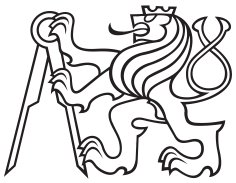
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

# Generative Models for High Energy Physics Measurements

**Bc. Lukáš Viceník**

Supervisor: doc. Boris Flach, Dr. rer. nat. habil.
Second supervisor: doc. Dr. André Sopczak
Study program: Cybernetics and Robotics
May 2024

# MASTER‘S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Viceník Lukáš** Personal ID number: **483424**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

## II. Master’s thesis details

Master’s thesis title in English:

**Generative Models for High Energy Physics Measurements**

Master’s thesis title in Czech:

**Generativní modely pro měření ve fyzice vysokých energií**

Guidelines:

The simulation of detector responses for signal and background events is an important tool for experimental High Energy Physics research. In many cases, the complexity of these simulators does not allow the creation of sufficiently large datasets for training advanced deep networks. The aim of this thesis is to design and learn latent variable models for generating realistic detector responses that can be used to augment (enlarge) training data e.g. for deep network event/background classifiers. The particular tasks for the diploma thesis are:
1. Familiarise with the current state of hierarchical variational autoencoders (VAE) and diffusion models as well as with their learning approaches.
2. Design a hierarchical VAE model suitable for generating mixed type feature vectors with real valued/categorical components as obtained from simulators of the CERN ATLAS detector.
3. Learn the model by the standard ELBO approach as well as by the recently proposed symmetric equilibrium learning for VAEs.
4. Validate the generative capabilities of the learned models w.r.t. suitable distribution similarity criteria and w.r.t. physical relationships between feature components.
5. Choose a deep network classifier for discriminating signal classes and background events. Learn it on data generated by the simulation of detector responses. Augment (enlarge) the training set by data generated form the obtained hierarchical VAE model. Compare the classifiers w.r.t. their sensitivity and achieved classification accuracies. Evaluate if the augmented data set increases the sensitivity including systematic uncertainties.

Bibliography / sources:

[1] Kingma, Diederik P. and Welling, Max, An Introduction to Variational Autoencoders, Foundations and Trends in Machine Learning: Vol. 12 (2019), eprint arXiv:1906.02691
[2] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. NeurIPS 2016.
[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. NeurIPS 2020.
[4] Boris Flach, Dmitrij Schlesinger, Alexander Shekhovtsov, Symmetric Equilibrium Learning of VAEs, 2023, eprint arXiv:2307.09883

Name and workplace of master's thesis supervisor:

**doc. Boris Flach, Dr. rer. nat. habil. Machine Learning FEE**

Name and workplace of second master's thesis supervisor or consultant:

**doc. Dr. André Sopczak High Energy Physics, IEAP CTU in Prague**

Date of master's thesis assignment: **19.01.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

| doc. Boris Flach, Dr. rer. nat. habil.<br>Supervisor's signature | prof. Dr. Ing. Jan Kybic<br>Head of department's signature | prof. Mgr. Petr Páta, Ph.D.<br>Dean's signature |
|---|---|---|

## III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

| . | |
|---|---|
| Date of assignment receipt | Student's signature |

# Acknowledgements

Thank you, to my supervisors, doc. Boris Flach and doc. André Sopczak, for their invaluable time, guidance, and unwavering support throughout this thrilling and unpredictable journey. I am also deeply thankful to my family for their constant encouragement and support.


# Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 24, 2024

Lukáš Viceník

# Abstract

In this thesis, we augmented Monte-Carlo simulated data for charged Higgs boson searches. Events are selected if they have two light leptons (electron or muon) of the same sign and exactly one hadronically decaying $\tau$-lepton. For the data generation, a variational autoencoder model was used with evidence lower bound and symmetric equilibrium learning. Both mentioned learning approaches were also tested with hierarchical (Ladder) architecture. For the data quality assessment, both qualitative and quantitative metrics were taken into account. The standard evidence lower bound (ELBO) learning model was selected as the best-performing option. The model was then used to generate data for the signal and background separation analysis experiments. The dependence of classifier performance on the training dataset size was demonstrated using two widely used machine learning paradigms for tabular data classification: gradient-boosted decision trees and deep neural networks.



**Keywords:** CERN, ATLAS, Particle physics, Higgs Boson, Data analysis, ROOT, RDataFrame, Significance, Deep learning, Generative AI, Variational autoencoders, Hierarchical VAE, Ladder VAE, Evidence lower bound, Symmetric Equilibrium Learning, Multilayer perceptron, XGBoost, t-SNE, $\chi^2$ test, Wasserstein Distance

**Supervisor:** doc. Boris Flach, Dr. rer. nat. habil.
**Second supervisor:** doc. Dr. André Sopczak



# Abstrakt

V této práci jsme augmentovali Monte-Carlo simulovaná data pro výzkum nabitého Higgsova bosonu. Jednotlivé eventy jsou vybrány, pokud obsahují dva lehké leptony (elektron, mion) stejného náboje a přesně jeden hadronicky se rozpadající tauon. Pro augmentaci dat byl zvolen jako model variační autoenkodér s evidence lower bound a symmetric equilibrium algoritmem učení. Obě metody učení byly také otestovány s hierarchickou (Ladder) architekturou. K posouzení kvality dat byly voleny jak kvantitativní, tak kvalitatnivní metriky. Standardní evidence lower bound model byl zvolen jako nejlepší na základě nejlepších dosažených výsledků. Model byl dále zvolen pro augmentaci dat určených pro experiment zaměřený na separaci signálu a pozadí. Závislost výkonu klasifikátoru na velikosti trénovacího datasetu byla demonstrována za použití dvou obecně známých paradigmat strojového učení pro tabulková data: gradient-boosted decision trees a hlubokých neuronových sítí.



**Klíčová slova:** CERN, ATLAS, Částicová fyzika, Higgsův boson, Analýza dat, ROOT, RDataFrame, Signifikance, Hluboké strojové učení, Generativní UI, Variační autoenkodéry, Hierarchické VAE, Ladder VAE, Evidence lower bound, Symmetric Equilibrium Learning, Multilayer perceptron, XGBoost, t-SNE, $\chi^2$ test, Wassersteinova vzdálenost

**Překlad názvu:** Generativní modely pro měření ve fyzice vysokých energií

# Contents

## Appendices

# Figures

# Tables

# Chapter 1

# Introduction

Nowadays, analyzing high-energy physics collider data is typically done by machine learning approaches. These approaches used mainly for signal and background separation require neural networks with complex architectures and therefore large training datasets. The datasets are generated by Monte Carlo (MC) simulations that are computationally costly and can run for a very long time until the data sets are produced. Moreover, the additional preselection significantly reduces the dataset size. To overcome these challenges, machine learning-based methods started to blossom even in the data generation domain.

Data augmentation can be addressed by a plethora of machine learning models applied on different levels of analysis. The subject of this thesis is the development of a generative model based on the variational autoencoder paradigm. The idea is to learn the underlying distribution from the already simulated data, thus the MC simulation process is not involved. As a result, a dataset of an arbitrary size can be generated from the learned distribution in a very small fraction of MC simulation time.

The quality of the generated data is then verified with a variety of evaluation metrics. After the evaluation, the best model is chosen to generate data for data separation analysis. Results are then compared with the MC-generated dataset.

# Part I

# Theory

# Chapter 2

# Physics

This chapter provides the definitions of basic physical quantities and the process of data acquisition in the ATLAS detector. The chapter is enclosed with the outline of the data transformation process preceding the analysis.

## 2.1 Definitions

Definitions in this section provide explanations necessary for the proper understanding of the ATLAS detector functionality and the data acquisition.

