model architecture for each solution.

For simplicity, the particles of the events generated in each experiment have been grouped into logical classes during representation:

- Jets: Which include jets and bjets (j, b).
- Leptons: Including positrons, eletrons, anti-muons and muons (e^+, e^-, m^+, m^-).
- Photons: Just the photons (g).

To be noted, we will also see the distribution of the ordered particles, to appreciate if the p_T is consistent with the ordering, together with the total energy and the total traversal momentum of the events as a measure of conservation these properties[1].

7.1 SM Experiment I: Particle subdivision

For our first experiment, we decided to focus on the **flexibility** of the events generated, mainly on the amount of particles that are observed per event.

Current implementations are limited by the size of event they can generate based on particle amount due to the nature of neural networks. In order to overcome this limitation, we propose the division of the generator in a *model-per-particle* basis, which ensures the appearance of even the most rare combinations and particles.

While this approach is more flexible, it also has a huge handicap, which lies on the fact that the event generated may not be "consistent". Even it if meets all the requirements, the generated data may be implausible, since the models are not connected to each other in any capacity.

Therefore, this approach will only be useful in the presence of a discriminator or a filter, which will, as the current MCMC implementations do, select the events which are valid and discard the other ones.

[1]The total energy has been calculated as the sum of all the energies of the present particles and the MET, and the total traversal momentum as the sum of the traversal momentum p_T of the particles.

All models in this experiment utilize MAFs, as described in the architecture presented in Section 4.2.3. The subsequent sections provide the hyperparameters that define each individual model.

7.1.1. Data partitioning

The data used in our experiments as pointed out before, is that of *ttbar*. As described, we are training a model per particle, including one for MET and METphi, therefore, we decided to partition the original CSV file into eight different files, one per particle. With this, we end up with about 2.2 GB of data in total.

As we can see on Table 7.1, *ttbar* contains a majority amount of jets, followed by the bjets, leptons and photons. This is not surprising, as *ttbar* focuses on "jet-heavy" events, Table 3.2.

File Name	File size (MB)	Particle amount
j_all.csv	1577	19,598,073
b_all.csv	537	6,657,017
m-_all.csv	33	405,903
m+_all.csv	33	403,769
e+_all.csv	25	310,145
e-_all.csv	25	309,770
g_all.csv	9	107,478
met.csv	145	5,412,187
Total	2384	33,204,342

Table 7.1: Particle amount by file for experiment I.

7.1.2. Data analysis and preprocessing

Before we started the experiment, it was important to analyze the dataset presented in Chapter 3 in the search of familiar distributions which have been known to facilitate the learning of Deep Learning models.

After a brief exploration of the data, and taking into account the insights provided by [14], we determined that between the kinematic features that describe each particle (E, p_T, η and ϕ) the Energy E and the Traversal Momentum p_T follow an almost log-normal distribution[2] (see Figure 7.1).

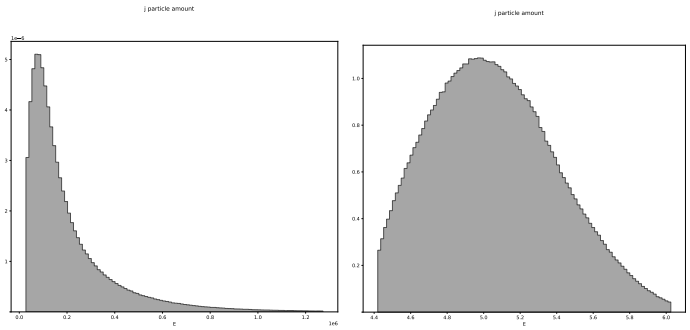


Figure 7.1: Comparison between the original jet energy (left) and the transformed log_{10} energy (right).

[2]The log-normal distribution is a continuous probability distribution that arises when the logarithm of a variable follows a normal distribution.

Therefore, based on these observations, the dataset was preprocessed and transformed accordingly. The logarithmic transformation was then applied to the MET (as can be seen in Figure 7.2), leaving the $MET\phi$ as it was. The aforementioned operation was also utilized in the Energy E and the Traversal Momentum p_T (see Figure 7.3), while the ϕ and η were left as they were in the dataset.

$$
\begin{array}{|l|}
\hline
log_{10}MET,\ \ MET\phi \\
log_{10}MET,\ \ MET\phi \\
. \\
. \\
. \\
log_{10}MET,\ \ MET\phi \\
\hline
\end{array}
$$

Figure 7.2: Format of met.csv on experiment I.

$$
\begin{array}{|l|}
\hline
log_{10}E,\ \ log_{10}p_T,\ \ \eta,\ \ \phi \\
log_{10}E,\ \ log_{10}p_T,\ \ \eta,\ \ \phi \\
. \\
. \\
. \\
log_{10}E,\ \ log_{10}p_T,\ \ \eta,\ \ \phi \\
\hline
\end{array}
$$

Figure 7.3: Format of the particles' CSV files on experiment I.

7.1.3. Model framework

For this first experiment, we decided to employ the MAF model, as described in the proposed solution in Section 4.2.3.

As pointed out before, we will be using one model per particle, including one separate model for MET and $MET\phi$.

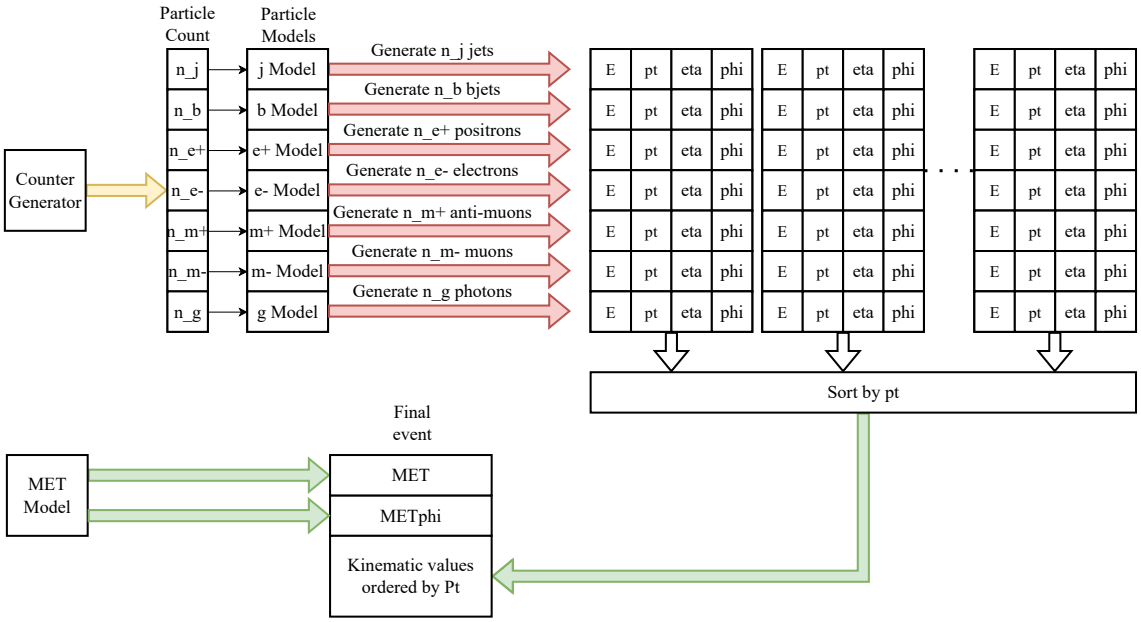


Figure 7.4: Experiment I model architecture.

Therefore, as seen in the figure above (Figure 7.4), our event generator will have essentially three components:

- **MET Model**: A masked autoregressive flow model trained on the distribution of MET and $MET\phi$ whose parameters are listed on Table 7.2 and Table 7.3.
- **Counter Generator**: A discrete 7-dimensional array generator that works as described in Section 4.1.1. It is in charge of providing the amount of each particle present in the current event, following the discrete distribution of the *ground truth*.

- **Particle Model**: A masked autoregressive flow model which has learnt the distribution of the kinematic properties of a particle type. We have seven of them, one per particle, and each one generates as many particles per event as the counter tells it to. Its parameters are listed on Table 7.4 and Table 7.5.

Parameter	Values
MAF layers	10
Permutation layers	9

Table 7.2: MET Model hyperparameters on experiment I.

Parameter	Values
MADE layers	3
Neurons per layer	50

Table 7.3: MET MAF hyperparameters on experiment I.

Parameter	Values
MAF layers	10
Permutation layers	9

Table 7.4: Particle Model hyperparameters on experiment I.

Parameter	Values
MADE layers	3
Neurons per layer	100

Table 7.5: Particle MAF hyperparameters on experiment I.

With this framework, in order to generate an event, we follow these detailed steps in order to guarantee the best performance and accuracy of the generation:

Step 1: Launch the MET Model, which provides the MET and METphi for the current event being generated.

Step 2: Trigger the Counter Generator to provide a 7-dimensional array which describes the number of particles per particle type. With this information, we then use the trained models for each particle to generate as many as needed.

Step 3: Sort the particles by pt, as they do in the dataset described in Chapter 3, and then assemble the event.

Step 4: Check for its validity requirements also indicated in the aforementioned chapter. Accept and save the event if it meets them, discard it if otherwise.

7.1.4. Training results

The training of the models presented in the before section was conducted using the hyperparameters listed in Table 7.2 and Table 7.7. As we can see, **batch size** was tailored to the observed amount of each particle in Table 7.1.

Parameter	Values
Optimizer	Adam[37]
Initial learning rate	10^{-4}
Loss function	Log-likelihood
Batch size	1024

Table 7.6: MET Model training parameters on experiment I.

Parameter	Values
Optimizer	Adam[37]
Initial learning rate	10^{-4}
Loss function	Log-likelihood
Batch size (j)	1024
Batch size (b)	512
Batch size (e^-, e^+, m^-, m^+)	32
Batch size (g)	16

Table 7.7: Particle Model training parameters on experiment I.

All the training procedures were conducted to up to 100 **epochs**[3], which resulted in the best results for this architecture.

MET and METϕ

As can be seen in Figure 7.5, the model that is in charge of the generation of MET and $MET\phi$ is able to reproduce accurately the distribution of these properties, accomplishing good results based on our metrics.

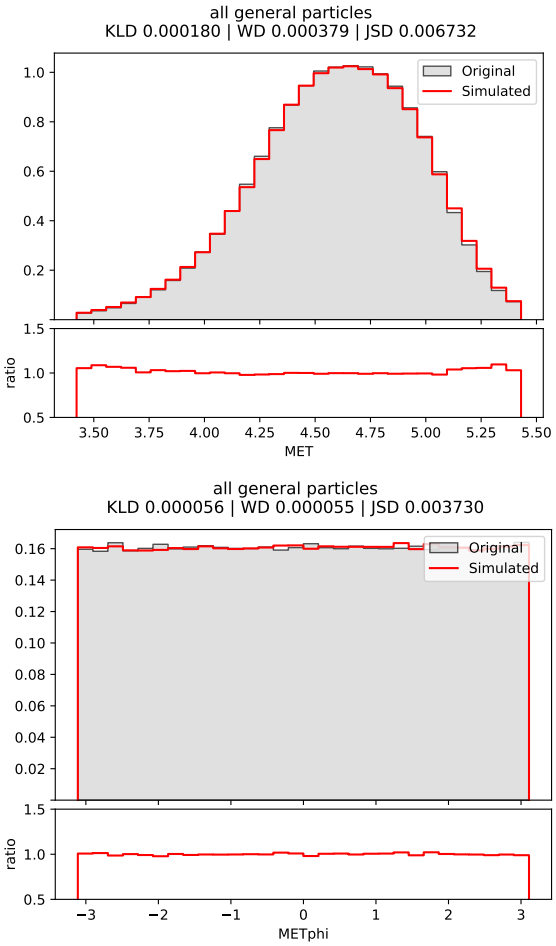


Figure 7.5: Histograms of MET and $MET\phi$ on experiment I.

[3] A single iteration through the entire dataset during model training.

Jets

For the model trained on all the jets present on *ttbar*, we can appreciate that it is capable to generate jets that follow the original distribution almost perfectly (see Figure 7.6). The biggest divergence can be found on the p_T distribution, which has trouble fitting the original data due to its unusual shape.

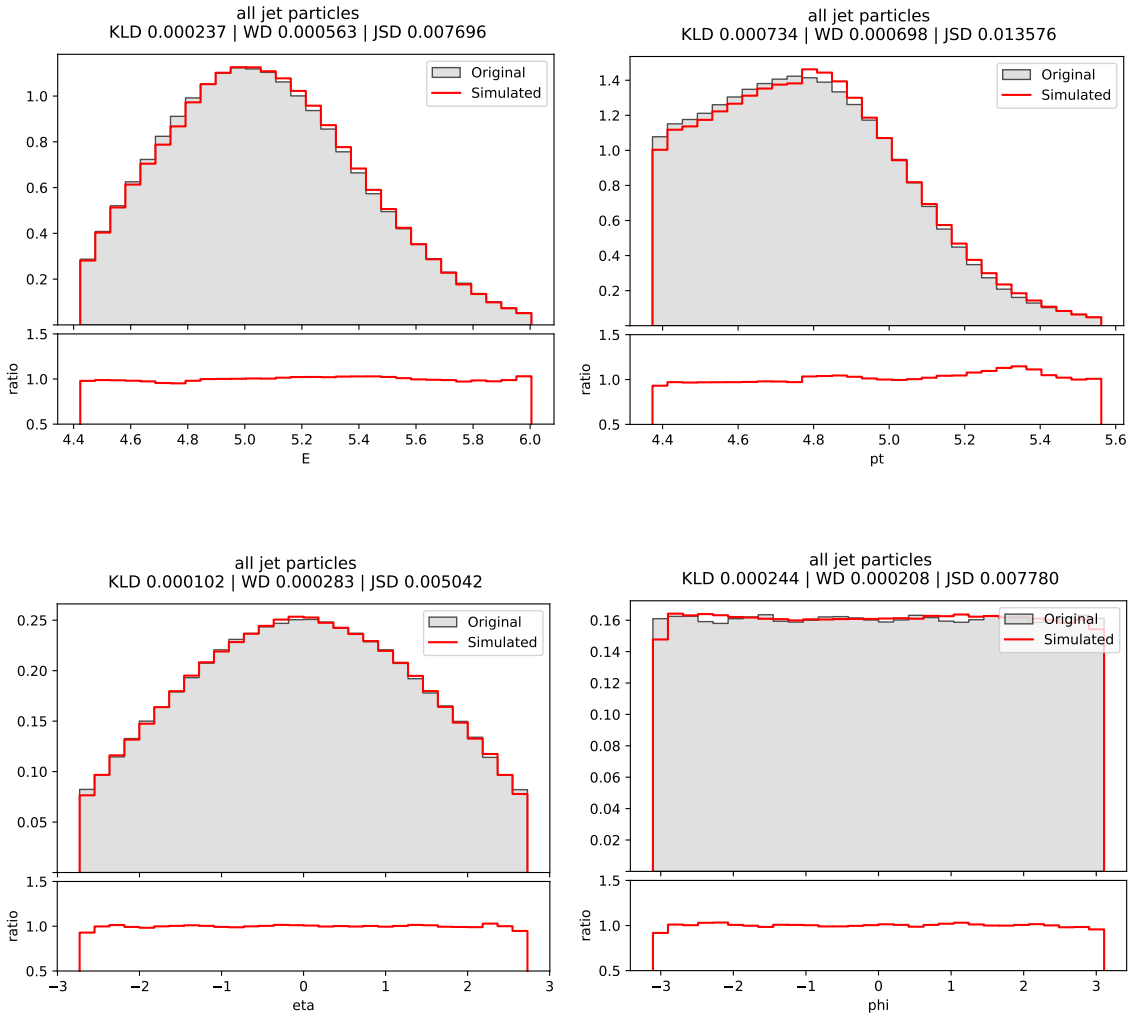


Figure 7.6: Histograms of the kinematic properties of all the jets (j, b) on experiment I.

But as expected, discrepancies may occur once we separate the jets by order. Even if the model can generate random jets perfectly, the small discrepancies in the learnt p_T propagate through the ordered jets.

The energy E loses some accuracy for the first jet, but stays mostly consistent for the rest of the jets (Figure 7.7), being able to produce the correct skewed distribution on the third and next jets.