### 2.1.1 Cross-section

In general, cross-section in particle and nuclear physics means the probability that two particles approaching each other will interact with each other. From the geometrical point of view, it can be understood as the area within which a reaction will take place. Units of the cross-section are therefore units of an area called bars defined as

$$1 \text{ barn} = 10^{-24}\text{cm}^2. \tag{2.1}$$

which is about the size of a uranium nucleus. The cross-section values for important reactions are shown in Figure 2.1. For the W and Z boson, the

cross-section values are very high, therefore the interactions occur very often compared to the Higgs boson. [1]

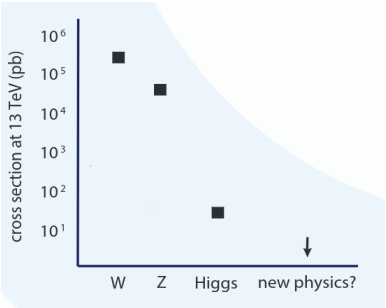



**Figure 2.1:** Cross-section values for important production processes.

### 2.1.2 Luminosity

The quantity that measures the ability of a particle accelerator to produce the number of required events is called luminosity. Its relation to the production cross-section reads as follows:

$$\frac{dR}{dt} = \mathcal{L} \cdot \sigma_p, \tag{2.2}$$

where on the left-hand side is a number of events per second and on the right-hand side is a product of luminosity and the production cross-section. The units are then consequently $\mathrm{cm}^{-2}\mathrm{s}^{-1}$.
In practice, however, what we are usually given is the integrated luminosity because it directly relates to a number of observed events:

$$\mathcal{L}_{int} = \int_0^T \mathcal{L}(t)dt \quad \longrightarrow \quad R = \mathcal{L}_{int} \cdot \sigma_p. \tag{2.3}$$

For detailed derivation, [2] can be consulted.

### 2.1.3 Pseudorapidity

Before we approach pseudorapidity, let us focus on its parental quantity called rapidity. To explain its definition we need to define energy in a relativistic way

$$E^2 = p_x^2c^2 + p_y^2c^2 + p_z^2c^2 + M^2c^4, \tag{2.4}$$

where $\boldsymbol{p}$ is a momentum, c is the speed of light and M is the rest mass of the particle. The rapidity is then defined as

$$y = \frac{1}{2} \ln \left( \frac{E + p_z c}{E - p_z c} \right). \tag{2.5}$$

To appreciate the usefulness of such a quantity we need to understand that particles produced after the collision are almost always directed in a plane perpendicular to the beam direction, for these particles we obtain $\ln 1$ meaning $y = 0$. On the contrary, for particles aligned with the beam axis $y \to \pm\infty$. Unfortunately, in practice, it can be very difficult to measure rapidity for highly relativistic particles that have very high momenta. For this reason, we define a new concept that leads to very similar results

$$\eta = -\ln \tan \frac{\theta}{2}, \tag{2.6}$$

which we call pseudorapidity. The $\theta$ parameter is an angle made by particle trajectory with the beam pipe, meaning $\cos\theta = p_z/p$. For highly relativistic particles $y \simeq \eta$ which is exactly the case of LHC. The full derivation of 2.4 and 2.6 can be found in [3]. Figure 2.2 shows the pseudorapidity denoted by $P$ in the Large Hadron Collider coordinate frame.

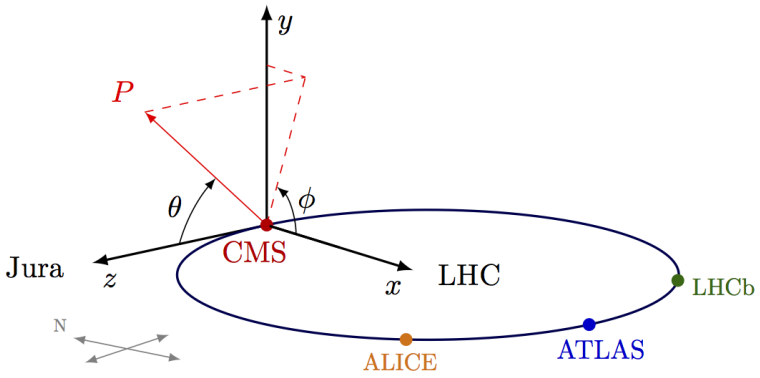



**Figure 2.2:** The pseudorapidity $P$ measured in LHC coordinate frame with origin in CMS detector [4].

## 2.2 CERN

Conseil européen pour la Recherche nucléaire known as CERN is international scientific organisation for the purpose of collaborative research into high-energy particle physics. It was founded in 1954 and its headquarters placed near Geneva. Research achievements encompass a wide range of achievements from Nobel prize-winning scientific discoveries to the invention of the World Wide Web [5].

The accelerator complex at CERN is a succession of machines that accelerate particles to increasingly higher energies. Each machine boosts the energy of a beam of particles before injecting it into the next machine in the sequence. The whole accelerator complex is shown in Figure 2.3. In the final stage, the

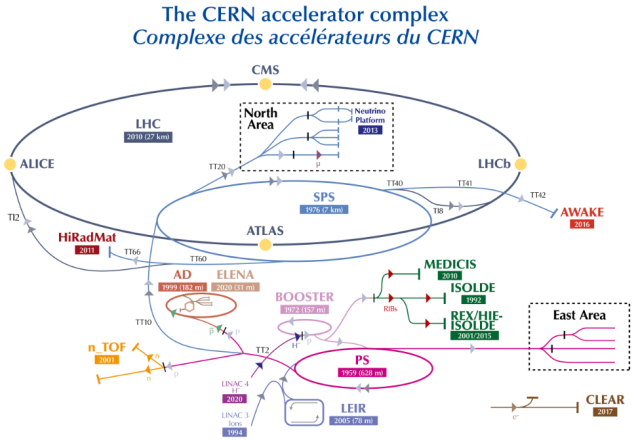



**Figure 2.3:** Schematic depiction of the CERN accelerator complex.

protons are transferred to the two beam pipes of the Large Hadron Collider. After 20 minutes protons reach their maximal energy 6.5 TeV. Beams circulate for many hours inside the LHC beam pipes under normal operating conditions. The two beams are brought into collision inside four detectors called ATLAS, CMS, ALICE, and LHCb where the total energy at the collision point reaches energy 13 TeV [6].

## 2.3 ATLAS detector

ATLAS is a general-purpose detector built in purpose to answer questions underlying the Standard Model of particles and its extensions. Its main goal is thus to search for new particles or increase the precision of already achieved discoveries. To perform the collision three types of constituent pairs can be chosen. The first one is proton-proton (pp) collision with peak luminosity $\mathcal{L} = 10^{34}\text{cm}^{-2}\text{s}^{-1}$ and center of mass energy $\sqrt{s} = 13$ TeV (one proton has energy 6.5 TeV). Another two collisions contain lead-lead (Pb-Pb) and proton-lead (p-Pb) constituents reaching peak luminosity $\mathcal{L} = 10^{27}\text{cm}^{-2}\text{s}^{-1}$ and center of mass energy $\sqrt{s} = 5.5$ TeV.

The detector is the dimension of a cylinder that is 46 meters long and its width is 25 meters. These measures make it the largest detector in the world. Its structure covers almost the entire solid angle around its collision

point and consists of several layers. The inner-most part is called the inner tracking detector (ID) which serves as the primary tracking device where the position and momentum of charged particles are measured with excellent resolution. It covers measurements for both primary and secondary vertex within the pseudorapidity range $|\eta| < 2.5$ (section 2.1.3). The whole device is enveloped by a superconducting solenoidal magnet with 2 T field. The ID itself consists of three sub-parts: the high-resolution Pixel detector which is located just 3.3 cm from the beamline and measures with precision just 10 $\mu$m to obtain the position and momentum of a particle. The second part is a semiconductor tracker used to reconstruct tracks of charged particles coming from the collision. The last part is the Transition Radiation Tracker consisting of thin tubes filled with a gas mixture. As the particles cross the gas they ionize it and create a detectable electric signal used to reconstruct their tracks [7].

Then follows electromagnetic, and hadronic calorimeters that measure deposited energy lost by particles passing through the detector. They are designed to stop a majority of particles except for muons and neutrinos. An electromagnetic calorimeter measures the energy of electrons and photons. A hadronic calorimeter quantifies the energy of hadrons as their atomic nuclei interact with the detector. The whole calorimetry system covers $|\eta| < 4.9$. The electromagnetic calorimeter consists of a barrel and endcap part that operates within $|\eta| < 3.2$ with the use of lead/Liquid Argon (LAr) and is preprocessed with presampler covering $|\eta| < 1.8$ for corrections. Hadron calorimetry makes use of materials like steel/scintillator-tile for three barrel segments within $|\eta| < 1.7$ and copper/LAr for an endcap calorimeter. The coverage is finalized with a forward copper/LAr calorimeter for electromagnetic and a tungsten/LAr calorimeter for hadronic measurements.

The outer and the thickest part of the detector is called the muon spectrometer. Its purpose is to measure the curvature of the muon tracks, particles with mass 105.7MeV (electron has mass 0.511MeV). It has a separate trigger and high-precision tracking chamber able to measure the track curvature of muons in a magnetic field generated by superconducting toroidal magnets with 2 to 6 T across the detector. More about the parameters of the detector can be found in [8]. The schematic depiction of the detector is shown in Figure 2.4.

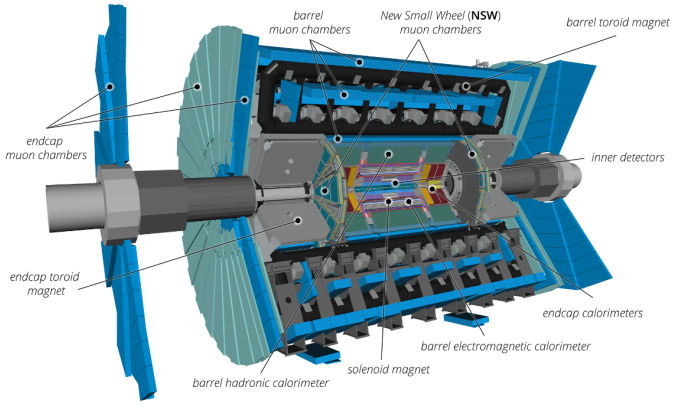



**Figure 2.4:** A schematic depiction of the most important parts of the ATLAS detector.

## 2.4 Charged Higgs boson

In this work, we demonstrate our results on the data simulating a charged Higgs boson which is a hypothetical particle beyond the Standard model that has not been discovered yet. Before the data itself is described let us reveal a glimpse of the theory behind the particle to better understand the structure of the data.

### 2.4.1 Higgs mechanism

Brout-Englert-Higgs (BEH) mechanism was postulated to remedy the consequences of the gauge invariance principle stating that W and Z gauge bosons should be massless [9]. The mechanism defines so-called Higgs field which is a scalar $SU(2)$ (special unitary group) doublet

$$\Phi = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi^{\dagger} \\ \phi^{0} \end{pmatrix} \tag{2.7}$$

with Higgs potential

$$V(\Phi) = \lambda \left(\Phi^{\dagger}\Phi\right)^{2} + \mu^{2} \left(\Phi^{\dagger}\Phi\right). \tag{2.8}$$

For $\mu^2 < 0$ the state with minimum energy is that with $\Phi = 0$ which correctly describes massless photon [10]. However, for $\mu^2 > 0$ the field $\Phi$ obtains a

vacuum expectation value

$$\langle \Phi \rangle = \sqrt{\frac{-2\mu^2}{\lambda}}, \tag{2.9}$$

which leads to spontantenious symmetry breaking and acquisition of mass by gauge bosons (except for photon).

An extension called 2 doublet Higgs Model introduces a second Higgs doublet $(\Phi_1, \Phi_2)$. The consequence of such an extension is the existence of five physical Higgs bosons:

- two neutral CP-even scalars: h (SM Higgs) and H (heavy Higgs);
- two charged Higgs bosons $H^{\pm}$;
- one neutral CP-odd pseudoscalar A.

The free parameters of the model are the mass of the Higgs bosons and $\tan\beta = \nu_2/\nu_1 = \langle \Phi_2 \rangle / \langle \Phi_1 \rangle$ [11].

### 2.4.2 Charged Higgs boson production ($tbH^{\pm}$)

The production of the charged Higgs boson during the $pp$ collisions can be separated into two regions defined by top quark mass $m_t$. For the masses $m_{H^{\pm}} < m_t$, the $H^{\pm}$ is mainly a product of $t$ decay to $H^{\pm}b$ in $tt$ production. The $m_{H^{\pm}} < m_t$, the $H^{\pm}$ region is dominated by two channels $H^{\pm} \to tb$ and $H^{\pm} \to \tau\nu$ [11].However, the subject of our interest is a more exotic channel $H^{\pm} \to hW$.

To better understand why is this specific $tbH^{\pm}$ channel important let us compare it with the $ttH$ decay channel. The comparison is shown in Figure 2.5, indicating the final state for both reactions, $ttH$ and $tbH^{\pm} \to tbWH$ $(H \to \tau\tau)$ . The final states for both reactions are identical.

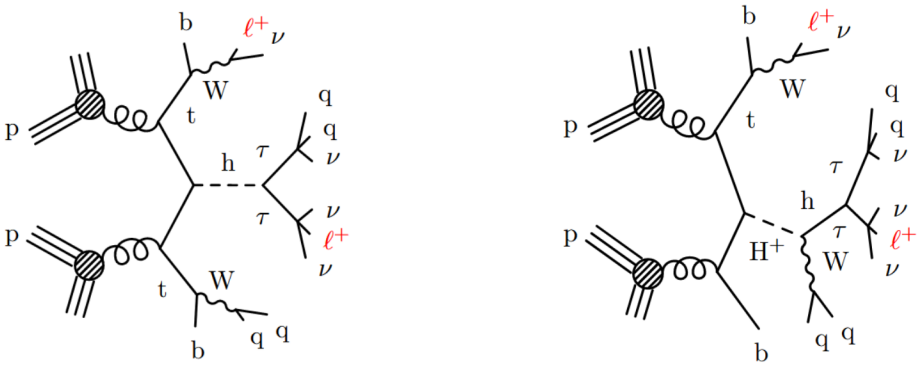



**Figure 2.5:** Left: Production of Higgs boson and two top quarks. Right: Production of the top quark, bottom quark, and charged Higgs boson.

## 2.5 Data representation

In the previous section properties of $tbH^{\pm}$ were described and it was shown why one particular decay channel is of interest. In this thesis, we are focused on reactions belonging to a specific channel called $2lSS + 1\tau$. It means that the final state of all the reactions has to contain two leptons with the same charge and one $\tau$ particle. It was already pointed out that $ttH$ and $tbH^{\pm}$ belong to this channel. We can however introduce more of them: a pair of top quark and anti-top quark ($tt$), a pair of top quarks with W boson ($ttW$), and a pair of top quarks with Z boson ($ttZ$). All five reactions can be registered with the same final state in the detector and the question arises of how they can be distinguished if we don't know anything about the intermediate states. In general, events corresponding to the given reactions can be described by a plethora of observables that can be detected. These observables tend to have different values based on the original reaction they correspond to. This means that statistical tools can be employed to properly assign the events to their reactions. Representation of these observables is explained in section 5.2.

For the purpose of the analysis, the particle we are searching for is called a signal. Except for the signal that might not even be present in reality, many other processes called background occur during data taking. For this thesis, $tbH^{\pm}$ ($tbH^{+}$) has been chosen as the signal and the background constitutes out of the remaining reactions, namely $ttH$, $ttW$, $ttZ$ and $tt$. At this point we could train a classifier and ultimately distinguish between the signal and background events, to do so, however, it is needed to explain the principles of machine learning classification first.