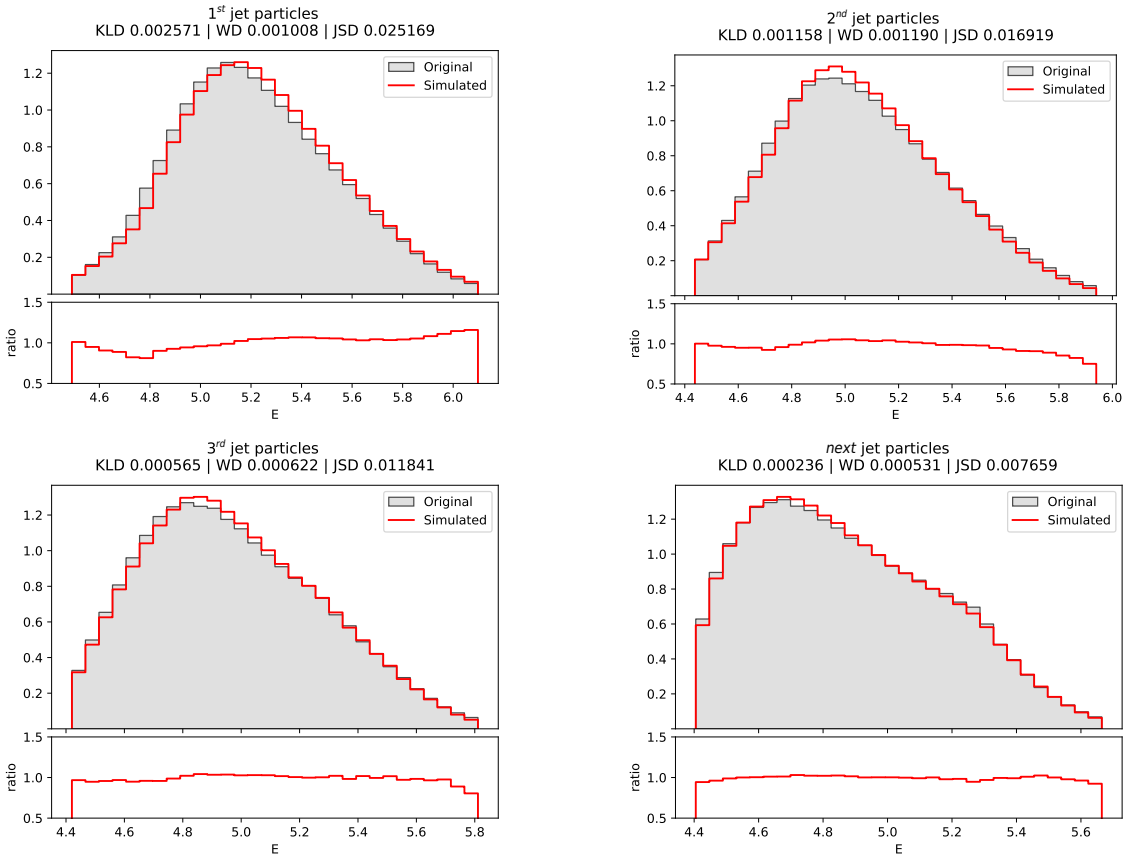
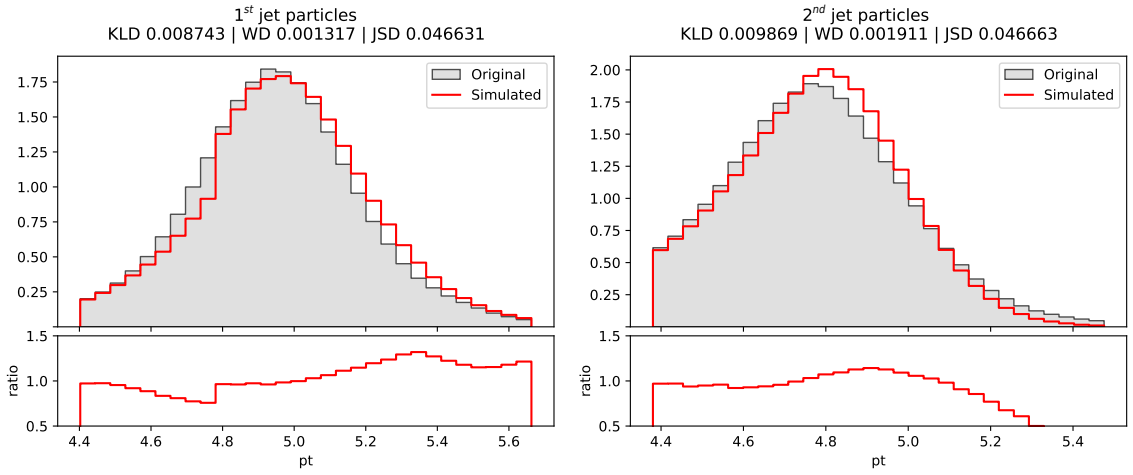


Figure 7.7: Histograms of the energy of the ordered jets (j, b) on experiment I.

For the traversal momentum p_T, we see a bigger disparity as pointed out before, having the first jet a displaced distribution by a small margin, while the other jets can't quite get the right proportions (Figure 7.8).



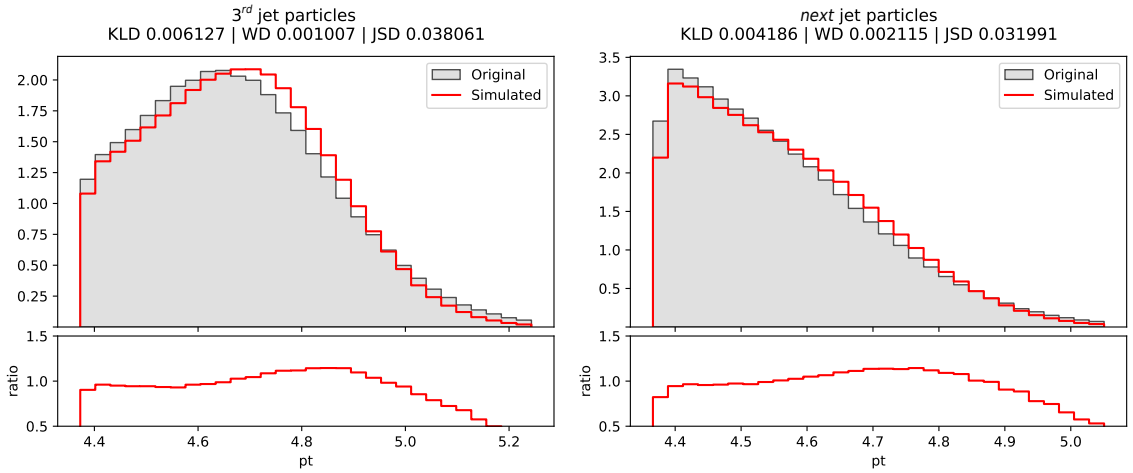


Figure 7.8: Histograms of the traversal momentum of the ordered jets (j, b) on experiment I.

In the case of the pseudo-rapidity η, even if the distribution is strange, it does not present a challenge for the model (Figure 7.10).

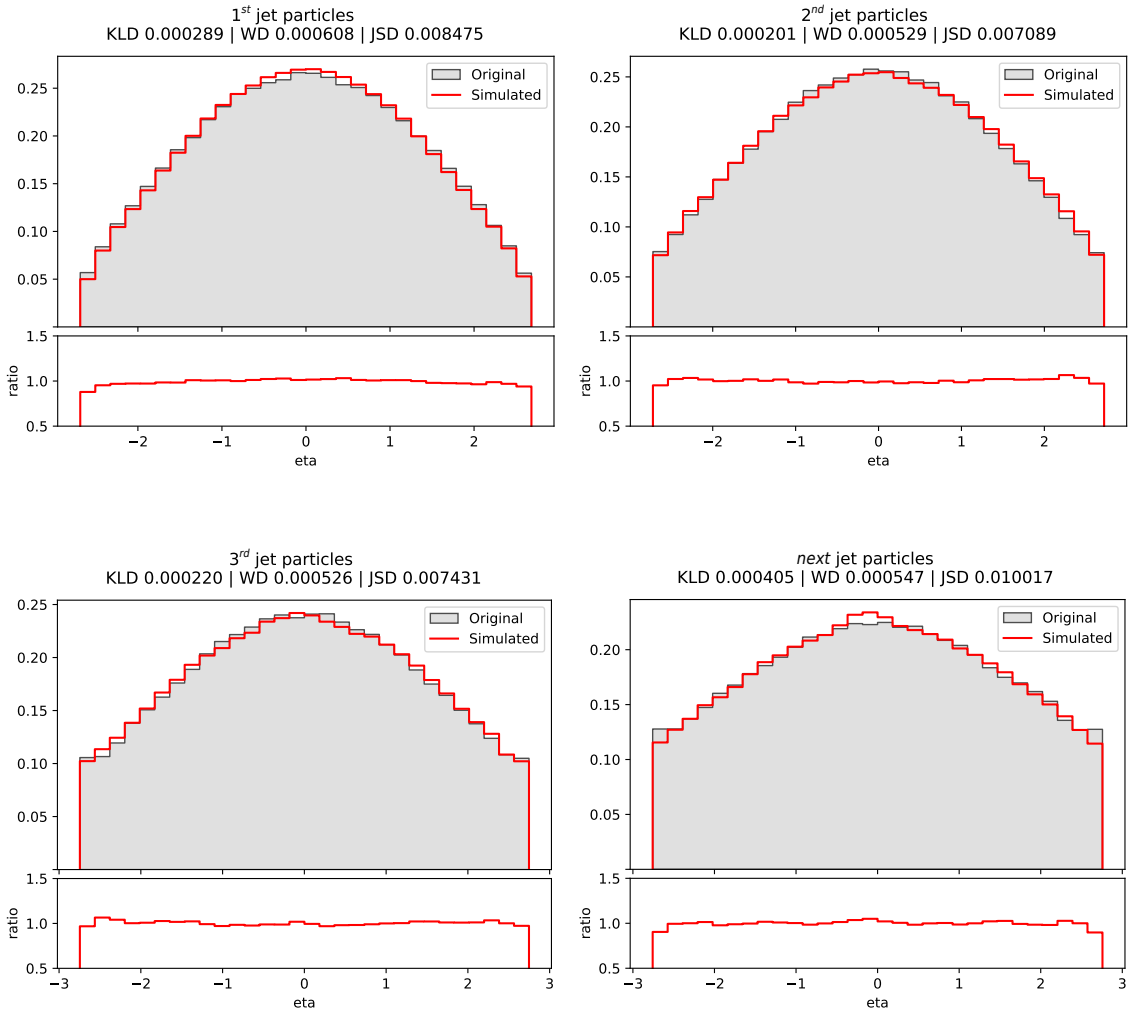


Figure 7.9: Histograms of the pseudo-rapidity of the ordered jets (j, b) on experiment I.

Finally, the azimuth angle ϕ, which presents itself as a uniform distribution (Figure 7.10), can also be easily simulated by the jets model.

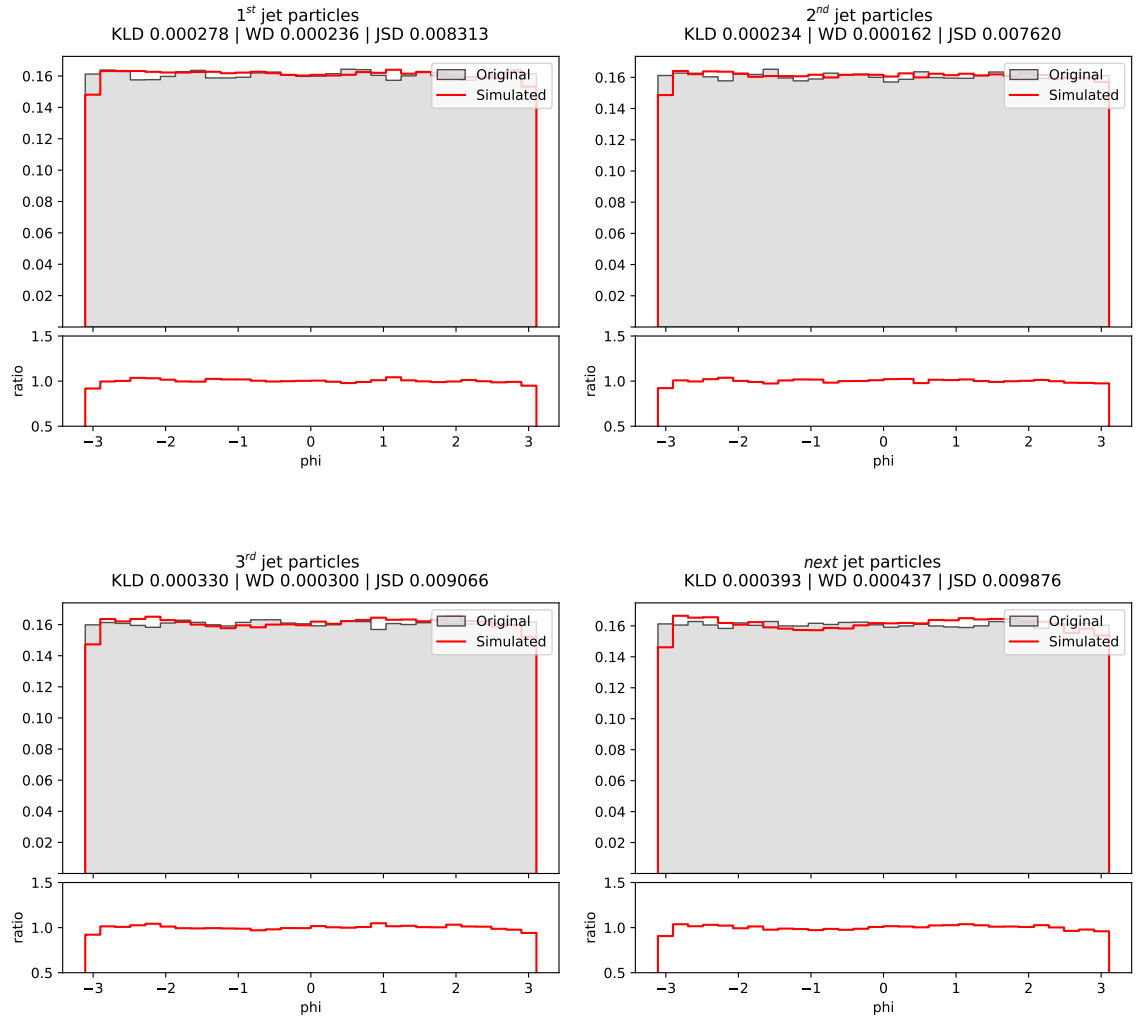
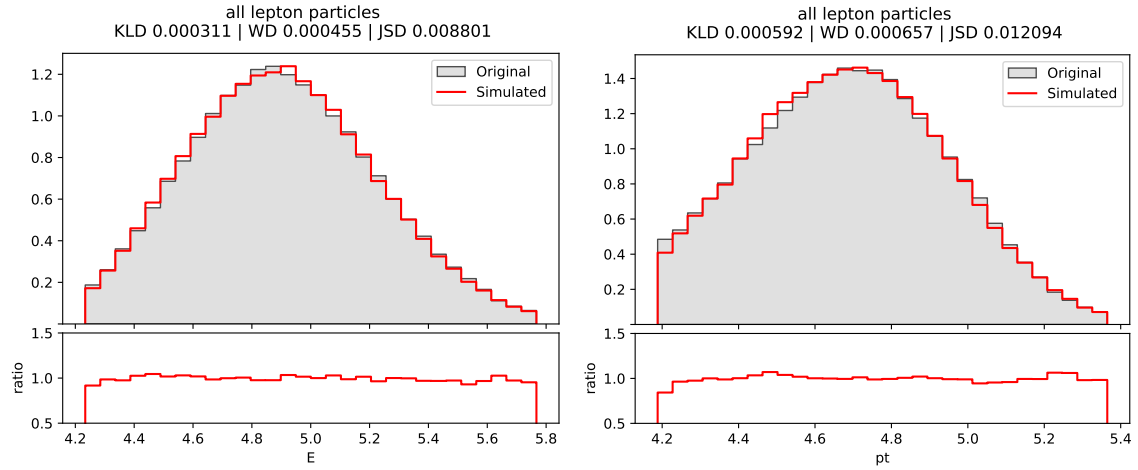


Figure 7.10: Histograms of the azimuth angle of the ordered jets (j, b) on experiment I.

Leptons

In the case of the leptons model, we can observe that the simulated distribution does a good job at imitating the original (Figure 7.11). Even though it is not as exact as the jets model, this can be explained by the difference in data to be trained on, as seen in Table 7.1.



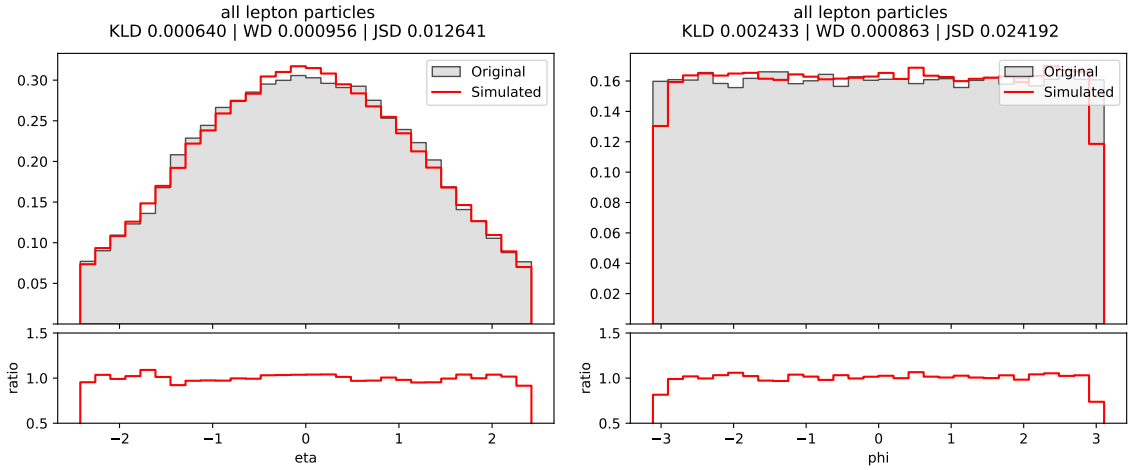


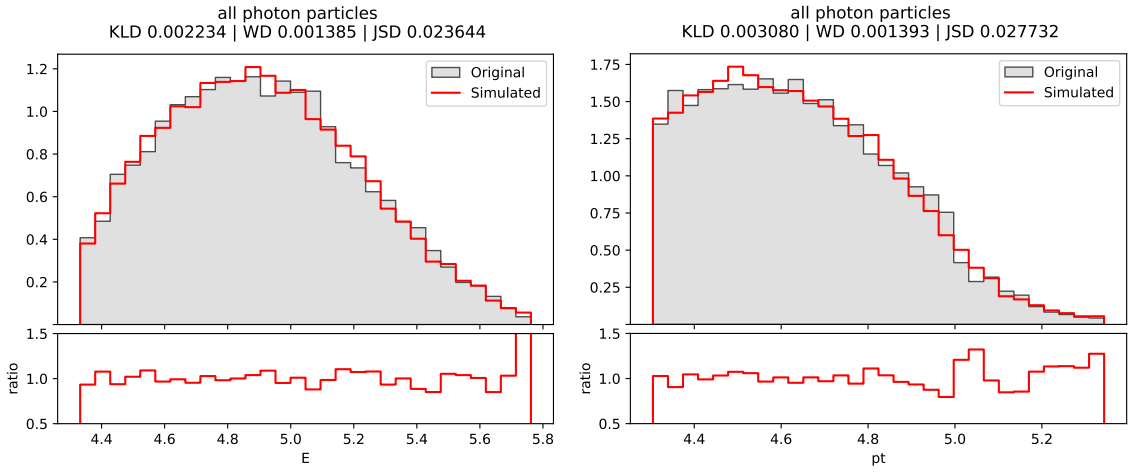
Figure 7.11: Histograms of the kinematic properties of all leptons (e^-, e^+, m^-, m^+) on experiment I.

Photons

Lastly, when it comes to the model responsible for generating photons, it is not surprising that it provides suboptimal results. This outcome was anticipated because the dataset contains only approximately 107 thousand photon particles, which accounts for a mere 0.32% of the total observed particles.

Despite the lack of training data, the model manages to capture the general trends and patterns of the distributions, with one exception being the pseudo-rapidity (η) due to its atypical shape.

The limited amount of data poses a challenge for accurately learning the complex characteristics of photons, yet the model still demonstrates a capacity to capture certain underlying tendencies within the available particle samples.