# Chapter 3

# Classification

In machine learning, classification can be viewed as a task in which we are given a vector $\mathbf{x}$ together with a corresponding target variable $y$, and the goal is to predict $y$ given a new value for $\mathbf{x}$. As an example, we can consider our search for a new particle. We want to determine if an input vector $\mathbf{x}$, represented as an event occurring in the detector, is a consequence of the new particle's presence which would be modeled by variable $y$. The $y$ variable is then represented as a binary variable where if $y = 0$ then the event corresponds to class $\mathcal{C}_1$ and if $y = 1$ the event is assigned to class $\mathcal{C}_2$. We then need to solve an inference problem or in other words, find a distribution $p(\mathbf{x}, y)$ that gives a complete probabilistic description of the variables [12, Chapter 5]. Once we have the probabilistic description we need to take a decision step. It means deciding whether the event corresponds to the new particle or not. This chapter describes how the mentioned problems were tackled and what models were used to achieve the results.

## 3.1 Multilayer perceptron

Let us consider a simple linear regression model function

$$y(\mathbf{x},\mathbf{w}) = \mathbf{w}^T\mathbf{x} + w_0, \tag{3.1}$$

where $\mathbf{x}$ is the input data, $\mathbf{w}$ is a weight vector and $w_0$ is bias. To transform the model to solve the classification task we need to apply a non-linear

function $f(\cdot)$ to obtain

$$y(\mathbf{x},\mathbf{w}) = f(\mathbf{w}^T\mathbf{x} + w_0). \tag{3.2}$$

Another step to improve the model would be to consider a vector of basis functions $\phi(\mathbf{x})$ which correspond to the application of fixed nonlinear transformation. By application of such a function, we can achieve linear separability in the new feature space $\phi(\mathbf{x})$ even if in the original space $\mathbf{x}$ the decision boundaries were not linear. The formula for the linear basis function model then reads as follows:

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(x) + w_0\right). \tag{3.3}$$

In general, we could expect that a sufficiently large set $\{\phi_j\}$ would be able to describe arbitrarily complicated function. Unfortunately, significant shortcomings like the curse of dimensionality arise from the assumption of a fixed basis [12, Chapter 5].

The underlying idea behind neural networks is to choose a basis functions $\phi_j(\mathbf{x})$ that are not fixed but themselves have learnable parameters. The most successful choice for the basis functions so far has been a linear combination of inputs similar to 3.3. This assumption can be extended recursively and result in a hierarchical learnable structure. The first layer of the neural network can then be modeled as

$$a_j = \sum_{i=1}^{D} w_{ji} x_i + w_{j0}, \qquad j = 1, \ldots, M, \tag{3.4}$$

where $M$ is the number of linear combinations. The $w_{ij}$ are called weights, the $w_{j0}$ represent biases, and the $a_j$ are referred to as pre-activations. The $a_j$ are then used as arguments for differentiable activation functions

$$z_j = h(a_j), \tag{3.5}$$

which represent the output of basis functions and in the context of neural networks are called hidden units.

Next, let us do the following. For the formula 3.4 we can define $x_0 = 1$ to include the bias $w_0$ in the linear combination to obtain

$$a_j = \sum_{i=0}^{D} w_{ji} x_i. \tag{3.6}$$

Based on this simplification the formula for a two-layer neural network reads as follows:

$$y_k(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=0}^{M} w_{kj} h\left(\sum_{i=0}^{D} w_{ij} x_i\right)\right), \tag{3.7}$$

which can be also written in the compact matrix form as

$$\mathbf{y}(\mathbf{x}, \mathbf{w}) = f(\mathbf{W}^{(2)} h(\mathbf{W}^{(1)} \mathbf{x})). \tag{3.8}$$

At this point, we can easily generalize the formula and obtain

$$\mathbf{z}^{(l)} = h^{(l)}(\mathbf{W}^{(l)} \mathbf{z}^{(l-1)}), \tag{3.9}$$

where $l$ is a number of layers [12, Chapter 6]. This is a function describing the multilayer perception network. We can conclude that the multilayer perceptron corresponds to a recursively nested hierarchy of linear combinations of weight and input vectors activated by nonlinear differentiable functions.

## 3.2 Boosted decision trees

The previous section explained the theoretical idea behind fully connected deep neural networks. Now is the time to introduce the second type of classifier deployed in the thesis called XGBoost. We start our explanation with decision trees and their ensembling. These concepts can then be extended to gradient boosting and finally XGBoost itself. This section is a refined version of the same explanation from [13].

### 3.2.1 Decision tree

A decision tree can be described as a predictor $h : \mathcal{X} \to \mathcal{Y}$. It predicts the target $y$ for a given input vector $\mathbf{x}$ by evaluating the conditions moving from the root node to the leaf of the tree. On this path a child is chosen based on a splitting of the input space [14, Chapter 18]. The splitting is based on one of the features defining the data. A leaf then corresponds to a specific label. Figure 3.1 shows a schematic example of the binary classifier used in this thesis.

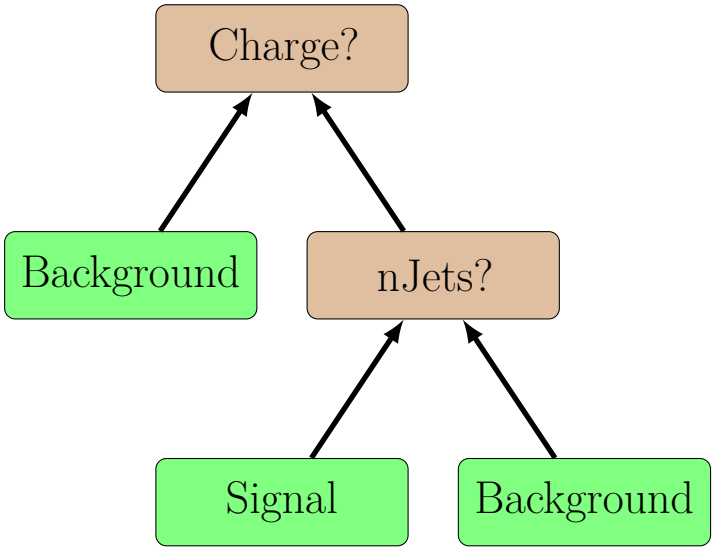



**Figure 3.1:** Decision tree based on particle physics observables. Roots correspond to conditions. Leaves correspond to decisions.

### 3.2.2 Ensemble learning

Ensemble learning is a machine learning paradigm where multiple models are trained to solve the same problem and combined to get better results. The main hypothesis is that when weak models are correctly combined, we can obtain more accurate models. The tree main classes of ensemble learning machines are bagging, stacking, and boosting [15]. In our case, we will focus on the boosting method.

### 3.2.3 Boosting

The boosting is a type of ensembling that uses a generalization of linear predictors to address mainly two important issues. The first one is the bias-complexity tradeoff. It means a better balance between approximation and estimation error of an Empirical Risk Minimization (ERM) learner [14, Chapter 2.2]. The second issue is the computational complexity of learning. It is sometimes infeasible to find ERM hypothesis therefore boosting amplifies the accuracy of a high number of weak learners. Weak learner usually performs just slightly better than a random guess and comes from simple classes. However, if implemented efficiently and aggregated with more similar learners it can achieve good prediction accuracy for even difficult classes [14, Chapter 10].

### 3.2.4 Gradient boosting

In gradient boosting, models are fit using any arbitrary differentiable loss function and gradient descent optimization algorithm. This gives the technique its name, as the loss gradient is minimized as the model is fit, it is very similar to how neural networks work. The theoretical idea behind gradient boosting is the following.

We are given some data $(\boldsymbol{x}_i, y_i)_{i=1}^n$ where $\boldsymbol{x}$ is a vector of input variables and $y$ is a target vector. Usually, a negative log-likelihood is taken as a loss function but in general, it can be any differentiable loss function in a form

$$\mathcal{L}(y, F(\mathbf{x})), \tag{3.10}$$

where $y$ is observed and $F(\mathbf{x})$ is a function equivalent to predicted value. Minimization formula is then

$$F_0(\mathbf{x}) = \arg\min_p \sum_{i=1}^{n} \mathcal{L}(y, p), \tag{3.11}$$

where $F_0(x)$ is initial model value and $p$ corresponds to predicted value. To minimize the loos function, we compute

$$r_{im} = -\left[\frac{\partial \mathcal{L}(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} \text{for } i = 1, ..., n, \tag{3.12}$$

where $r_{im}$ is the diferentiated pseudo-residual for sample $i$ and tree $m$ that we are building. This step corresponds approximately to the subtraction of a predicted value from the observed value.

Now we have to fit the regression and classification trees (CART) to pseudo-residuals and label terminal regions, $R_{jm}$ where $j$ is a number of regions that correspond to leaves of the tree. To determine the output values, we compute,

$$p_{jm} = \arg\min_p \sum_{x_i \in R_{ij}} \mathcal{L}(y_i, F_{m-1}(x_i) + p), \tag{3.13}$$

which is the same formula as 3.11 but the previous prediction is now considered. To compute the output value for each leaf, we pick only $y$ values that correspond to this region. Then we are looking for $p$ that minimizes the term. The final step is to make a new prediction, as

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \sum_{j=1}^{J_m} p_{jm} I(\mathbf{x} \in R_{jm}), \tag{3.14}$$

where $\nu$ is a learning rate. It means we take the original prediction and add corresponding leave values from our newly built tree multiplied by the learning rate. Then, we continue with the next iteration. It is desirable to mention that this step represents the boosting method described in 3.2.3. More information can be found in [16].

### 3.2.5 XGBoost

XGboost is a scalable end-to-end tree boosting system allowing to achieve state-of-the-art results on many machine learning tasks. Novel techniques are sparse-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More insight on cache access patterns is achieved to build a scalable tree boosting system[17].

Let us briefly explain how XGBoost builds its trees and what new ideas it offers compared to the gradient boosting algorithm. We start with an objective function

$$\begin{gathered}\mathcal{L}(\phi) = \sum_i l(y_i, p_i) + \sum_k \Omega(f_k), \\ \text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2.\end{gathered} \tag{3.15}$$

Here $l$ is a differentiable convex loss function, T is a number of leaves in a tree and $\Omega$ is a regularization term that helps to prevent overfitting. $i$ is the instance number. The formula 3.15 can not be optimized by standard optimization methods, therefore it is solved in additive manner

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t), \tag{3.16}$$

where $f_t(\mathbf{x}_i)$ is the best choice function for $t$-th iteration. After second-order Taylor approximation and removal of the constant term, we obtain

$$\mathcal{L}^{(t)} \approx \sum_{1}^{n} \left[ g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i) \right] + \Omega(f_t), \tag{3.17}$$

where $g$ is gradient ant $h$ corresponds to hessian.
If we define $I_j = \{i | q(\boldsymbol{x_i}) = j\}$, then we can write

$$\tilde{\mathcal{L}} = \sum_{j=1}^{T} \left[ \left( \sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left( \sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T. \tag{3.18}$$

Here outer sum just iterates over all leaves. The first inner sum is an expansion of sum from 3.17, where we sum all the gradients belonging to one leaf and

multiply them by the corresponding leaf output value w. The second sum is the same, except for $\lambda$ the term that came from the expansion of $\Omega$. When we minimize

$$\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2 \tag{3.19}$$

by standard method, we obtain

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}. \tag{3.20}$$

Now we just plug $w_j^*$ in equation 3.18 and obtain

$$\tilde{\mathcal{L}}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} + \gamma T. \tag{3.21}$$

Let us reiterate that $g_i$ and $h_i$ are gradients and Hessians corresponding to chosen loss function $l$ that we could substitute for some particular example as negative log-likelihood. Equation 3.21 measuring the quality of a tree structure $q$. The greedy algorithm is then used to iteratively add nodes while aiming to reduce the loss. The algorithm is called greedy because the overall structure is not considered when building the trees. The tree is built in such a way that the condition is chosen according to the division made by this condition, even though there could be a better one if combined with another consequential condition. More information about XGBoost features and full derivation is given in [16].

## 3.3 Significance and performance assessment

To assess the performance of our classifiers, we use various techniques from the machine learning background like precision, f1 score, or accuracy. For particle physics purposes, however, a different metric called significance is used. In the following section, we describe the theory behind it and how it relates to our classification results.

### 3.3.1 Counting experiments

In experimental particle physics, the appearance of events in time and position of the detector follows a Poisson distribution. A discrete random variable can be described by a Poisson distribution if the following conditions are fulfilled:

- Individual events are represented by integers
- If an event occurs, it does not affect the occurrence of another event.
- The average rate at which events occur is independent of any occurrences.

A probability mass function is then given like

$$f(n) = \frac{\lambda^n e^{-\lambda}}{n!}, \qquad n \in \mathbb{N}, \tag{3.22}$$

where $n$ is a number of occurrences and $\lambda > 0$ is a parameter corresponding to expected values of $n$. It is desirable to point out that for a large value of $\lambda$ a Poisson distribution transforms into a Normal distribution with mean and variance given by $\lambda$. We can then write

$$\text{Poisson}(\lambda) \approx \mathcal{N}(\mu = \lambda, \sigma^2 = \lambda), \qquad \text{for large } \lambda. \tag{3.23}$$

This result offers us a useful simplification for the data. We no longer need to assume the Poisson distribution but instead, utilize the advantages of Gaussian distribution properties.

### 3.3.2 Hypothesis testing

The particle physics analysis is usually based on hypothesis testing. When we are looking for a new particle beyond the Standard model our hypothesis $H_1$ could be stated as the existence of such a particle. The corresponding null hypothesis $H_0$ then assumes the correctness of the Standard model meaning that the particle beyond the Standard model does not exist [18].

To test the hypothesis a test statistic is defined as a quantity derived from the data that can indicate whether the hypothesis is true or not. In the previous section, we explained that the data is described by a Poisson distribution but for a large number of events, there is an entanglement with a Gaussian distribution described by 3.23. The level of agreement with a given hypothesis $H$ is then obtained by computing the p-value. It is a probability, under the assumption of $H$, of finding data of equal or greater incompatibility with the predictions of $H$. The hypothesis is excluded if the obtained p-value is below a specific threshold. In particle physics, a quantity called significance is usually used instead. It is defined as

$$Z = \Phi^{-1}(1 - p), \tag{3.24}$$

where $\Phi^{-1}$ is the quantile.

### 3.3.3 Approximate formula

Now let us assume that for each event from the signal, a variable $x$ is measured and these values are used to construct a histogram $\mathbf{n} = (n_1, \ldots, n_N)$. The expectation value is then given as

$$E[n_i] = \mu s_i + b_i, \tag{3.25}$$

where $s_i$ and $b_i$ are numbers of entries in the $i$th bin from signal and background and $\mu$ is a parameter determining a signal strength. The $\mu = 0$ then states the background only hypothesis and $\mu = 1$ the nominal signal hypothesis. Usually, $\mu$ is assumed to be non-negative.

To test a hypothesized value of $\mu$ we use the profile likelihood ratio defined as

$$\lambda(\mu) = \frac{L(\mu, \hat{\hat{\boldsymbol{\theta}}})}{L(\hat{\mu}, \hat{\boldsymbol{\theta}})}, \tag{3.26}$$

where the numerator is a profile likelihood function. The quantity $\hat{\hat{\boldsymbol{\theta}}}$ represents the value of $\boldsymbol{\theta}$ that maximises $L$ for the specific $\mu$. The denominator is the maximized likelihood function. The test statistic is then defined as

$$t_\mu = -2 \ln \lambda(\mu). \tag{3.27}$$

From the definition 3.26 we can infer $0 \leq \lambda \leq 1$. If $\lambda$ is close to 1, good agreement with the data can be assumed.