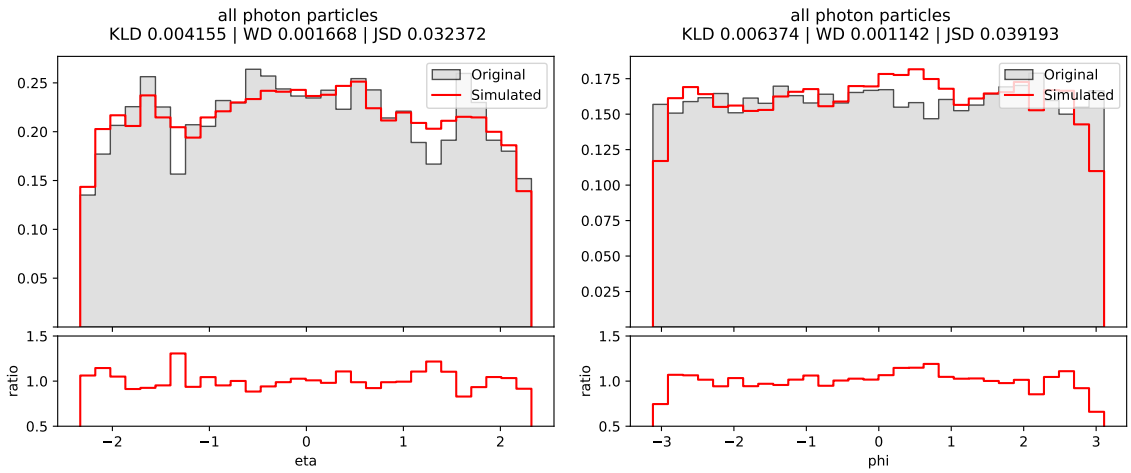


Figure 7.12: Histograms of the kinematic properties of all photons (g) on experiment I.

Conservation of energy and traversal momentum

We represent the total energy and the total traversal momentum in Figure 7.13. In this first experiment, we can appreciate a notable disparity between the *ground truth* and the generated values.

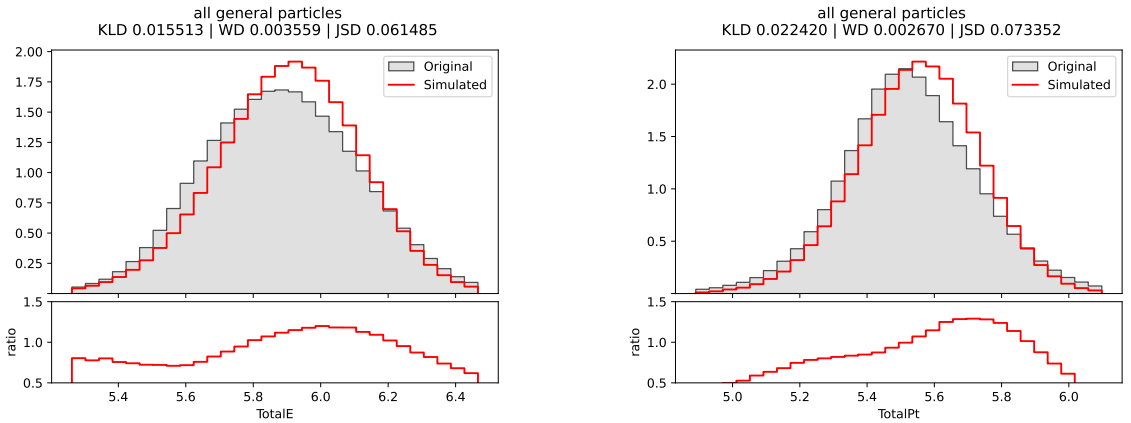


Figure 7.13: Histograms of the total energy and the total momentum in experiment I.

This was to be expected, as the particle models are not connected in any way, which can result in events that posses uncorrelated energy and traverse momentum amounts.

7.2 SM Experiment II: Cumulative particle subdivision

In this second experiment, we try to improve on the framework of experiment I. We attempt to maintain the **flexibility** that we explained before, while increasing the **correlation** between the generated particles.

In order to do this, we decided to train two models per particle. The first one will only focus on the first appearance of each particle in an event, and the second one on the next appearances on an event. In order to facilitate the needed operations to accomplish more particle correlation, the models were given more information, which we will describe in the following sections.

Even with this new procedure, plausibility of the generated events cannot be guaranteed, even if it meets all the requirements exposed in Chapter 3. Therefore, as exposed in the before experiment, a discriminator or filter would prove necessary to guarantee the validity of the generated events.

This experiment employs MAFs in all its models, following the architecture presented on Section 4.2.3. The hyperparameters that define each model are presented in the sections that come.

7.2.1. Data partitioning

As previously mentioned, our experiments utilize the *ttbar* dataset. To train the two models for each particle, including MET and $MET\phi$, we made the decision to divide the original CSV file into 15 separate files, two per particle, and one for the MET. This partitioning resulted in a total data size of approximately 2.2 GB.

Table 7.8 illustrates the distribution of particle types in the *ttbar* dataset. Jets constitute the majority, followed by bjets, leptons, and photons. This observation aligns with the one made in experiment I, as shown in Table 3.2.

Given the substantial amount of data, we opted to batch the files appropriately to optimize memory usage and training speed, without compromising the accuracy of the models.

File Name	File size (MB)	Particle amount
j_1.csv	428	5,315,678
j_next.csv	1664	14,282,395
b_1.csv	359	4,453,838
b_next.csv	257	2,203,179
m-_1.csv	33	405,777
m-_next.csv	1	126
m+_1.csv	33	403,664
m+_next.csv	1	105
e+_1.csv	25	310,108
e+_next.csv	1	37
e-_1.csv	25	309,730
e-_next.csv	1	40
g_1.csv	9	106,641
g_next.csv	1	837
met.csv	145	5,412,187
Total	2983	33,204,342

Table 7.8: Particle amount by file on experiment II.

7.2.2. Data analysis and preprocessing

For the preprocessing of the data of this experiment, we took into account the observations made on experiment I (Section 7.1.2) and on [14].

As mentioned in the introduction to this experiment, its models have been trained with more data than the models on experiment I. Concretely, for the models that generate first particles, we have included the MET and $MET\phi$ as dimensions of the event they belong to (illustrated on Figure 7.15), and for the *next* models, we have also included the cumulative energy and cumulative traversal momentum of that particle type (as seen on Figure 7.15). The logarithmic transformation has been applied as expected on the MET, energy E, traversal momentum p_T, cumulative energy $cumE$ and cumulative traversal momentum $cump_T$.

Finally, it is worth mentioning that the file *met.csv* (Figure 7.14) has been left the same as on experiment I, therefore the trained model will be the same.

$$
\begin{array}{|l|}
\hline
log_{10}MET,\ MET\phi \\
log_{10}MET,\ MET\phi \\
. \\
. \\
. \\
log_{10}MET,\ MET\phi \\
\hline
\end{array}
$$

Figure 7.14: Format of met.csv on experiment II.

$$
\begin{array}{|l|}
\hline
log_{10}MET,\ MET\phi,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
log_{10}MET,\ MET\phi,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
. \\
. \\
. \\
log_{10}MET,\ MET\phi,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
\hline
\end{array}
$$

Figure 7.15: Format of the first particles' CSV files on experiment II.

$$
\begin{array}{|l|}
\hline
log_{10}MET,\ MET\phi,\ log_{10}cumE,\ log_{10}cump_T,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
log_{10}MET,\ MET\phi,\ log_{10}cumE,\ log_{10}cump_T,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
. \\
. \\
. \\
log_{10}MET,\ MET\phi,\ log_{10}cumE,\ log_{10}cump_T,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
\hline
\end{array}
$$

Figure 7.16: Format of the next particles' CSV files on experiment II.

7.2.3. Model framework

As said before, for this second experiment we will use the same model as for the first, the MAF model, as proposed in Section 4.2.3. We will be employing two models per particle, including one separate model for MET and $MET\phi$.

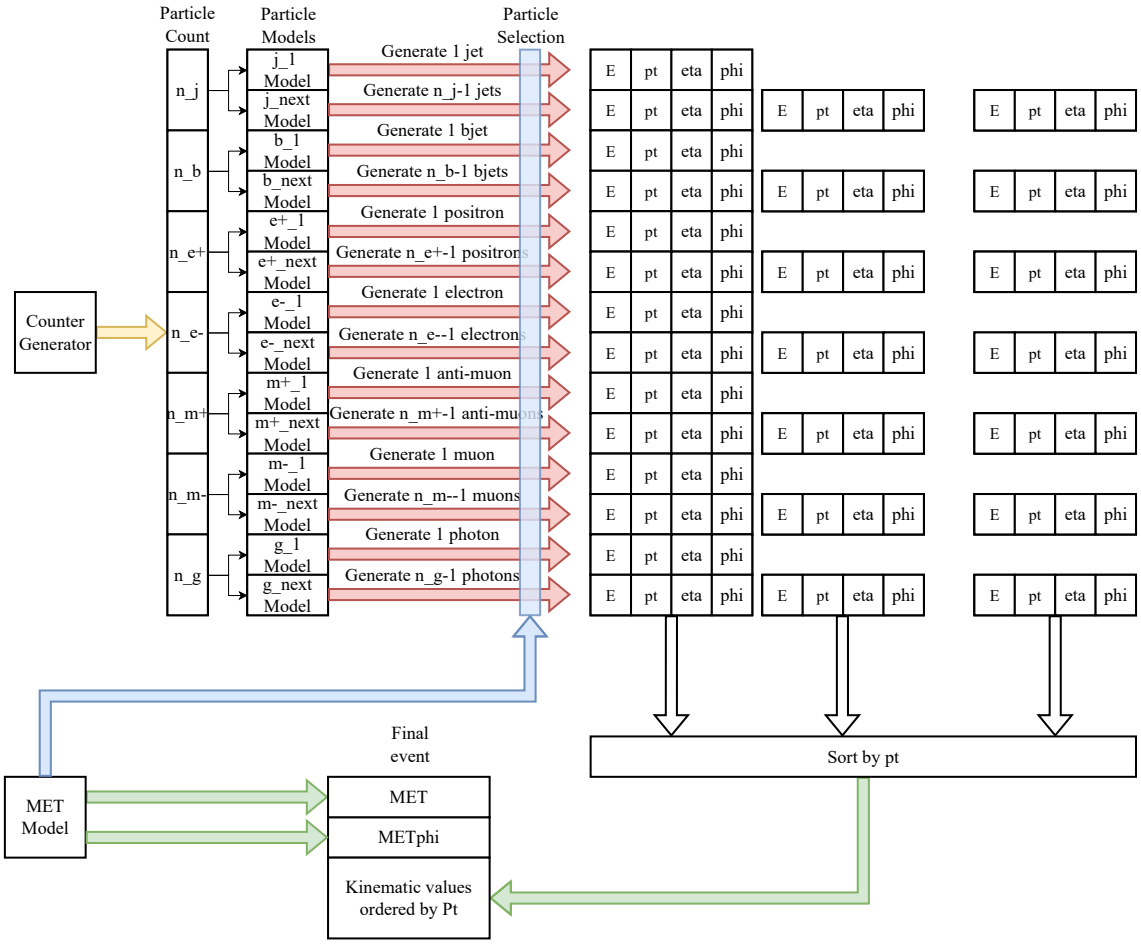


Figure 7.17: Experiment II model architecture.

Following the figure above (Figure 7.17), our event generator will have mainly four distinct components:

- **MET Model**: A masked autoregressive flow model trained on the distribution of MET and METphi whose parameters are listed on Table 7.9 and Table 7.10.
- **Counter Generator**: A discrete 7-dimensional array generator, explained in Section 4.1.1, that provides the particle quantities in the current event. It follows the discrete distribution of the *ground truth*.
- **Particle Models**: A masked autoregressive flow model that has learned the distribution of kinematic properties for each particle type. There are fourteen particle models, two per particle, and they generate the specified number of particles per event based on the counter. The model's parameters can be found in Table 7.11 and Table 7.12.
- **Particle Selector**: Selection made over a batch of generated particles of each type based on the minimization of the euclidean distance.

– For first particles, the distance to minimize is the following:

$$((MET - SampleMET)^2 + (MET\phi - SampleMET\phi)^2)^{1/2}$$

– For second particles, we need to minimize the next function:

$$((MET - SampleMET)^2 + (MET\phi - SampleMET\phi)^2 +$$
$$+(CumE - SampleCumE)^2 + (Cump_T - SampleCump_T)^2)^{1/2}$$

Parameter	Values
MAF layers	10
Permutation layers	9

Table 7.9: MET Model hyperparameters on experiment II.

Parameter	Values
MADE layers	3
Neurons per layer	50

Table 7.10: MAF hyperparameters (MET) on experiment II.

Parameter	Values
MAF layers	10
Permutation layers	9

Table 7.11: Particle Model hyperparameters on experiment II.

Parameter	Values
MADE layers	3
Neurons per layer	100

Table 7.12: MAF hyperparameters (Particle) on experiment II.

With this framework, in order to generate an event, we follow these detailed steps in order to guarantee an improvement on particle generation correlation from experiment I:

Step 1: Launch the MET Model, which provides the MET and $MET\phi$ for the current event being generated. This values will work as the "seed" of the event.

Step 2: Trigger the Counter Generator to provide a 7-dimensional array which describes the number of particles per particle type.

Step 3: If it precedes, use the first particle generator to produce 1000 particles of the type that is needed. Select the one sample that minimizes the euclidean distance as described before.

Step 4: If there are more particles after the first one, use the second particle generator to produce 1000 particles of the type that is needed. Select the one sample that minimizes the euclidean distance as described before for the second particles. The $cumE$ and $cump_T$ will need to be added as the generation continues per particle type.

Step 5: Sort the particles by p_T, as they do in the dataset described in Chapter 3, and then assemble the event.

Step 6: Check for its validity requirements also indicated in the aforementioned chapter. Discard the event if it's not valid, save it if otherwise.

Following the aforementioned instructions, we generated enough events to visualize how this approach fits the training data.

7.2.4. Training results

The optimization of the models presented was done using the training hyperparameters listed in Table 7.13 and Table 7.14.

Parameter	Values
Optimizer	Adam[37]
Initial learning rate	10^{-4}
Loss function	Log-likelihood
Batch size	1024

Table 7.13: MET Model training parameters on experiment II.

Parameter	Values
Optimizer	Adam[37]
Initial learning rate	10^{-4}
Loss function	Log-likelihood
Batch size (j_1, b_1)	256
Batch size (j_{next})	1024
Batch size (b_{next})	512
Batch size $(e_1^-, e_1^+, m_1^-, m_1^+)$	16
Batch size $(g_1, g_{next}, e_{next}^-, e_{next}^+, m_{next}^-, m_{next}^+)$	2

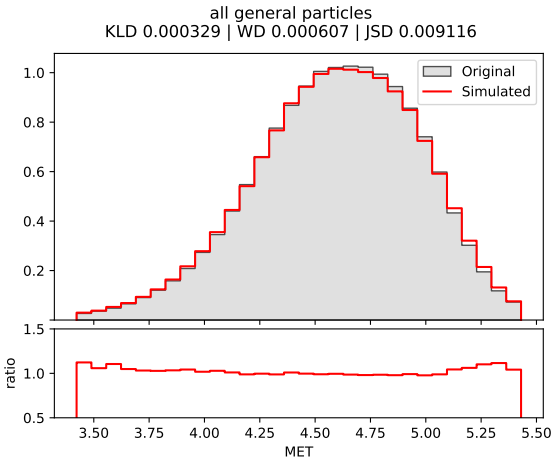
Table 7.14: Particle Model training parameters on experiment II.

The models learnt using various procedures, with each training session spanning up to 100 **epochs**.

MET and METϕ

The MET Model perfomed as expected, since it is exactly the same architecture as the MET Model presented on the first experiment.

It is then able to reproduce the distributions almost perfectly (see Figure 7.18), which in this case has more importance, as it serves as the seed for the particle models to generate.



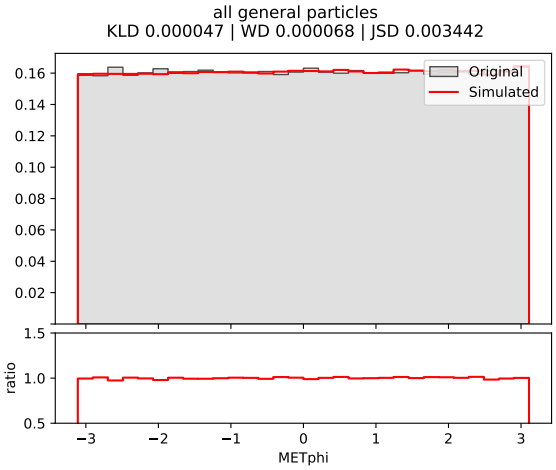


Figure 7.18: Histograms of MET and $MET\phi$ on experiment II.

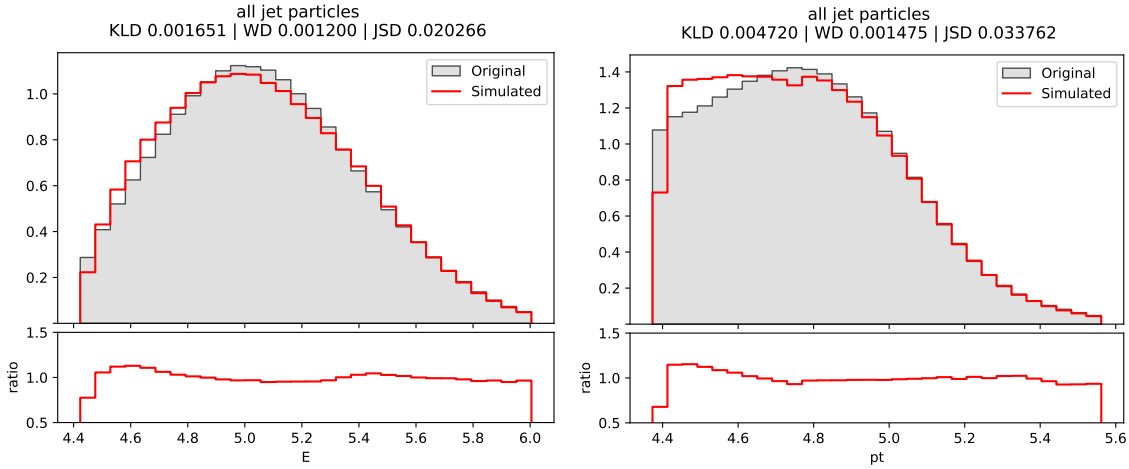
Jets

In the analysis of jet models in particle physics, both models demonstrate impressive performance concerning the overall distribution of jet particles, as illustrated in Figure 7.19.