If only one bin histogram is assumed for simplicity we can put these assumptions together and define test statistics $q_0$ for the discovery of a positive signal and known background as

$$q_0 = \begin{cases} -2 \ln \frac{L(0)}{L(\hat{\mu})}, & \hat{\mu} \geq 0, \\ 0, & \hat{\mu} < 0, \end{cases} \tag{3.28}$$

where the likelihood function is defined as follows

$$L(\mu) = \frac{(\mu s + b)^n}{n!} e^{-(\mu s + b)}. \tag{3.29}$$

It can be shown that

$$Z_0 = \sqrt{q_0} = \begin{cases} \sqrt{2(n \ln \frac{n}{b} + b - n)}, & \hat{\mu} \geq 0, \\ 0, & \hat{\mu} < 0. \end{cases} \tag{3.30}$$

Median significance for the nominal signal hypothesis can then be approximated by stating $n = s + b$ to obtain

$$Z_{0,med} = \sqrt{2\left((s+b)\ln\left(1+\frac{s}{b}\right) - s\right)}. \tag{3.31}$$

After the expansion of the logarithm in $s/b$ we get

$$Z_{0,med} = \frac{s}{\sqrt{b}}(1 + \mathcal{O}(\frac{s}{b})) \approx \frac{s}{\sqrt{b}}. \tag{3.32}$$

which can be used if $s \ll b$.

Definitions 3.31 and 3.32 are commonly used approximations of significance and therefore were also used in the thesis to measure the separation power of the classifier. The original proof paper [19] provides a more detailed derivation of the significance formulas.

# Chapter 4

# Augmentation

In this chapter, the data augmentation approach is explained. As already mentioned in the introduction, the generative model called variational autoencoder was chosen for the purposes of this thesis. [20]. Firstly the focus is put on a standard learning approach with evidence lower bound (ELBO) maximization. Then the novel approach called Symmetric equilibrium learning (SEL) [21] is introduced and the final part discusses the extension of these algorithms for hierarchical (Ladder) VAEs [22].

## 4.1 VAE with evidence lower bound learning

Derivation of the standard evidence lower bound learning is mostly inspired by [20].

### 4.1.1 Generative model

Firstly, let us explain the underlying theory in general. Given the training data $\mathcal{X} \subset \mathbb{R}^m \times \{0,1\}^m$ drawn from the dataset, we want to find a distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$ parametrized by $\boldsymbol{\theta}$ that maximizes the probability of observed data $\mathbf{x} \in \mathcal{X}$. The standard approach how to do so is to introduce an unobservable latent variable denoted by $\mathbf{z}$ from $\mathcal{Z} \subset \mathbb{R}^n$ and learn the relations between

them and the datapoints $\mathbf{x} \in \mathcal{X}$ [23]. In general, there is not a precise definition of what the $\mathbf{z}$ should look like, therefore we have to make an assumption by defining $p(\mathbf{z})$ which is called the prior distribution and usually corresponds to $\mathcal{N}(0, 1)$. At this point, we can write the following relation

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z}), \tag{4.1}$$

which states that the probability of a given $\mathbf{x}$ occurring with a specific latent $\mathbf{z}$ is computed as a product of likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ (a probability of $\mathbf{x}$ belonging to a specific $\mathbf{z}$ and prior $p(\mathbf{z})$). The likelihood is also called a stochastic decoder and can be represented by a neural network with parameters $\boldsymbol{\theta}$. It is already known to us that even though every $\mathbf{x}$ corresponds to some $\mathbf{z}$ we can never observe $\mathbf{z}$ directly, meaning $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ can be learned only by trying all the possible $\mathbf{z}$'s written as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \sum_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}), \tag{4.2}$$

which is unfortunately intractable for larger $\mathbf{z}$ and can not be computed directly. The only other option is to substitute precise computation by sampling which can be done with the help of already defined prior $p(\mathbf{z})$. We can apply 4.1 and obtain

$$\log \sum_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = \log \sum_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = \log \left[\mathbb{E}_{p(\mathbf{z})} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]. \tag{4.3}$$

A lower bound can then be found by applying Jensen's inequality

$$\log \left[\mathbb{E}_{p(\mathbf{z})} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] \geq \sum_{\mathcal{Z}} p(\mathbf{z}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathbb{E}_{p(\mathbf{z})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}). \tag{4.4}$$

It is important to point out that at this moment we are no longer optimizing $p(\mathbf{x})$ but its lower bound which can be sometimes considerably far from the true value. The following approach minimizes the gap between the lower bound and the true value.

To find a tighter lower bound we need to explain a notion of posterior distribution denoted by $p(\mathbf{z}|\mathbf{x})$ for which applies the following identity:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}. \tag{4.5}$$

From 4.2 we already know that that $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ is intractable which implies intractability for $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. To obtain a tractable problem we introduce an entity called approximate posterior or stochastic encoder denoted by $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ where $\boldsymbol{\phi}$ indicates its parameters. The distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can be parametrized by a neural network in which case $\boldsymbol{\phi}$ corresponds to the weights and biases of the network. The schematic structure of the model is shown in Figure 4.1.

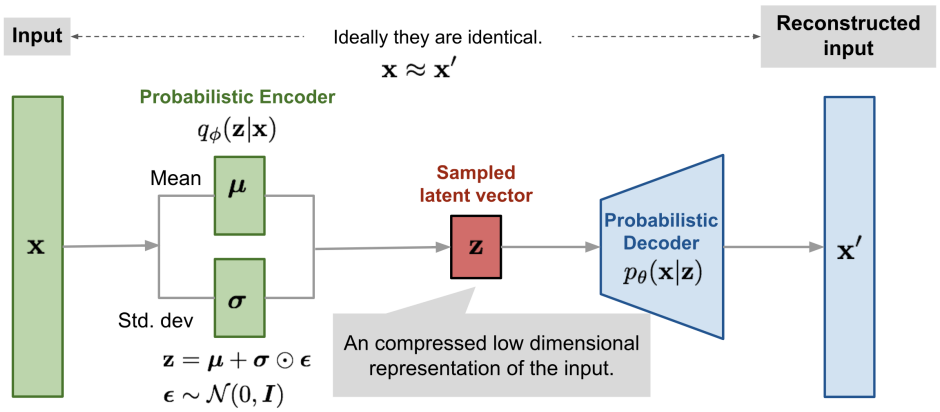



**Figure 4.1:** Schematic depiction of a variational autoencoder model [24].

### 4.1.2 Evidence lower bound (ELBO)

The optimization objective of the variational autoencoder is the evidence lower bound, abbreviated as ELBO and sometimes also called a variational lower bound. For an arbitrary inference model $q_\phi(\mathbf{z}|\mathbf{x})$ with parameters $\boldsymbol{\phi}$ we have

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \left[ \sum_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) \right] = \log \left[ \sum_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \right]. \tag{4.6}$$

So far, we have only expanded the expression. Now let us multiply both the numerator and denominator by $q_\phi(\mathbf{z}|\mathbf{x})$ to obtain:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \left[ \sum_{\mathcal{Z}} \left[ \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \right] \right]. \tag{4.7}$$

The sum can then be replaced by an expected value

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \left[ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \frac{1}{q_\phi(\mathbf{z}|\mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \right] \right]. \tag{4.8}$$

Thanks to Jensen's inequality we can move the logarithm inside and obtain

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \left[ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \left[ p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \right], \tag{4.9}$$

which can be rewritten as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]. \tag{4.10}$$

The first term is called reconstruction loss and in the second term, we can recognize KL divergence of the $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z})$. The final equation for the ELBO reads:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] + D_{KL}\left(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right). \tag{4.11}$$

### 4.1.3 Stochastic gradient descent optimization

Given the dataset with i.i.d. data, the ELBO objective is the sum or average of the individual data points

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{T}) = \sum_{\mathbf{x}\in\mathcal{T}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}). \tag{4.12}$$

The gradient of individual datapoint $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$ is in general intractable but good and unbiased estimator $\tilde{\nabla}_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$ can be found. An unbiased gradient of ELBO w.r.t. parameter $\boldsymbol{\theta}$ is easy to obtain from

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \nabla_{\boldsymbol{\theta}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right] \tag{4.13}$$

since the ELBO expectation does not depend on $\boldsymbol{\theta}$, the gradient can be moved inside. Expectation value can then be substituted by random sampling from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to obtain

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) \simeq \nabla_{\boldsymbol{\theta}}(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) = \nabla_{\boldsymbol{\theta}}(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}). \tag{4.14}$$

For gradient w.r.t. $\boldsymbol{\phi}$ however the differentiation is not that simple since the expectation depends on this parameter. In general, estimation is needed, but in the case of continuous latent variables, we can approach the problem using a change of variables.