The transverse momentum, denoted as p_T, appears to have the least accurate distribution fit among the studied variables. This discrepancy can be attributed to the prevalence of lower transverse momentum jets in the second model's training data

The second model is responsible for generating up to 13 jets that succeed the initial jet. As discussed in Chapter 3's dataset description, higher-order jets typically possess lower p_T values. Consequently, the divergence in the p_T distribution becomes more pronounced when examining the jets by order.

By expanding upon the jets' order, the divergence in the p_T distribution will become more evident, providing valuable insights into the performance of the jet models.



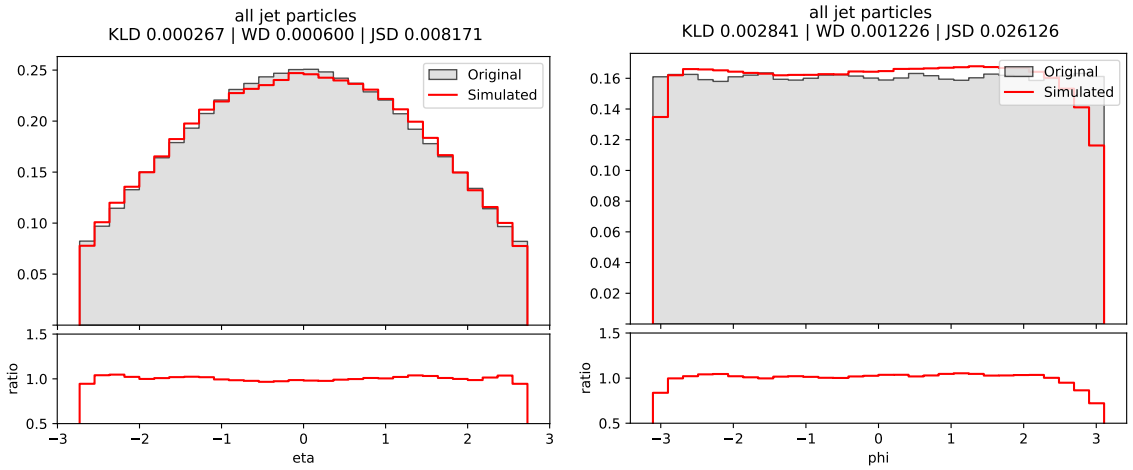


Figure 7.19: Histograms of the kinematic properties of all the jets (j, b) on experiment II.

When analyzing the jets by order, we can observe how many disparities arise mostly on the second model's task. Figure 7.20 illustrates the ordered distributions, where the first jet model behaves as expected, while the second, third and next distributions are not well fitted by the second model.

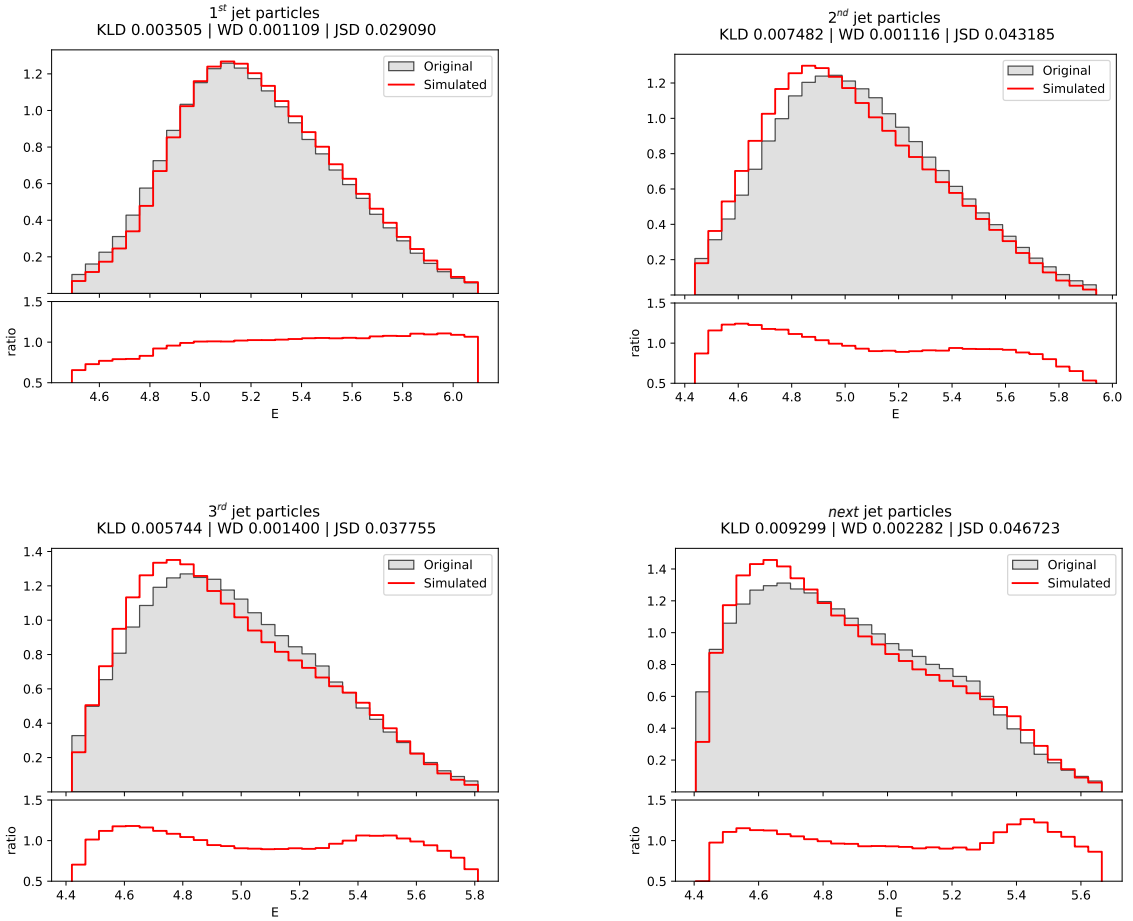


Figure 7.20: Histograms of the energy of the ordered jets (j, b) on experiment II.

As pointed out in the description of Figure 7.19, the discrepancies on the learnt distribution of the traverse momentum p_T propagate through the ordered jets, providing an

irregular fit over the data, while slightly approaching its distribution tendency, as seen in Figure 7.21.

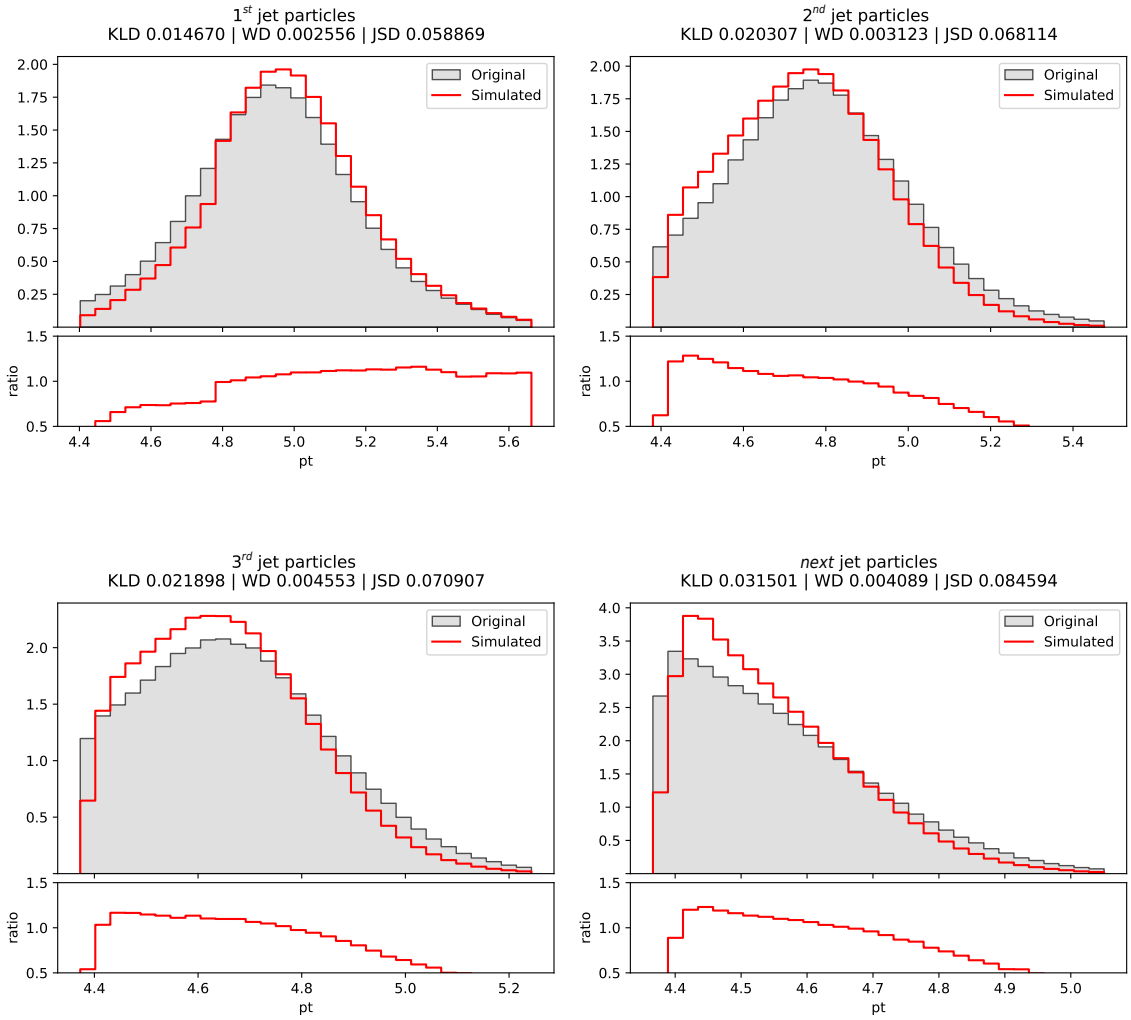
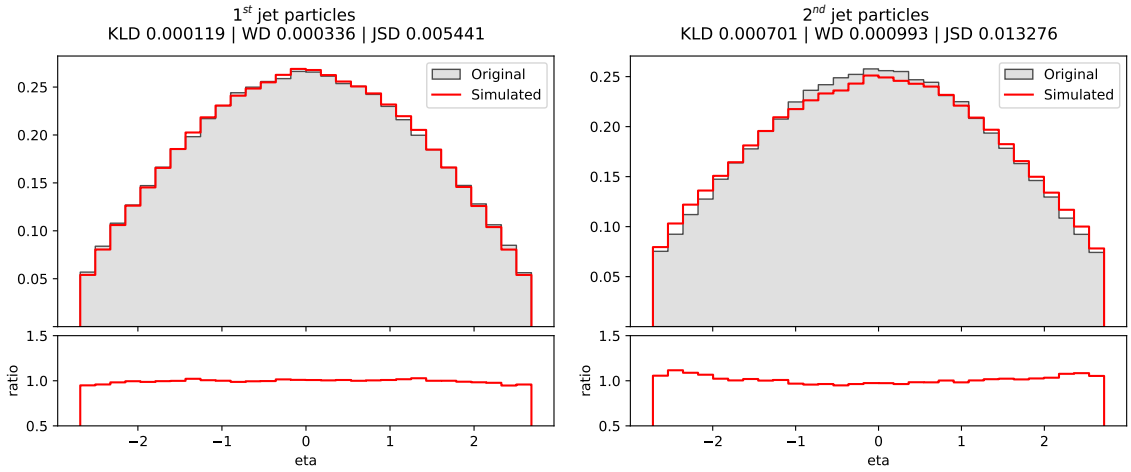


Figure 7.21: Histograms of the traversal momentum of the ordered jets (j, b) on experiment II.

The distribution of the pseudo-rapidity η poses no challenge for the models as shown in Figure 7.22.



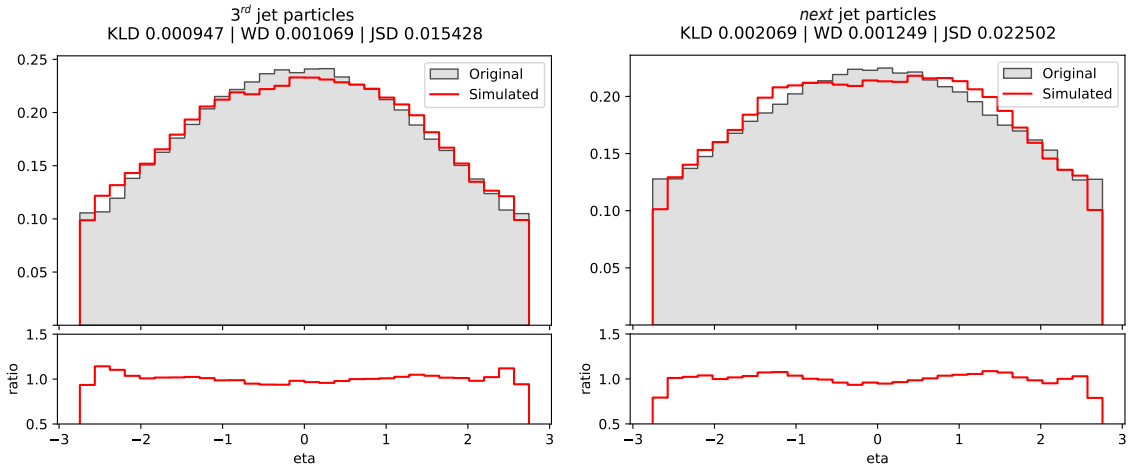


Figure 7.22: Histograms of the pseudo-rapidity of the ordered jets (j, b) on experiment II.

The azimuth angle ϕ, can be seen following a uniform distribution in Figure 7.23. This property is well fitted by the first jet model, but the second jet model struggles on the edge values.

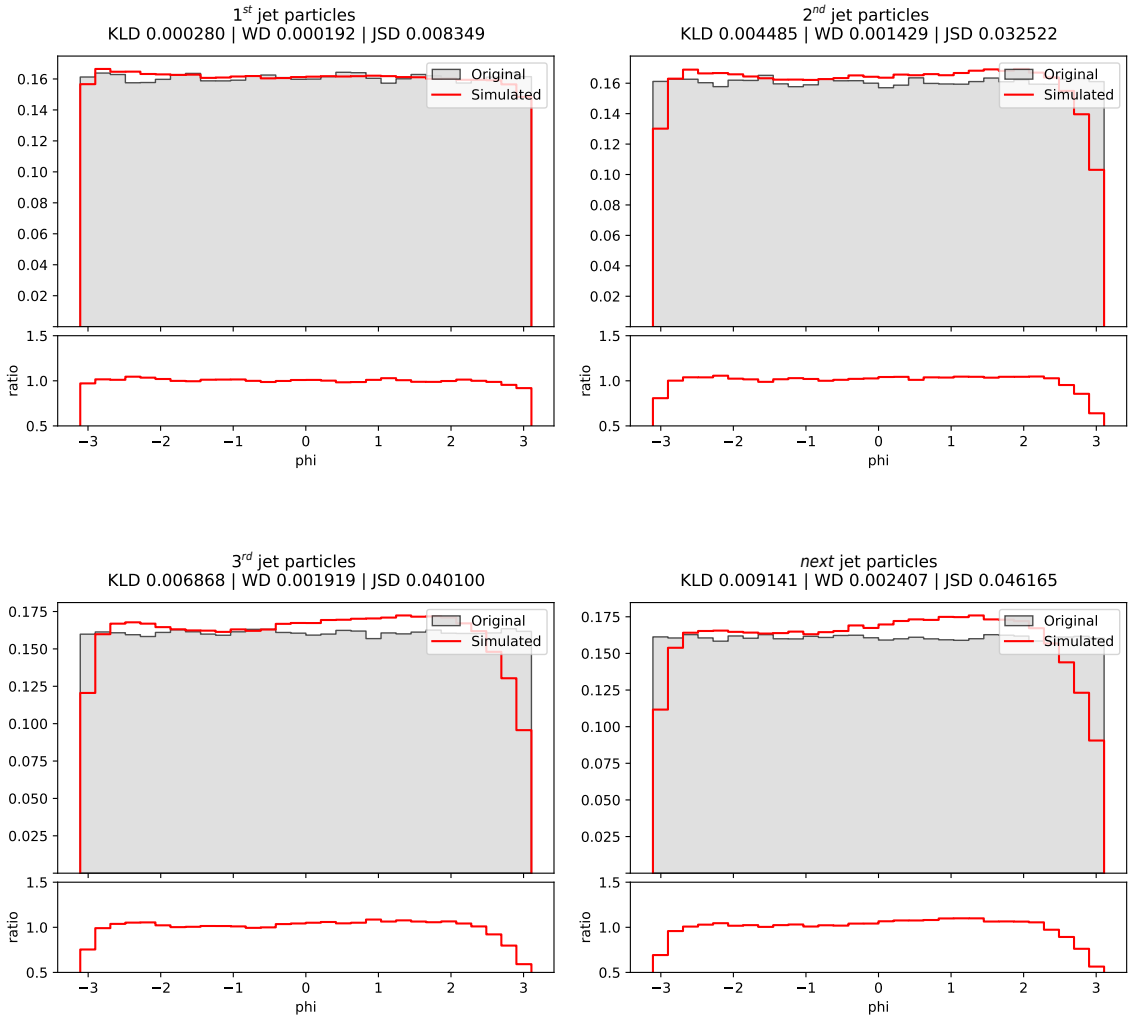


Figure 7.23: Histograms of the azimuth angle of the ordered jets (j, b) on experiment II.

Leptons

Both lepton models do a great job at following the original kinematic properties' distributions (Figure 7.24). However, they suffer from the same problem as the jets model did, the learnt p_T has a larger amount of lower traversal momentum values than originally.