### 4.1.4 Reparametrization trick

To ensure the ability to differentiate w.r.t. both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ a technique called reparametrization trick is used. The first step is to express $\mathbf{z}$ as a differentiable transformation of random variable $\boldsymbol{\epsilon}$:

$$\mathbf{z} = g(\boldsymbol{\epsilon},\boldsymbol{\phi}), \tag{4.15}$$

where the distribution is independent of $\mathbf{x}$ or $\boldsymbol{\phi}$. Expectation dependency can then be rewritten as

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})}[f(\mathbf{z})] = \mathbb{E}_{p_{(\boldsymbol{\epsilon})}}[f(\mathbf{z})]. \tag{4.16}$$

Computation of the gradient is now feasible and a simple Monte Carlo estimator can be formed:

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{p_{(\boldsymbol{\epsilon})}}[f(\mathbf{z})] = \mathbb{E}_{p_{(\boldsymbol{\epsilon})}}[\nabla_{\boldsymbol{\phi}} f(\mathbf{z})] \simeq \nabla_{\boldsymbol{\phi}} f(g(\boldsymbol{\epsilon}, \boldsymbol{\phi})). \tag{4.17}$$

Figure 4.4 shows the graphical comparison of the original and reparametrized versions.

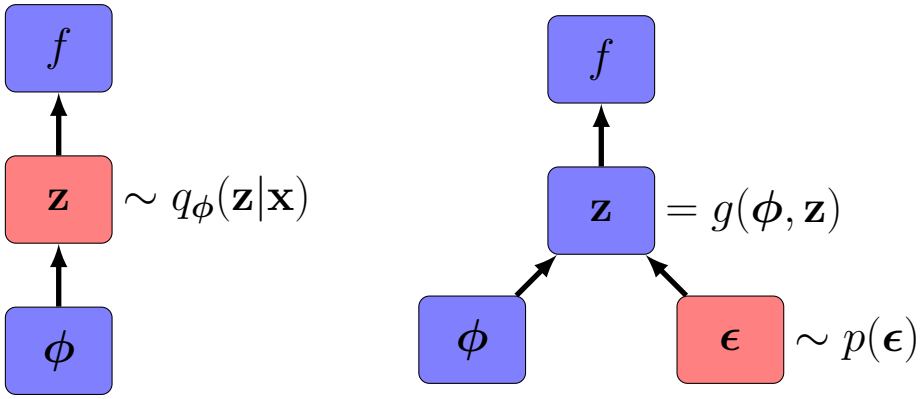



**Figure 4.4:** Left: Path for backpropagation is blocked since we cannot differentiate $f$ w.r.t. $\boldsymbol{\phi}$. It means that gradients cannot be backpropagated through the random variable $\mathbf{z}$. Right: Variable $\mathbf{z}$ is reparametrized to become deterministic. The randomness is ensured by the newly introduced random variable $\boldsymbol{\epsilon}$.

## 4.2 VAE with Symmetric Equilibrium Learning (SEL)

The second learning approach considered in the thesis is called Symmetric Equilibrium Learning [21]. Symmetric means treating the encoder and decoder evenly, which contrasts with the standard learning approach that optimizes ELBO using the encoder only for auxiliary purposes.

The goal of the algorithm is the same as in the previous case, jointly learn an encoder-decoder pair $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ by optimizing the likelihood of the observed data but also enforcing the encoder and decoder consistency at the same time. The task is then formulated symmetrically as finding a Nash equilibrium of a two-player game. The decoder corresponds to the first-player strategy and the encoder corresponds to the second-player strategy. The utility function of a player is the likelihood of the training data w.r.t. its strategy. Consequently, training examples are completed by the other player's

strategy. The utility functions are modeled as follows:

$$\mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] \tag{4.18}$$

$$\mathcal{L}_q(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})} \left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right]. \tag{4.19}$$

A Nash equilibrium of the game is then a pair $(\boldsymbol{\theta}_*, \boldsymbol{\phi}_*)$ such that

$$\mathcal{L}_p(\boldsymbol{\theta}_*, \boldsymbol{\phi}_*) \geq L_p(\boldsymbol{\theta}, \boldsymbol{\phi}_*) \quad \forall \boldsymbol{\theta} \tag{4.20}$$

$$\mathcal{L}_p(\boldsymbol{\theta}_*, \boldsymbol{\phi}_*) \geq L_q(\boldsymbol{\theta}_*, \boldsymbol{\phi}) \quad \forall \boldsymbol{\phi} \tag{4.21}$$

which describes a point at which neither player can improve its objective function.

### 4.2.1 Stochastic gradient descent optimization

Considering the gradient descent algorithm, each player tries to improve its utility w.r.t. its strategy

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\phi}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] \tag{4.22}$$

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_q(\boldsymbol{\theta}, \boldsymbol{\phi}) = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})} \left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right]. \tag{4.23}$$

An unbiased gradient estimates are obtained by differentiating Monte-Carlo estimates of the expectations as described for ELBO learning. In comparison to the ELBO, $\nabla_{\boldsymbol{\theta}} \mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\phi})$ does not need to be differentiated w.r.t. $\boldsymbol{\phi}$, and the same is true for the $\nabla_{\boldsymbol{\phi}} \mathcal{L}_q(\boldsymbol{\theta}, \boldsymbol{\phi})$ and differentiation w.r.t. $\boldsymbol{\theta}$. This notion completely eliminates the need for reparametrization in the case of continuous variables or special estimation in other cases.

## 4.3 HVAE with evidence lower bound learning

So far, we have explained variational autoencoders with just one latent variable. However, in practice, these can be chained to achieve improved results in several domains. For hierarchical VAE we can choose a very straightforward extension of the standard VAE that would look as follows.

At the beginning let us define two latent variables $\mathbf{z}_1$ and $\mathbf{z}_2$. The joint distribution we want to learn is then factorized as

$$p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2) = p(\mathbf{x}|\mathbf{z}_1) p(\mathbf{z}_1|\mathbf{z}_2) \tag{4.24}$$

which implies the generative process that starts with sampling of $\mathbf{z}_2$ from $\mathcal{N}(0,1)$ and continues by sampling $\mathbf{z}_1$ and $\mathbf{x}$ given their predecessors. We already know that computing true posterior is intractable therefore we define a family of variational posteriors $Q(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x})$. Factorization is then performed in the reverse direction as follows:

$$Q(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x})q(\mathbf{z}_2|\mathbf{z}_1, \mathbf{x}) \tag{4.25}$$

This extension seems to be very straightforward but in some cases can lead to suboptimal behavior. A broader discussion can be found in [25, Chapter 4.5]. For the purposes of this thesis, another approach was chosen that we explain in the following section.