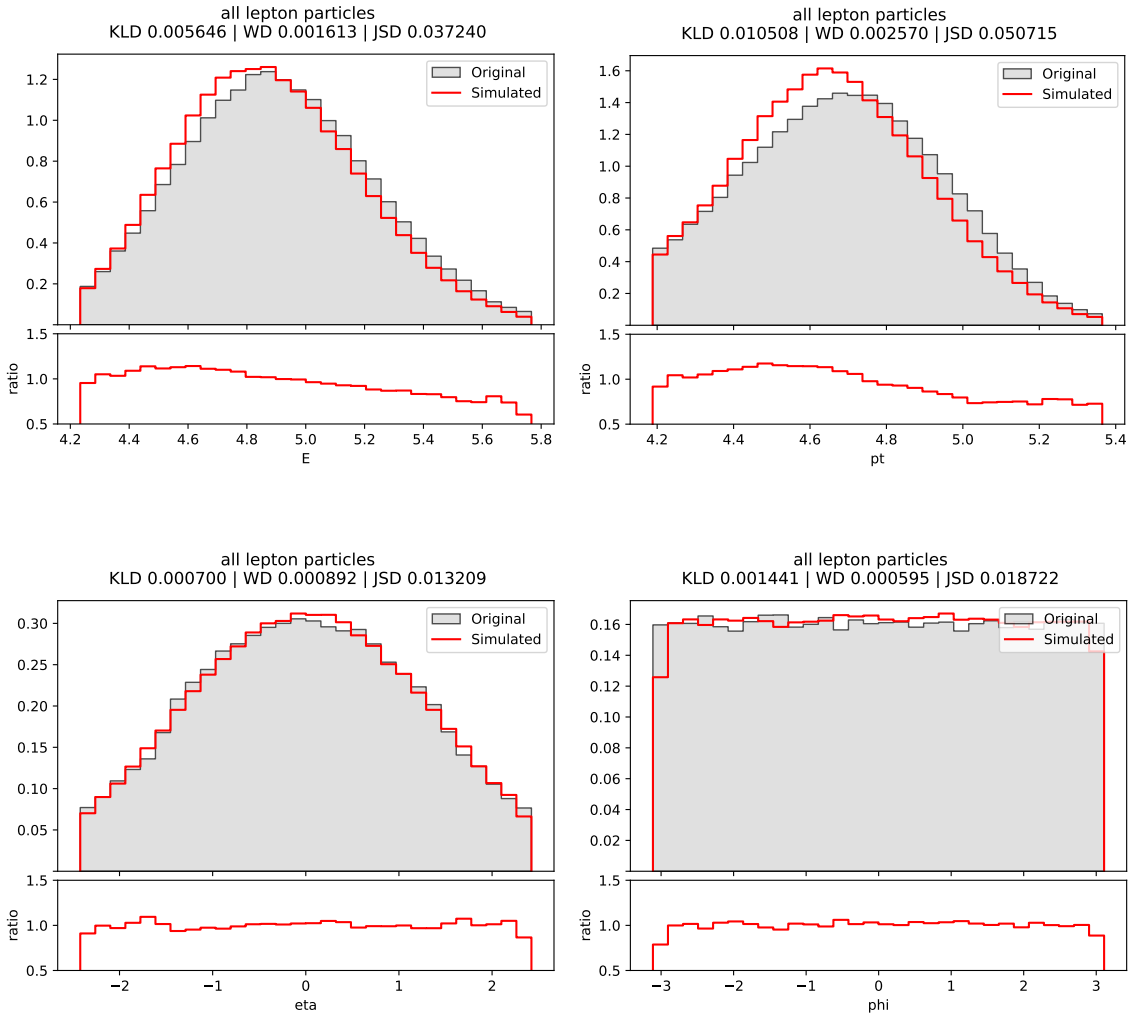


Figure 7.24: Histograms of the kinematic properties of all leptons (e^-, e^+, m^-, m^+) on experiment II.

Photons

Finally, the photons models work surprisingly well (Figure 7.25), mostly struggling on the pseudo-rapidity η.

The models are mostly able to learn the tendencies of the distributions and can produce sound photons.

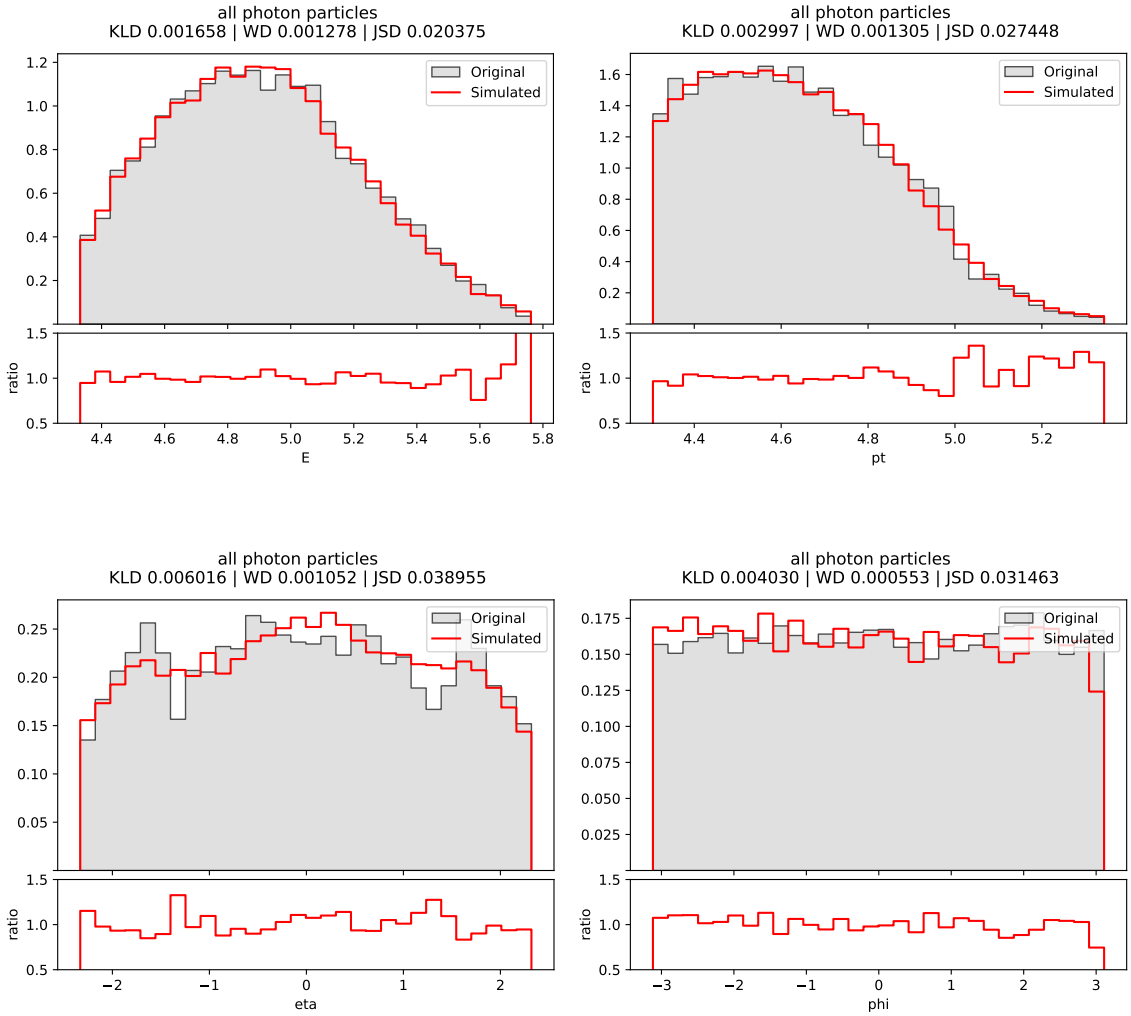


Figure 7.25: Histograms of the kinematic properties of all photons (g) on experiment II.

Conservation of energy and traversal momentum

Experiment II failed to improve the correlation between energy and traverse momentum among particles compared to experiment I. The model still struggles to generate events at the distribution extremes 7.26.

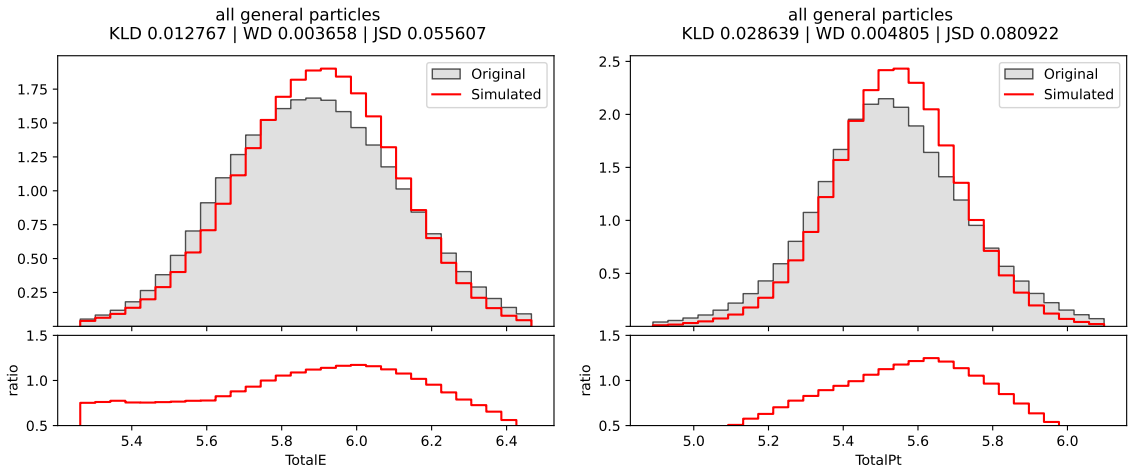


Figure 7.26: Histograms of the total energy and the total momentum in experiment II.

7.3 SM Experiment III: Tailored generator

This third and final experiment on the Standard Model dataset was dedicated to producing **plausible samples**, sacrificing the flexibility that the two experiments presented before accomplished.

As this experiment employs only one model in charge of learning and generating entire events at once, it has the ability to find inter-relationships between the particles in the events, as opposed to the experiments that came before. Therefore, we believe there is little need for a filter or discriminator, since we trust *most* of the data generated will be believable.

Due to the high dimensionality required per event, which we will detail further in the next sections, a more flexible and adaptable model was needed. Therefore, for this experiment we decided to use the A-RQS architecture, presented in Section 4.2.4, which has proven to work better than the other model architecture in this study for high dimensional data.

7.3.1. Data partitioning

Since in this approach we will only be training one model, the *ttbar* dataset has had to be adapted. A series of filters based on the need of the model training were put in place.

As the model has a fixed amount of maximum particles per particle type in an event, which is displayed in the table below, only events that did not go over these particle amounts would be considered valid.

Particle Type	j	b	e^+	e^-	m^+	m^-	g
Maximum amount	4	2	2	2	2	2	2

Table 7.15: Maximum amount of each particle type in an event on experiment III.

Therefore, by that criteria, all events which had more than the number specified in Table 7.15 of a certain particle type, were discarded, leaving us with the amount displayed in Table 7.16, just about 1.7 GB.

File Name	**File size (MB)**	**Event amount**
ttbar_conditioned.csv	1657	3,688,244
Total	1657	3,688,244

Table 7.16: Event amount in processed file on experiment III.

7.3.2. Data analysis and preprocessing

Bearing in mind the findings on [14], as we did in experiment I and II, we decided to carry on applying the logarithmic transformation to the MET and $MET\phi$, together with the E and p_T of each particle.

For this last experiment, as we already mentioned, we are using a single model to generate the entirety of the event. As presented on previous section, we have limited our sample size to a maximum of 66 dimensions: 2 belonging to MET and $MET\phi$, and the 64 remaining shared by the particles.

In the events saved after processing, which can be seen on Figure 7.27, we will have the particles ordered as they have been listed on Table 7.15. In the case of a particle not appearing on an event, its kinematic values were set to 0, symbolizing its absence.

$$
\begin{array}{l}
log_{10}MET,\ MET\phi,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi,\ \cdot\ \ \cdot\ \ \cdot\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
log_{10}MET,\ MET\phi,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi,\ \cdot\ \ \cdot\ \ \cdot\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi \\
. \\
. \\
. \\
log_{10}MET,\ MET\phi,\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi,\ \cdot\ \ \cdot\ \ \cdot\ log_{10}E,\ log_{10}p_T,\ \eta,\ \phi
\end{array}
$$

Figure 7.27: Format of ttbar_conditioned.csv on experiment III.

7.3.3. Model framework

The framework for this experiment has as the main component the A-RQS model implemented following the architecture presented in Section 4.2.4.

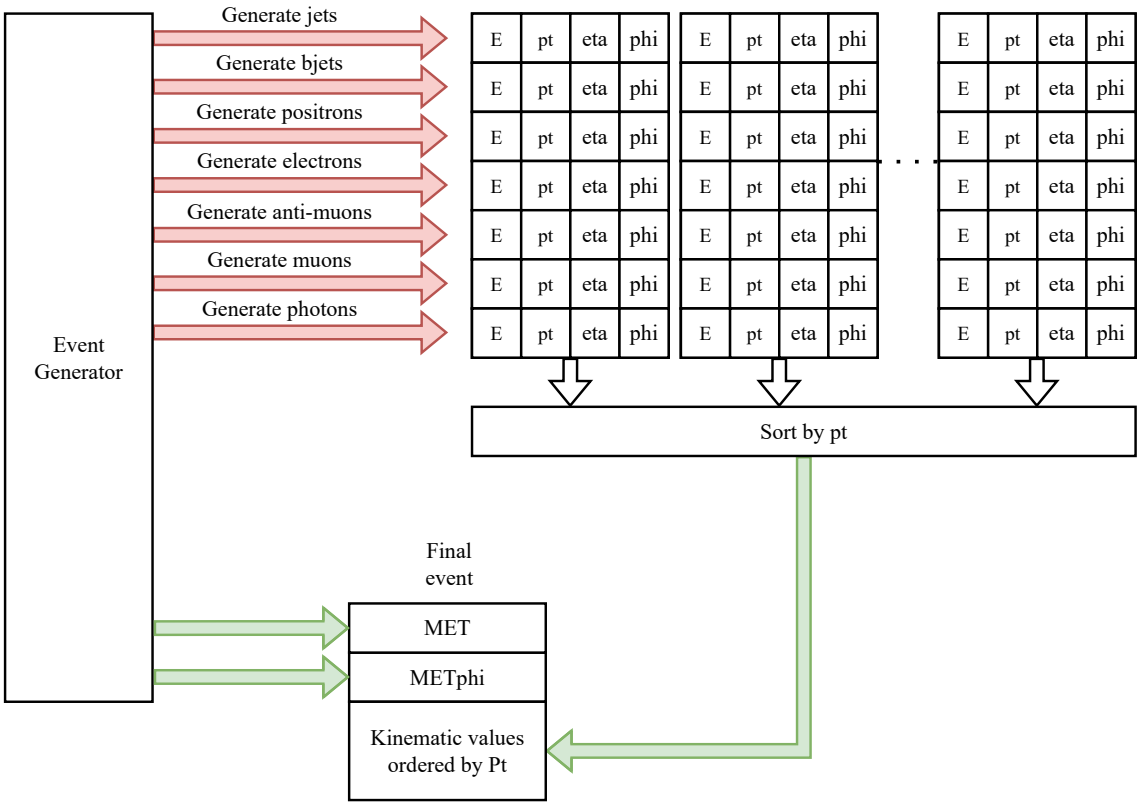


Figure 7.28: Experiment III model architecture.

As represented in the previous figure (Figure 7.28), this event generator framework only contains one major actor:

- **Event Generator Model**: An autoregressive rational quadratic spline flow model that has been trained on the distribution of the properties of an event as described in Section 7.3.2. The model's parameters can be found in Table 7.17 and Table 7.18.

Parameter	Values
Flow layers	10
Permutation layers	9

Table 7.17: Model hyperparameters on experiment III.

Parameter	Values
Hidden layers	3
Neurons per layer	512

Table 7.18: A-RQS hyperparameters on experiment III.

Unsurprisingly, the steps to generate events using this framework are straightforward:

Step 1: Sample from the event generator, which will provide us with all the values pertaining to an event.

Step 2: Discard the particles which we can interpret are not present on the event based on their low values, which approximate 0.

Step 3: Arrange the particles based on their transverse momentum (p_T) following the procedure outlined in Chapter 3. Proceed with constructing the event by assembling the sorted particles.

Step 4: Check for its validity requirements and save if it meets all of them.

7.3.4. Training results

The training of the model employed in this experiment was done using the parameters listed below (Table 7.19.

Parameter	Values
Optimizer	Adam[37]
Learning rate	10^{-4}
Loss function	Log-likelihood
Batch size	64

Table 7.19: Event Model training parameters on experiment III.

The model went over the training data about 100 times, that is, it was optimized through 100 **epochs**.

For this model, it is important to point out that the results were not optimal, but this was to be expected, as it is trying to learn a very unusual distribution. It tries to model, on the one hand the absence of a particle (meaning it's kinematic values are close to 0), and on the other the real distribution of its kinematic properties in case of particle presence.

This can be illustrated in the following plot of the first jet's energy distribution (Figure 7.29).

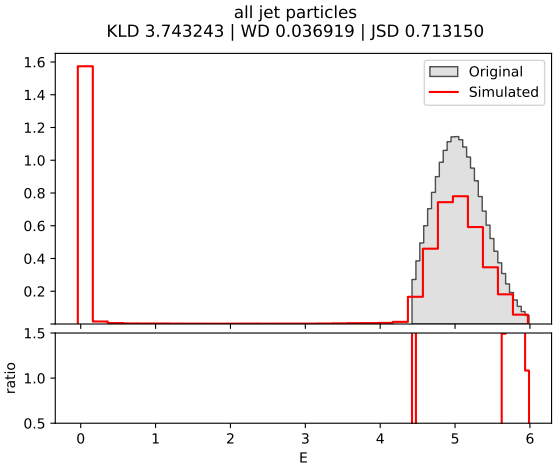


Figure 7.29: Histogram of the total learnt distribution for the energy of the jets on experiment III.

MET and METϕ

Even though MET and $MET\phi$ are not influenced by the effect mentioned before, it seems the model fails to learn their distribution well, while trying to follow their trend.

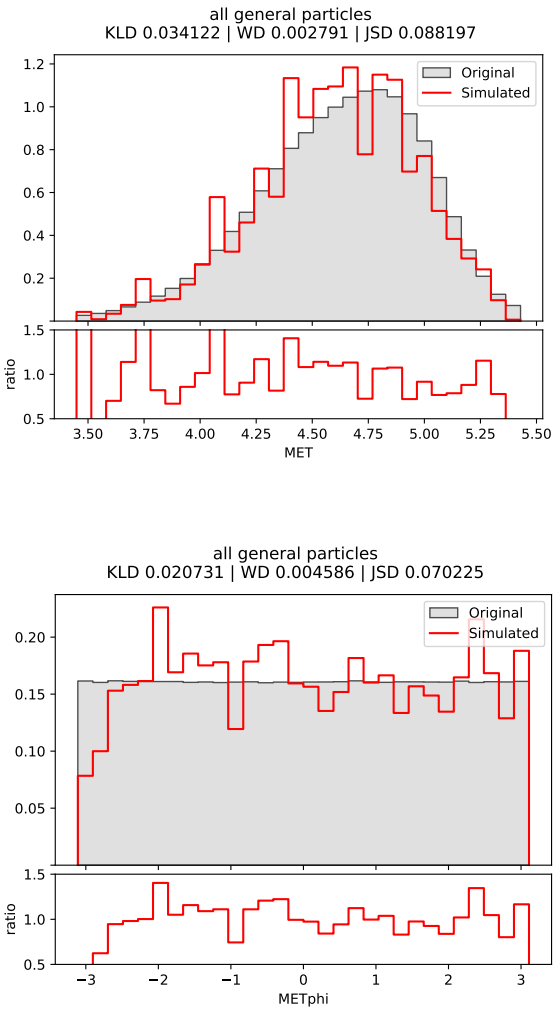


Figure 7.30: Histograms of MET and $MET\phi$ on experiment III.

Jets

The model does a better job at generating jets, having a harder time fitting the distribution of the pseudo-rapidity η and the azimuth angle ϕ.

The jets have been influenced by the "curse" defined in the beginning of the experiment, and even with that, the model proves capable of generating them accordingly.

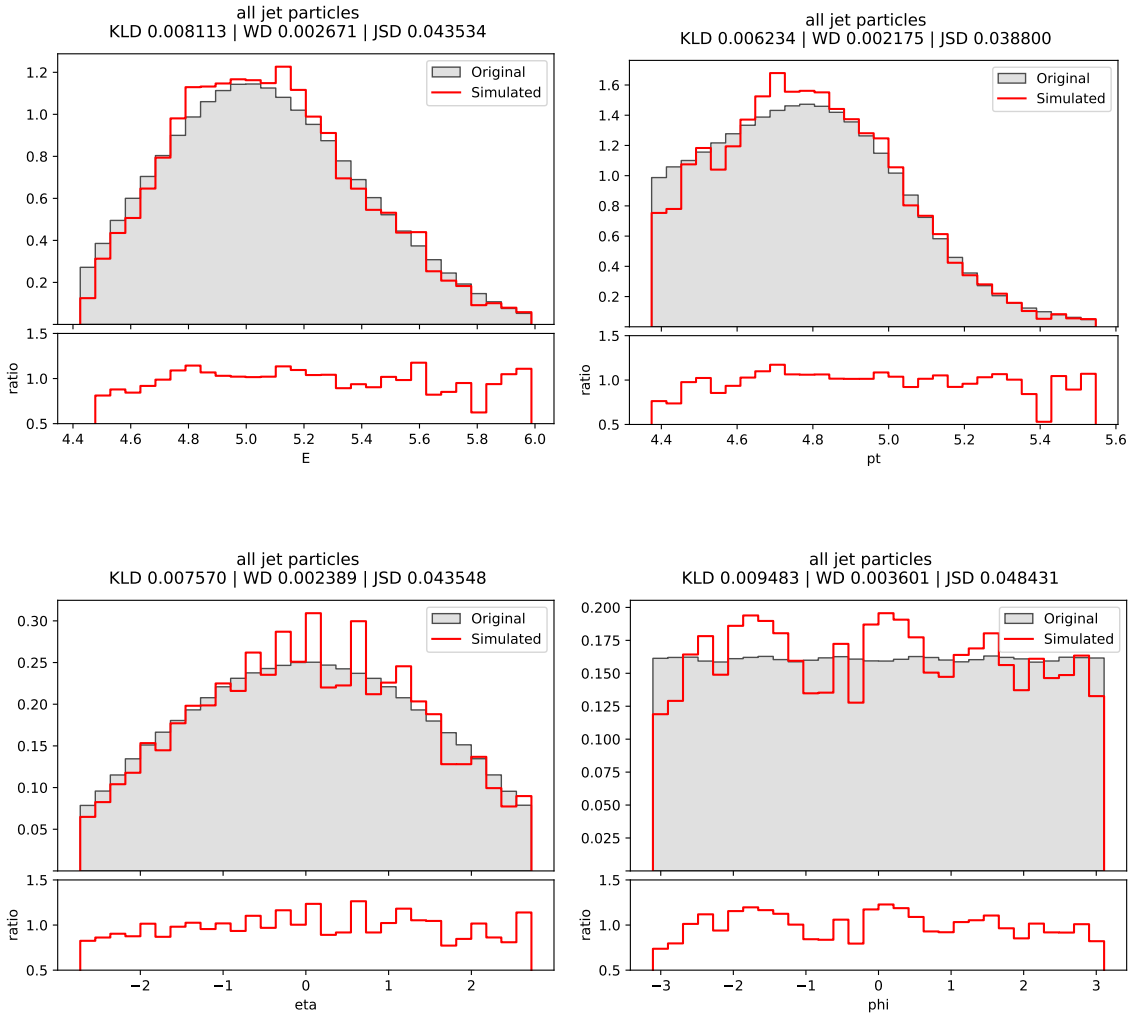


Figure 7.31: Histograms of the kinematic properties of all the jets (j, b) on experiment III.

Once again, as in experiment I and II, once we observe each consecutive jet by itself, the distributions can be seen diverging more and more.

The energy, depicted in Figure 7.32, is not as bad in the first three jets, but it doesn't do a great job in the next jets that appear on the events.

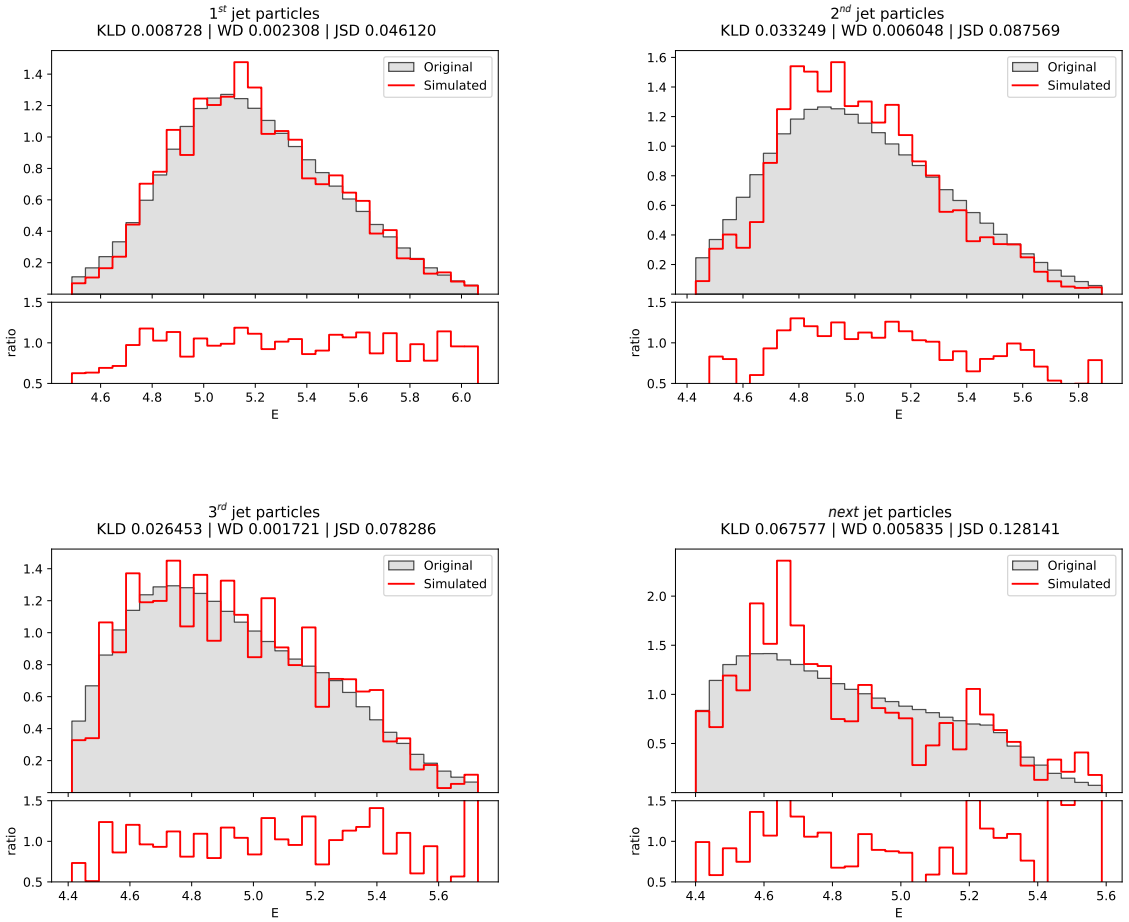
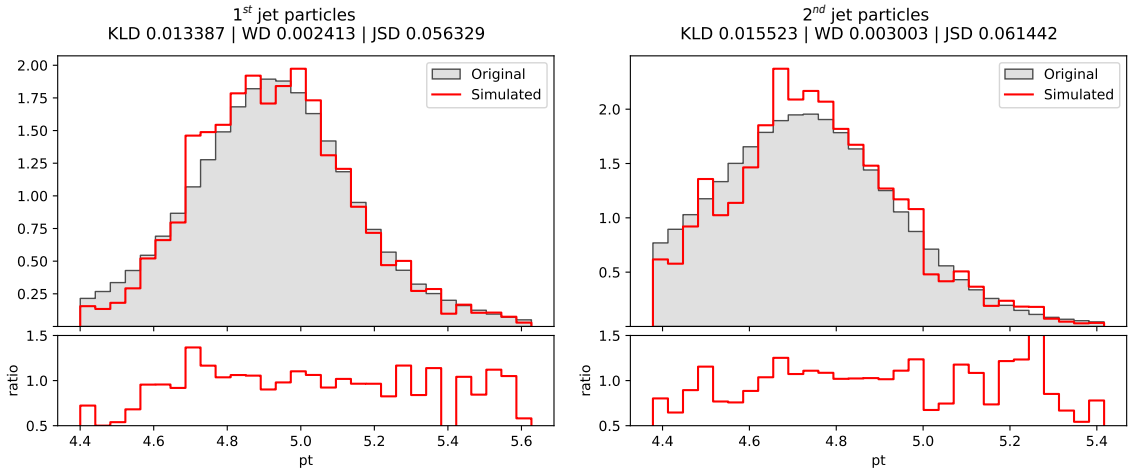


Figure 7.32: Histograms of the energy of the ordered jets (j, b) on experiment III.

The same thing we can say about the p_T, represented in Figure 7.33. A similar divergence than the one observed in the energy above is notable here.



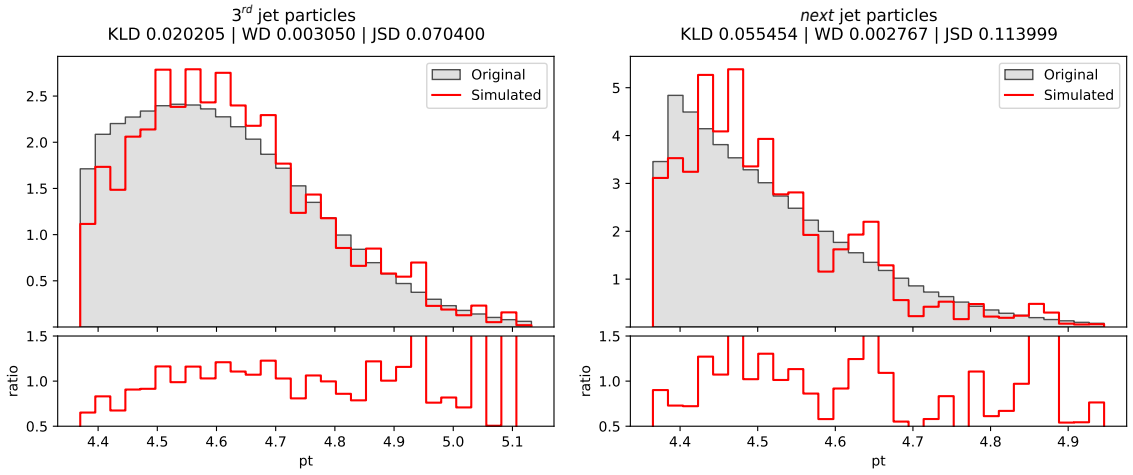


Figure 7.33: Histograms of the traversal momentum of the ordered jets (j, b) on experiment III.

In the case of the pseudo-rapidity η, even if the distribution is strange, it does not present a challenge for the model (Figure 7.34).

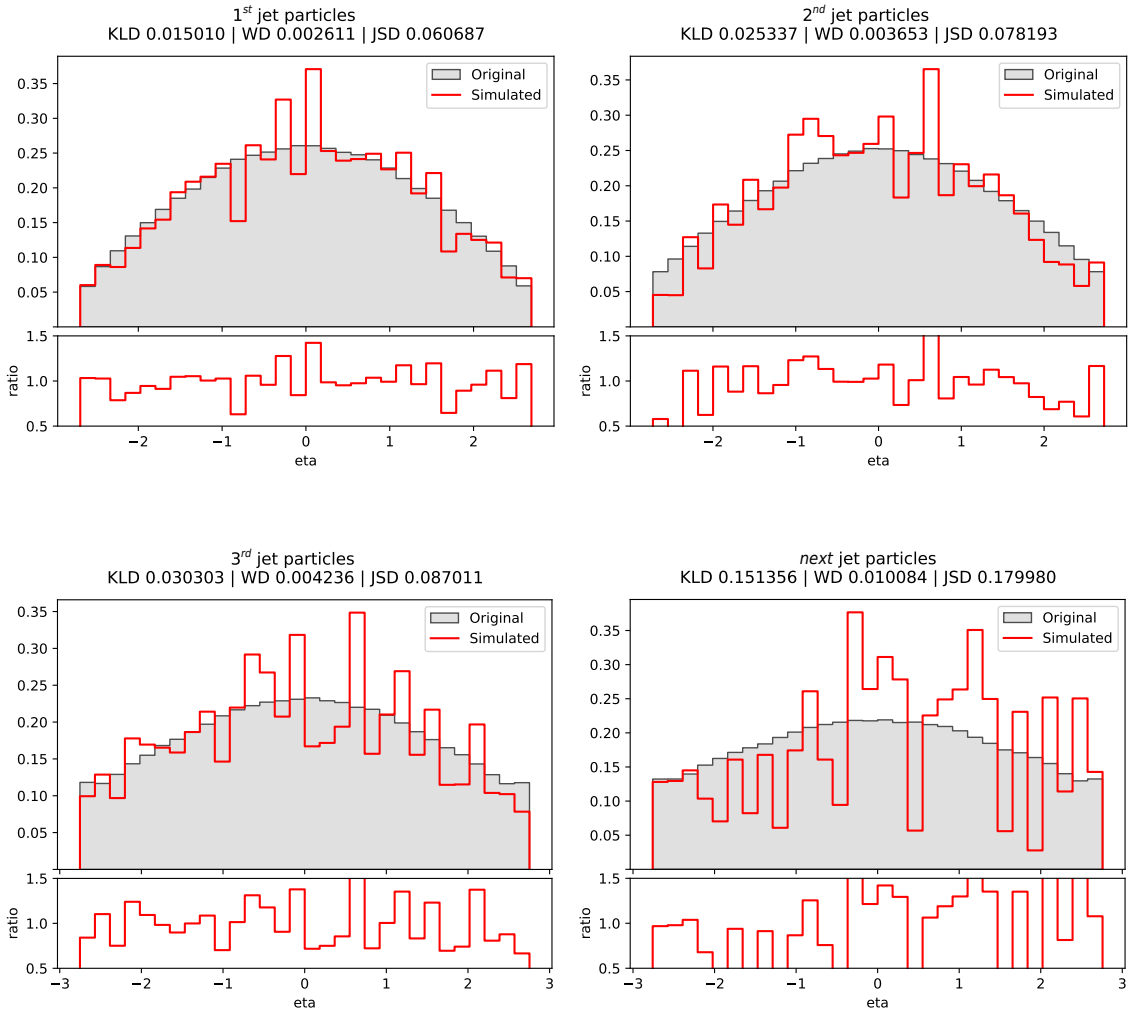


Figure 7.34: Histograms of the pseudo-rapidity of the ordered jets (j, b) on experiment III.

Finally, the azimuth angle ϕ, which presents itself as a uniform distribution (Figure 7.35), can also be easily simulated by the jets model.

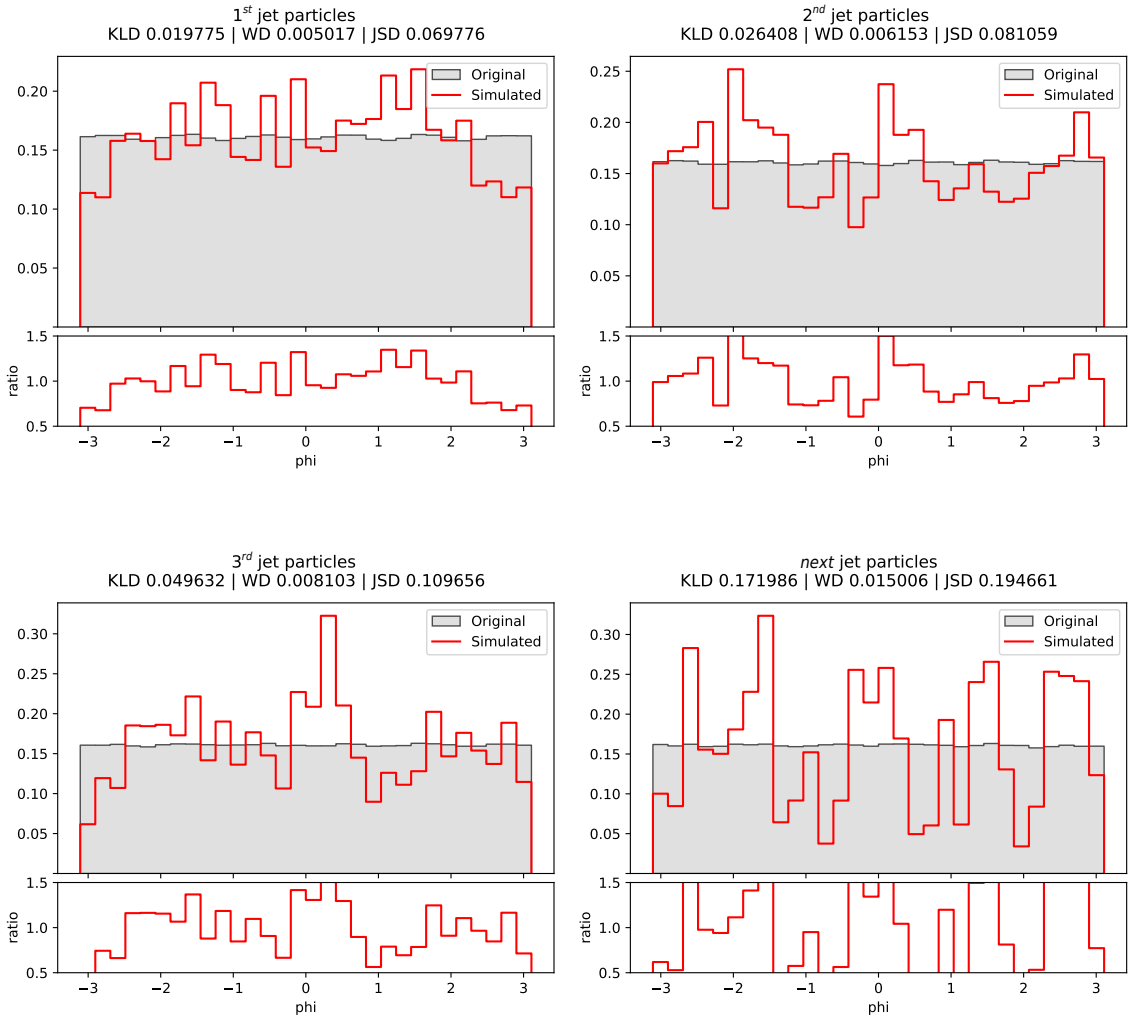
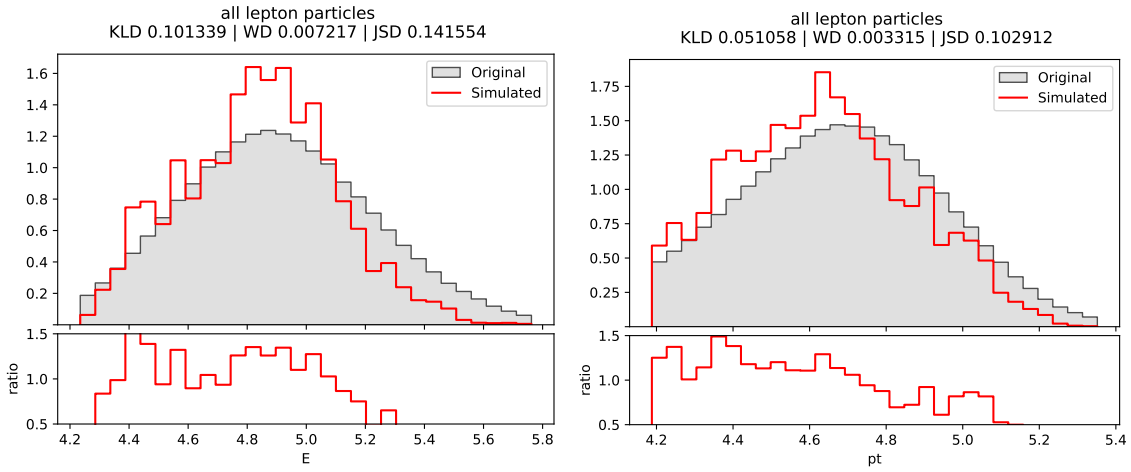


Figure 7.35: Histograms of the azimuth angle of the ordered jets (j, b) on experiment III.

Leptons

Leptons became a hurdle for the model (Figure 7.36). The lepton data produced by the model was not able to follow the original distribution of the kinematic properties.



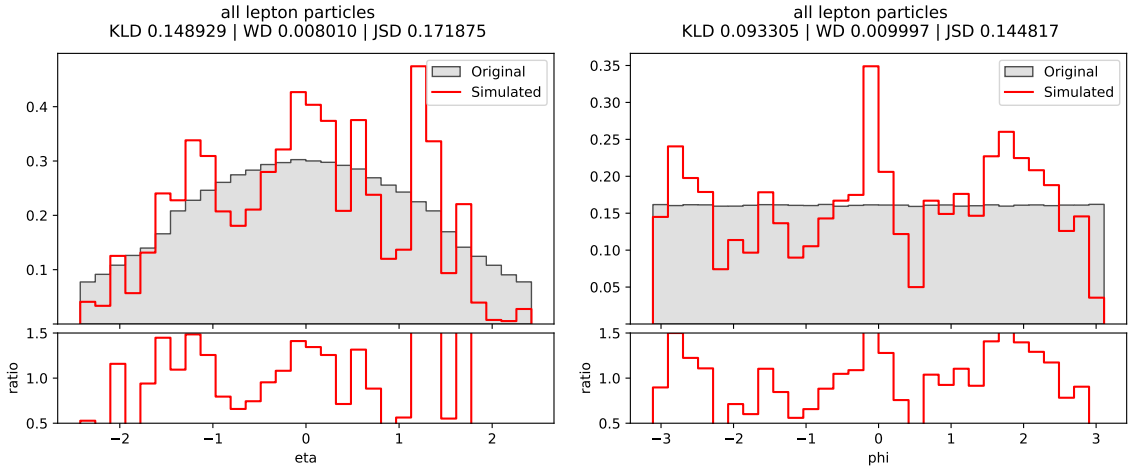


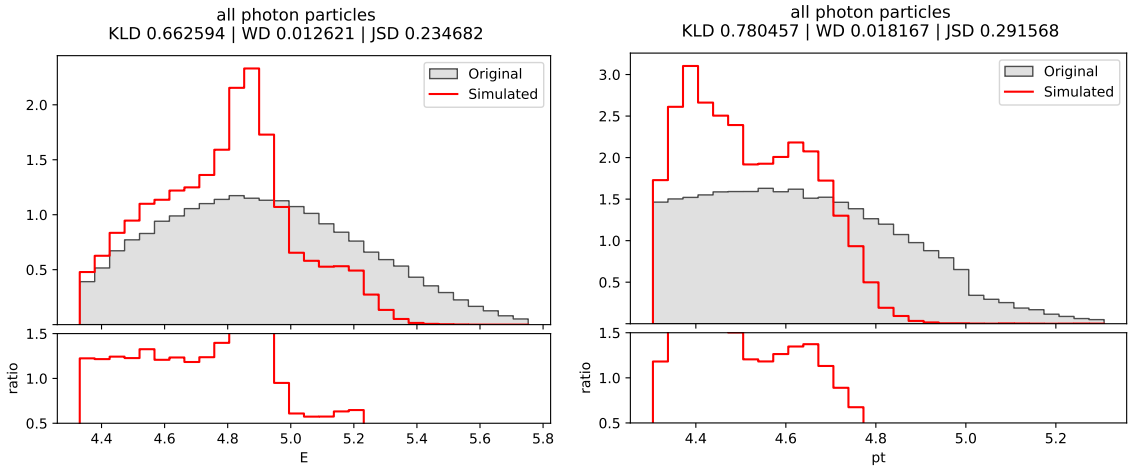
Figure 7.36: Histograms of the kinematic properties of all leptons (e^-, e^+, m^-, m^+) on experiment III.

Like in the jets, generated leptons' pseudo-rapidity η and azimuth angle ϕ did not fit the original data too closely.

Photons

The model fails to produce photons following the original distribution, as seen in Figure 7.37.

This can be attributed to the rareness of a photon appearing in an event, which makes it hard for the model to really capture the high end of the distributions.



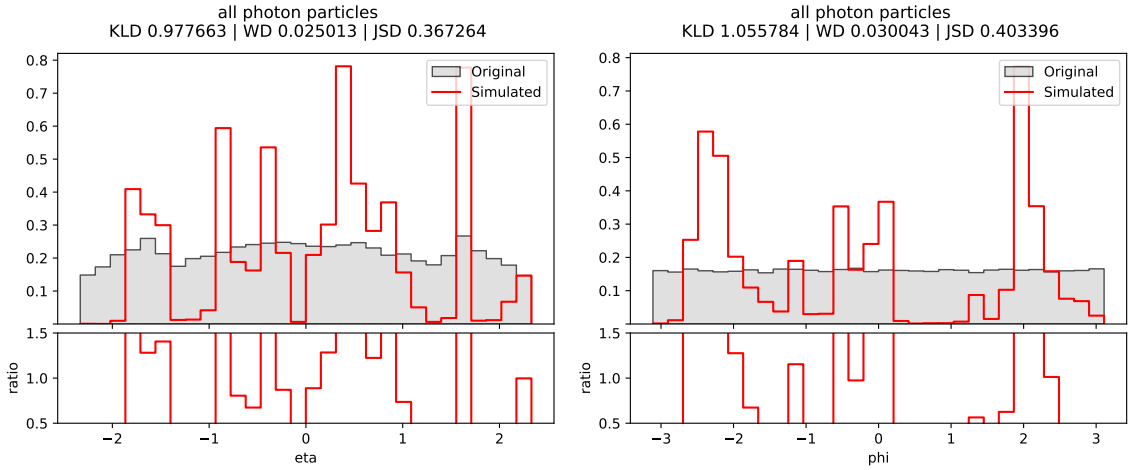


Figure 7.37: Histograms of the kinematic properties of all photons (g) on experiment III.

Conservation of energy and traversal momentum

Finally, the conservation of energy and traversal momentum, which we use as a possible metric to measure the production of plausible events, was satisfactory.

By this metric, the model is able to produce credible events, as can be seen in Figure 7.38. In this case, this experiment outperformed the preceding two by a large margin.

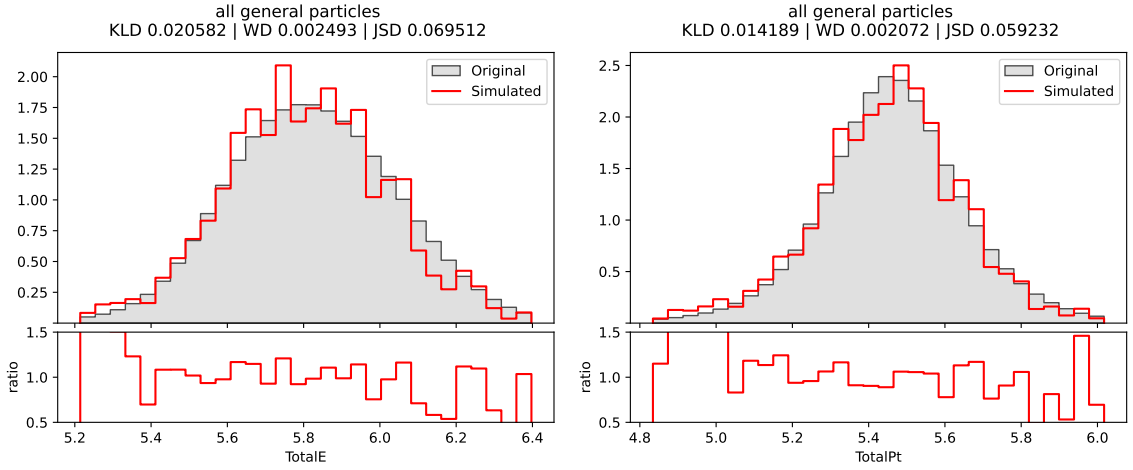


Figure 7.38: Histograms of the total energy and the total momentum in experiment III.

7.4 Model comparison

After conducting all the experiments, we decided to compare the models accuracy to replicate the *ttbar* dataset using the metrics presented in Chapter 6.

The values we see on Table 7.20 have been calculated using a pondered average. For each experiment, we took into account the metrics' values on each particle distribution generated accordingly to the amount of particles that we had in the original dataset, giving more weight to the models which had more data to train on.

Model	W Divergence	KL Divergence	JS Distance	Events/s[4]
Experiment I	**0.00835558**	**0.00268654**	**0.06028002**	10,4
Experiment II	0.02181364	0.00543950	0.11161955	7,2
Experiment III	0.10907629	0.01388603	0.25341768	**583,9**

Table 7.20: Experiment metrics comparison.

As expected, experiments I and II provided better metrics' results than experiment III. This is due to the fact that the first two experiments had multiple models with low dimensionality data instead of a single model that had to learn a much larger dimensional set of samples.

That said, even though both experiment I and II scored better on the metrics side of things, they may produce implausible events as pointed out on their respective sections, and will need a filter to validate the produced data. This is less the case of the model trained on experiment III, which makes it a better candidate to test on the Beyond Standard Model Physics experiment that proceeds.

7.5 BSM Experiment

To conclude our experimentation and this study, we decided to select the model from the third experiment to test against the *Beyond the Standard Model* event dataset, outlined in Table 3.2. In particular, we selected $stop\_01$ as the process to focus on, since it's the one that has the most reasonable amount of data.

The data has followed the exact same partitioning and preprocessing described in Section 7.3.1 and Section 7.3.2 respectively, leaving us with 183,719 events as training data.

7.5.1. Training results

The model was trained using the same parameters as seen in experiment III on Section 7.3.4, with 100 total passes through the data. We list them here again for reproducibility:

Parameter	Values
Optimizer	Adam[37]
Learning rate	10^{-4}
Loss function	Log-likelihood
Batch size	64

Table 7.21: Event Model training parameters on the BSM experiment.

[4]Calculated with 10^6 events generated using the setup described in 5.2.

Now, we will list the distributions of all the particles observed in the generation of the model compared with the original dataset, as we have done on the SM experiments. In this case, for simplicity and in order to avoid redundancy, we will only comment on the histograms which differ the most from the results on experiment III.

MET and METϕ

The results for the Missing Energy Traversal MET and of its angle $MET\phi$ (Figure7.39) are very similar to the results obtained in experiment III (Figure 7.30).

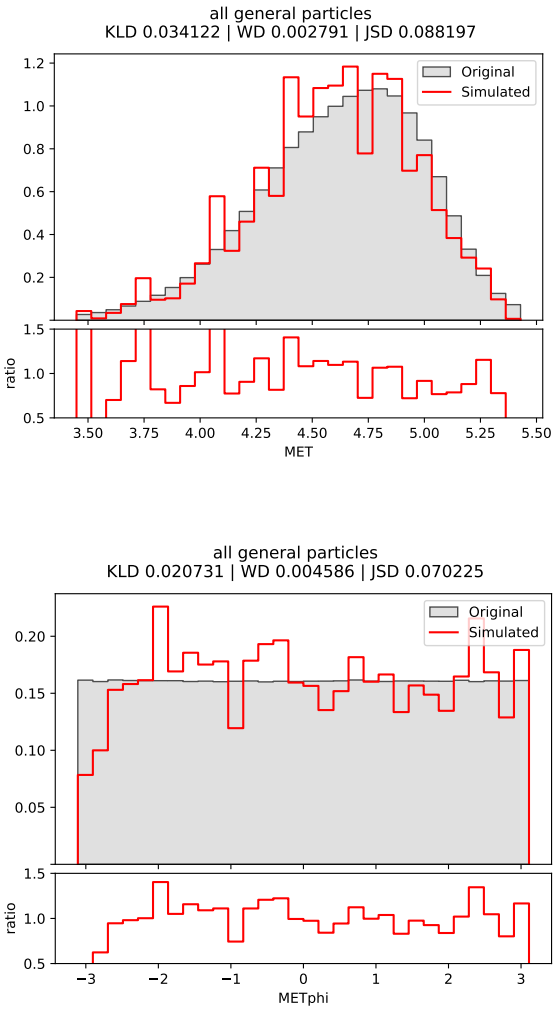


Figure 7.39: Histograms of MET and $MET\phi$ on the BSM experiment.

Jets

It again does a very similar performance to the observed in experiment III for the jets.

It is able to correcly assess the trend of the distributions, again having a tough time with the pseudo-rapidity η and the azimuth angle ϕ.

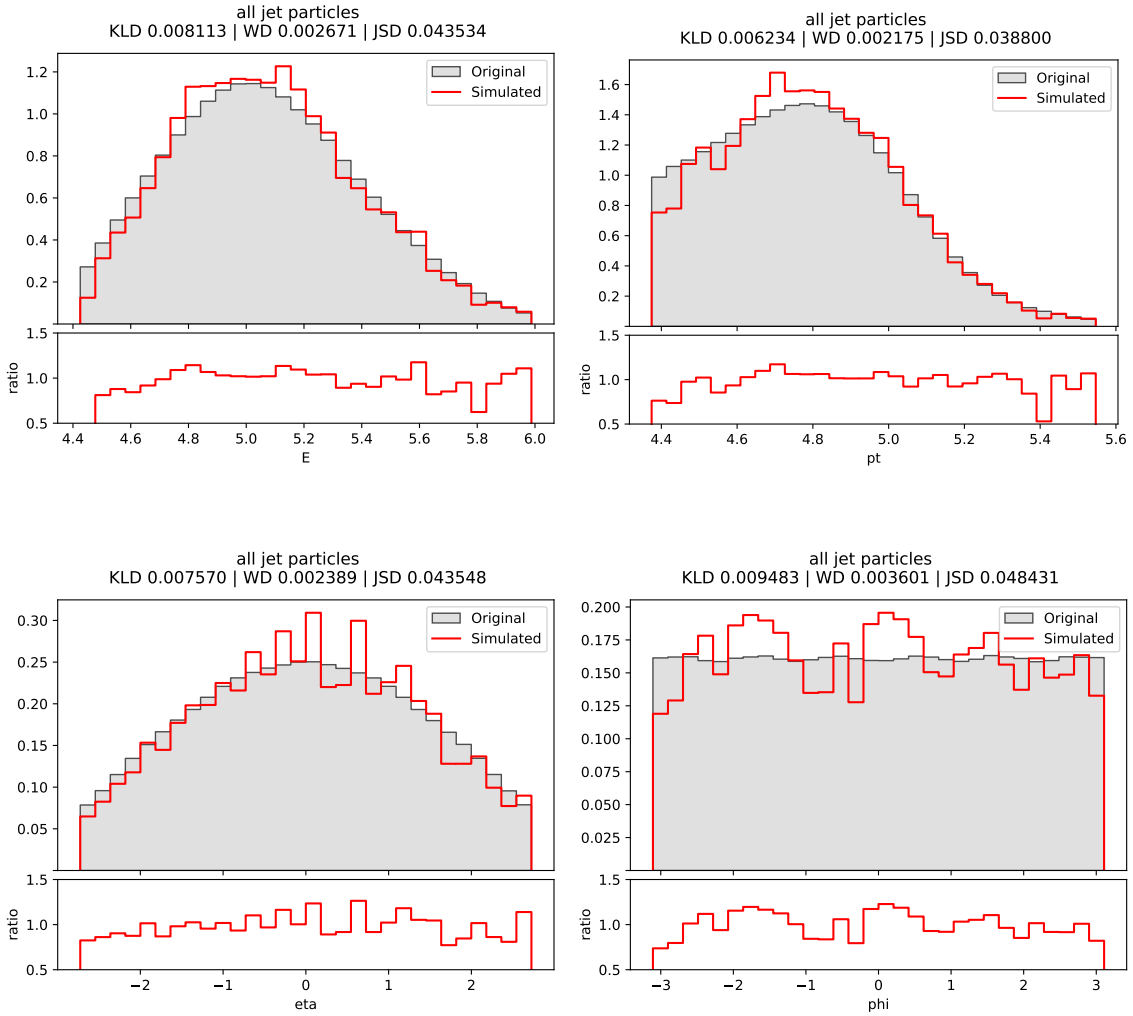
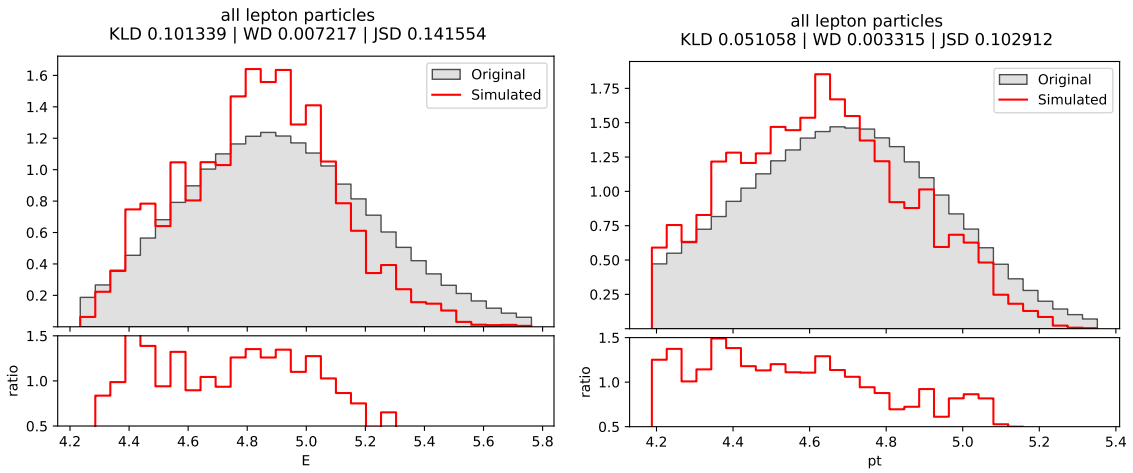


Figure 7.40: Histograms of the kinematic properties of all the jets (j, b) on the BSM experiment.

Leptons

Here, the model performs poorly (Figure 7.41), as Leptons became more of a challenge in this experiment, the reason being we have less samples to train from.



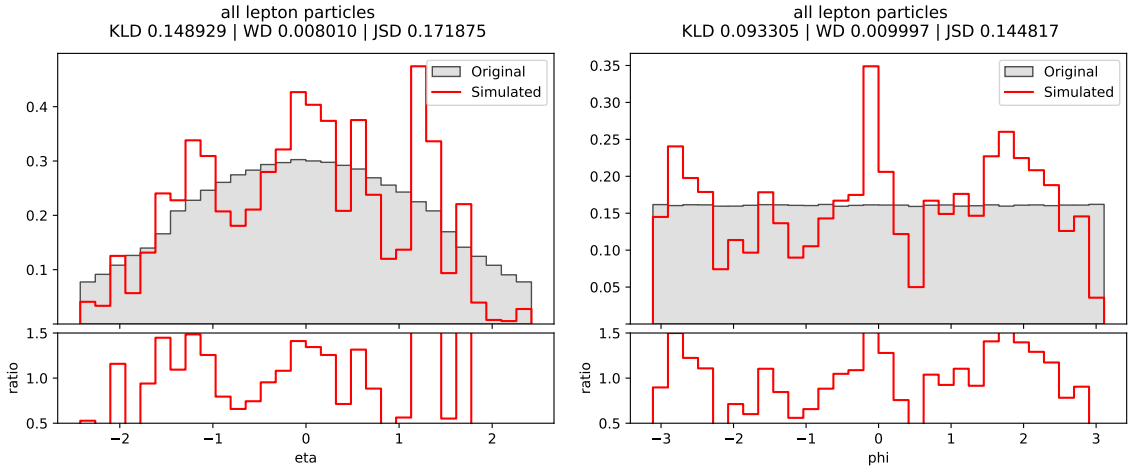


Figure 7.41: Histograms of the kinematic properties of all leptons (e^-, e^+, m^-, m^+) on the BSM experiment.

Photons

Not enough photons could be generated to obtain a representative distribution, since *stop_01* has a pretty limited amount of events, as pointed out in the beginning of this experiment.

Conservation of energy and traversal momentum

Similar results to experiment III were obtained in this regard, having less of a good fit due to the few samples the utilized dataset had (Figure 7.42).

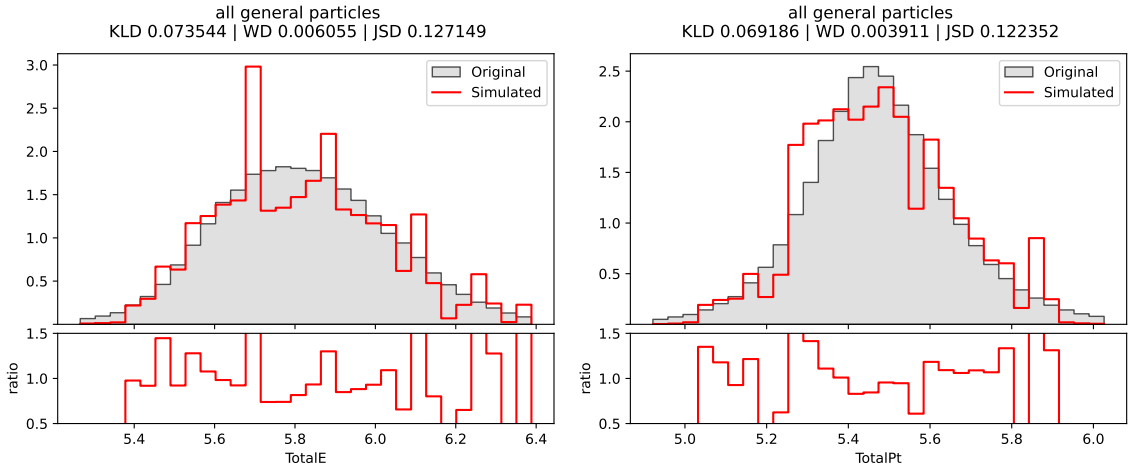


Figure 7.42: Histograms of the total energy and the total momentum on the BSM experiment.

CHAPTER 8
Conclusions

We acquired numerous intriguing outcomes in our work after creating and experimenting in several ways. Using methodologies diverse methodologies which provided different benefits and downsides, we generated events from the Standard Model and Beyond the Standard Model datasets more efficiently than typical Monte Carlo methods.

We employed different types of Autoregressive Normalizing Flows architectures such as MAF and A-RQS, paired with multiple frameworks, each focusing on a specific aspect of event generation. Even though the results weren't optimal in some cases, we believe a larger dataset and more complex architectures than the ones presented on this study could provide more accurate and usable results.

Taking into account the objectives listed in the beginning of this work, we feel like we have satisfied most of them, as we have provided a more flexible and faster way to produce events than current methods, while testing the performance of standalone normalizing flows in the context of event generation.

8.1 Future work

We propose further work that might be done starting from what we generated because the issue given here still has many opportunities to study, and because of its crucial importance in the future, when it will be required to artificially generate a large number of events.

As we have pointed out on experiment I and II, in order for these proposals to be useful for event generation, we should implement a discriminator or filter, which could select the plausible events and discard those that are impossible attending to the laws of nature.

Secondly, in order to improve the results of experiment III, which implemented a regular Autoregressive Normalizing Flow using A-RQS, we propose pairing this implementation using conditional normalizing flows (CNF), which could help producing better results for the particles that have less presence in the dataset.

Finally, exploring other Normalizing Flows such as GLOW, RealNVP, or Flow++ may be interesting to see how they behave in the context of generating events all at once, as in the last experiment.

Bibliography

[1] ATLAS Collaboration. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020. URL: https://doi.org/10.1016%2Fj.physletb.2012.08.020.

[2] Jory Sonneveld. *Searches for physics beyond the standard model at the LHC*. 2019. arXiv: 1905.06239 [hep-ex].

[3] Joshua S. Speagle. *A Conceptual Introduction to Markov Chain Monte Carlo Methods*. 2020. arXiv: 1909.12313 [stat.OT].

[4] Wolfgang Kilian, Thorsten Ohl, and Jürgen Reuter. "WHIZARD—simulating multi-particle processes at LHC and ILC". In: *The European Physical Journal C* 71.9 (Sept. 2011). DOI: 10.1140/epjc/s10052-011-1742-y. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-011-1742-y.

[5] Sydney Otten et al. *Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer*. 2021. arXiv: 1901.00875 [hep-ph].

[6] ATLAS Collaboration. "A strategy for a general search for new phenomena using data-derived signal regions and its application within the ATLAS experiment". In: *The European Physical Journal C* 79.2 (Feb. 2019). DOI: 10.1140/epjc/s10052-019-6540-y. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-019-6540-y.

[7] Thea Aarrestad et al. "The Dark Machines Anomaly Score Challenge: Benchmark Data and Model Independent Event Classification for the Large Hadron Collider". In: *SciPost Physics* 12.1 (Jan. 2022). DOI: 10.21468/scipostphys.12.1.043. URL: https://doi.org/10.21468%5C%2Fscipostphys.12.1.043.

[8] Christina Gao et al. "Event generation with normalizing flows". In: *Physical Review D* 101.7 (Apr. 2020). DOI: 10.1103/physrevd.101.076002. URL: https://doi.org/10.1103%5C%2Fphysrevd.101.076002.

[9] Fady Bishara and Marc Montull. *(Machine) Learning amplitudes for faster event generation*. 2020. arXiv: 1912.11055 [hep-ph].

[10] Simon Badger and Joseph Bullock. "Using neural networks for efficient evaluation of high multiplicity scattering amplitudes". In: *Journal of High Energy Physics* 2020.6 (June 2020). DOI: 10.1007/jhep06(2020)114. URL: https://doi.org/10.1007%5C%2Fjhep06%5C%282020%5C%29114.

[11] Bobak Hashemi et al. *LHC analysis-specific datasets with Generative Adversarial Networks*. 2019. arXiv: 1901.05282 [hep-ex].

[12] Riccardo Di Sipio et al. "DijetGAN: a Generative-Adversarial Network approach for the simulation of QCD dijet events at the LHC". In: *Journal of High Energy Physics* 2019.8 (Aug. 2019). DOI: 10.1007/jhep08(2019)110. URL: https://doi.org/10.1007%5C%2Fjhep08%5C%282019%5C%29110.

[13] Joshua Lin, Wahid Bhimji, and Benjamin Nachman. "Machine learning templates for QCD factorization in the search for physics beyond the standard model". In: *Journal of High Energy Physics* 2019.5 (May 2019). DOI: 10.1007/jhep05(2019)181. URL: https://doi.org/10.1007%5C%2Fjhep05%282019%29181.

[14] Raúl Balanzá García. "Use of Deep Learning generative models for Monte Carlo event simulation in the context of LHC experiments". PhD thesis. Universitat Politècnica de València, 2022.

[15] Pratik Jawahar et al. "Improving Variational Autoencoders for New Physics Detection at the LHC With Normalizing Flows". In: *Frontiers in Big Data* 5 (Feb. 2022). DOI: 10.3389/fdata.2022.803685. URL: https://doi.org/10.3389%5C%2Ffdata.2022.803685.

[16] DarkMachines community. *LHCsimulationProject*. Version v1. Zenodo, Feb. 2020. DOI: 10.5281/zenodo.3685861. URL: https://doi.org/10.5281/zenodo.3685861.

[17] T. Aarrestad et al. "The Dark Machines Anomaly Score Challenge: Benchmark Data and Model Independent Event Classification for the Large Hadron Collider". In: *SciPost Phys.* 12 (2022), p. 043. DOI: 10.21468/SciPostPhys.12.1.043. URL: https://scipost.org/10.21468/SciPostPhys.12.1.043.

[18] Alastair J. Walker. "New fast method for generating discrete random numbers with arbitrary frequency distributions". In: *Electronics Letters* 10 (1974), pp. 127–128.

[19] Alastair J. Walker. "An Efficient Method for Generating Discrete Random Variables with General Distributions". In: *ACM Trans. Math. Softw.* 3 (1977), pp. 253–256.

[20] "Wikimedia Commons". *"Alias Table"*. "File: Alias_Table.jpg". "2022". URL: https://commons.wikimedia.org/wiki/File:Alias_Table.png.

[21] Mathieu Germain et al. *MADE: Masked Autoencoder for Distribution Estimation*. 2015. arXiv: 1502.03509 [cs.LG].

[22] George Papamakarios, Theo Pavlakou, and Iain Murray. *Masked Autoregressive Flow for Density Estimation*. 2018. arXiv: 1705.07057 [stat.ML].

[23] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML].

[24] Andrea Coccaro et al. *On the curse of dimensionality for Normalizing Flows*. 2023. arXiv: 2302.12024 [stat.ML].

[25] Conor Durkan et al. *Neural Spline Flows*. 2019. arXiv: 1906.04032 [stat.ML].

[26] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[27] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[28] François Chollet et al. *Keras*. https://keras.io. 2015.

[29] Sharan Chetlur et al. *cuDNN: Efficient Primitives for Deep Learning*. 2014. arXiv: 1410.0759 [cs.NE].

[30] David Luebke. "CUDA: Scalable parallel programming for high-performance scientific computing". In: (2008), pp. 836–838. DOI: 10.1109/ISBI.2008.4541126.

[31] Assyareefah Saad. "Automated Pterygium Detection in Anterior Segment Photographed Images using Deep Convolutional Neural Network". In: *International Journal of Advanced Trends in Computer Science and Engineering* 8 (Dec. 2019), pp. 225–232. DOI: 10.30534/ijatcse/2019/3481.62019.

[32] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[33] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[34] Yossi Rubner, Carlo Tomasi, and Leonidas Guibas. "The Earth Mover's Distance as a metric for image retrieval". In: *International Journal of Computer Vision* 40 (Jan. 2000), pp. 99–121.

[35] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. ISSN: 00034851. URL: http://www.jstor.org/stable/2236703 (visited on 06/28/2023).

[36] J. Lin. "Divergence measures based on the Shannon entropy". In: *IEEE Transactions on Information Theory* 37.1 (1991), pp. 145–151. DOI: 10.1109/18.61115.

[37] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

APPENDIX A
Sustainable Development Goals

Degree of relevance of the work to the Sustainable Development Goals (SDGs).

Sustainable Development Goals	High	Medium	Low	Unrelated
SDG 1. **No poverty.**				X
SDG 2. **Zero hunger.**				X
SDG 3. **Good health and well-being.**				X
SDG 4. **Quality education.**				X
SDG 5. **Gender equality.**				X
SDG 6. **Clean water and sanitation.**				X
SDG 7. **Affordable and clean energy.**		X		
SDG 8. **Decent work and economic growth.**				X
SDG 9. **Industry, innovation, and infrastructure.**		X		
SDG 10. **Reduced inequalities.**				X
SDG 11. **Sustainable cities and communities.**				X
SDG 12. **Responsible consumption and production.**				X
SDG 13. **Climate action.**			X	
SDG 14. **Life below water.**				X
SDG 15. **Life on land.**				X
SDG 16. **Peace, justice, and strong institutions.**				X
SDG 17. **Partnerships for the goals.**			X	

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

ETS Enginyeria Informàtica
Camí de Vera, s/n, 46022, València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es

Euro-Inf Bachelor awarded by EQANIE

Discussion on the relationship of this work with the selected SDGs

This study is important to multiple Sustainable Development Goals (SDGs), more concretely to 7, 9, 13, and 17. By applying deep learning-based generative models for event generation, we can help promote sustainable energy use, boost innovation, support climate action, and establish relationships in the high-energy physics research community

The project's significance to **Affordable and Clean Energy** (SDG 7) stems from its ability to lower the computational resources and energy consumption required for particle collision simulation. This has the potential to promote more sustainable energy consumption in high-energy physics research and lead to improvements in energy-efficient devices. The project's emphasis on enhancing event generating efficiency coincides with SDG 7 Target 7.3, which aims to double the worldwide rate of progress in energy efficiency by 2030.

In terms of **Industry, Innovation, and Infrastructure** (SDG 9), the experimentation's approach to high-energy physics modeling can help to develop robust infrastructure and promote sustainable industry. The use of generative models for event generation in LHC experiments coincides with SDG 9 Target 9.5, which strives to improve scientific research and industrial sector technological capacities. The project's emphasis on improving event generation efficiency and accuracy can also contribute to Target 9.4, which calls for upgrading infrastructure to make them more sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies.

Although the project's direct impact on **Climate Action** (SDG 13) may not be obvious at first, its potential contributions to energy efficiency and sustainable energy use can help climate action efforts indirectly. The project can assist decrease greenhouse gas emissions linked with high-energy physics research by lowering the energy consumption required for modeling particle collisions, which is consistent with SDG 13 Target 13.2, that asks for the incorporation of climate change measures into national policies, strategies, and planning.

The study's significance to **Partnerships for the Goals** (SDG 17) stems from its ability to develop collaborations amongst physics researchers, institutions, and others. These collaborations can promote the exchange of knowledge, experience, and resources, hence boosting the creation and implementation of more efficient and sustainable event generation approaches. The project's development and publication coincides with SDG 17 Target 17.6, which asks for increased cooperation and access to science, technology, and innovation, as well as increased knowledge-sharing on mutually agreed-upon terms.

We can then say that the initiative of this project is devoted to the SDGs set by the United Nations in the 2030 Agenda for Sustainable Development, as we have explained. These objectives symbolize a fantastic opportunity for all of us to embark on a new path that will allow us to improve our quality of life while remaining mindful of the impact of any project as it is implemented.