### 4.3.1 Ladder variational autoencoder (LVAE)

In the standard HVAE approach, the dependencies of the generative and variational paths are in reverse order 4.25. The idea behind the Ladder variation autoencoder is to change the order in which the inference is done to provide a tighter connection through a shared parametrization. It was introduced in [22]. The class of variational posteriors changes as follows:

$$Q(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x})q(\mathbf{z}_2|\mathbf{x}). \tag{4.26}$$

The definition of the inference and generative path can be generalized into a model where the $\mathbf{z}$ variable is split into $N$ layers. The generative part reads as follows:

$$p_{\boldsymbol{\theta}} = p_{\boldsymbol{\theta}}(\mathbf{z}_N) \prod_{i=1}^{N-1} p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{z}_{i+1}). \tag{4.27}$$

$$p_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{z}_{i+1}) = \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_{p,i}(\mathbf{z}_{i+1}), \sigma^2_{p,i}(\mathbf{z}_{i+1}), \quad p_{\boldsymbol{\theta}}(\mathbf{z}_N) = \mathcal{N}(\mathbf{z}_N|\mathbf{0}, \mathbf{1}) \tag{4.28}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{p,0}(\mathbf{z}_1), \boldsymbol{\sigma}^2_{p,0}(\mathbf{z}_1)), \quad p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) = \mathcal{B}(\mathbf{x}|\boldsymbol{\mu}_{p,0}(\mathbf{z}_1)). \tag{4.29}$$

The definition is also given for Bernoulli distribution since it is utilized in this thesis. The inference path is then defined similarly

$$q_{\boldsymbol{\phi}} = q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{x}) \prod_{i=2}^{N} q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{z}_{i-1}). \tag{4.30}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\mathbf{z}_1|\boldsymbol{\mu}_{q,1}(\mathbf{x}), \boldsymbol{\sigma}^2_{q,1}(\mathbf{x})) \tag{4.31}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{z}_{i-1}) = \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_{q,i}(\mathbf{z}_{i-1}), \sigma^2_{q,i}(\mathbf{z}_{i-1})). \tag{4.32}$$

To summarize, let us go back to our simplified model with just $\mathbf{z}_1$ and $\mathbf{z}_2$. We have a top-down generative path that stays the same defined by

$p(\mathbf{x}|\mathbf{z}_1)$, $p(\mathbf{z}_1|\mathbf{z}_2)$ and $p(\mathbf{z}_2)$. $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ parameters are given by deep neural networks as usual. On the contrary, the bottom-up inference path introduces a new concept of deterministic paths that transforms the input data $\mathbf{x}$ by a function $\mathbf{d}_1 = f_1(\mathbf{x})$ and $\mathbf{d}_2 = f_2(\mathbf{d}_1)$ and recursively corrects the generative distribution. The function $f$ is defined by a deep neural network that serves as input for an additional network that produces modifications in $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ parameters. From this point forward, we will refer to the HVAE architecture as LVAE to emphasize the approach utilized.

### 4.3.2 ELBO extension for LVAE

In the section 4.1.2 we derived the ELBO for standard variational autoencoder. This time, however, we must consider an extended form of the joint distribution we aim to learn, as well as a new family of variational posteriors discussed at the beginning of this section. Now let us go back to equation 4.8 where we stated the ELBO representation using Jensen's inequality. We can rewrite it considering the LVAE paradigm as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \left[ \int_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \frac{q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x})q(\mathbf{z}_2|\mathbf{x})}{q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x})q(\mathbf{z}_2|\mathbf{x})} p(\mathbf{x}|\mathbf{z}_1)p(\mathbf{z}_1|\mathbf{z}_2)p(\mathbf{z}_2) \right]. \quad (4.33)$$

Now let us rewrite the expression in the form of expectation value to obtain

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \left[ \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \frac{1}{q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x})q(\mathbf{z}_2|\mathbf{x})} p(\mathbf{x}|\mathbf{z}_1)p(\mathbf{z}_1|\mathbf{z}_2)p(\mathbf{z}_2) \right] \right] \quad (4.34)$$

and rewrite it to a better readable form

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \left[ \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ p(\mathbf{x}|\mathbf{z}_1) \frac{p(\mathbf{z}_1|\mathbf{z}_2)}{q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x})} \frac{p(\mathbf{z}_2)}{q(\mathbf{z}_2|\mathbf{x})} \right] \right]. \quad (4.35)$$

At this point, we apply Jensen's inequality to obtain

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \log p(\mathbf{x}|\mathbf{z}_1) \right] + \\ + \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \log \frac{p(\mathbf{z}_1|\mathbf{z}_2)}{q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x})} \right] + \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \log \frac{p(\mathbf{z}_2)}{q(\mathbf{z}_2|\mathbf{x})} \right]. \end{aligned} \quad (4.36)$$

The final equation for LVAE ELBO then reads as

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \log p(\mathbf{x}|\mathbf{z}_1) \right] + \\ \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \log \frac{p(\mathbf{z}_1|\mathbf{z}_2)}{q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x})} \right] + \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \log \frac{p(\mathbf{z}_2)}{q(\mathbf{z}_2|\mathbf{x})} \right], \end{aligned} \quad (4.37)$$

which can be rewritten as

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{Q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})} \left[ \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) \right] + \\ + D_{KL} \left( q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x}) || p(\mathbf{z}_1|\mathbf{z}_2) \right) + D_{KL} \left( q(\mathbf{z}_2|\mathbf{x}) || p(\mathbf{z}_2) \right). \end{aligned} \quad (4.38)$$

The final equation can be easily extended for $N$ latent variables as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{Q(\mathbf{z}_1,\ldots,\mathbf{z}_N|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1)\right] + \\ + \sum_{i=1}^{N-1} D_{KL}\left(q(\mathbf{z}_i|\mathbf{z}_{i+1},\mathbf{x})||p(\mathbf{z}_i|\mathbf{z}_{i+1})\right) + D_{KL}\left(q(\mathbf{z}_N|\mathbf{x})||p(\mathbf{z}_N)\right). \tag{4.39}$$

### 4.3.3 Distribution parameter shift

The modification of parameters can be done in several ways. We will follow the original approach from [22] that derives the update from the exponential family representation of the Gaussian distribution.

Given a measure $\eta$, we define an exponential family of probability distributions as those distributions whose density (relative to $\eta$) have the following general form [26]:

$$p(x|\eta) = h(x)\exp\left[\eta^T T(x) - A(\eta)\right], \tag{4.40}$$

where $\eta$ is the canonical parameter, $T(x)$ is referred to as sufficient statistics and $A(\eta)$ is knows as the cumulant function.

Let us start with the Gaussian density function that can be written as follows:

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.41}$$

which can be expanded as

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}}\exp\left[\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 - \log\sigma\right]. \tag{4.42}$$

Then in exponential family form, we have

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}}\exp\left[\left\langle\begin{pmatrix} x \\ x^2\end{pmatrix}, \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2\end{pmatrix}\right\rangle - \frac{\mu^2}{2\sigma^2} - \log\sigma\right]. \tag{4.43}$$

For the canonical parameter $\eta$ we can write

$$\begin{pmatrix}\eta_1 \\ \eta_2\end{pmatrix} = \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2\end{pmatrix}, \tag{4.44}$$

which can be rewritten for $\mu$ and $\sigma^2$ like

$$\begin{pmatrix}\mu \\ \sigma^2\end{pmatrix} = \begin{pmatrix} -\eta_1/2\eta_2 \\ -1/2\eta_2\end{pmatrix}. \tag{4.45}$$

Now let us assume an increment in $\eta$

$$\tilde{\eta} = \eta + \hat{\eta}, \tag{4.46}$$

we can plug in the derived value for $\eta$ and obtain

$$\tilde{\eta}_1 = \frac{\mu}{\sigma^2} + \frac{\hat{\mu}}{\hat{\sigma}^2} \tag{4.47}$$

$$\tilde{\eta}_2 = -\frac{1}{2\sigma^2} - \frac{1}{2\hat{\sigma}^2}. \tag{4.48}$$

To compute increment in $\mu$ and $\sigma^2$ we plug 4.47 second equation in 4.45 second equation and after algebraic manipulation obtain

$$\tilde{\sigma}^2 = \frac{\sigma^2\hat{\sigma}^2}{\sigma^2 + \hat{\sigma}^2}. \tag{4.49}$$

The $\mu$ parameter is computed similarly. The result looks as follows:

$$\tilde{\mu} = \frac{\mu\hat{\sigma}^2 + \hat{\mu}\sigma^2}{\sigma^2 + \hat{\sigma}^2}. \tag{4.50}$$

The expression 4.49 and 4.50 represent the modified parameters of Gaussian distribution. The $\hat{\mu}$ and $\hat{\sigma}^2$ carry the inference bottom-up information whereas $\mu$ and $\sigma$ carry the generative top-down prior information. This parametrization has a probabilistic motivation by viewing $\hat{\mu}$ and $\hat{\sigma}^2$ as the approximate Gaussian likelihood that is combined with a Gaussian prior $\mu$ and $\sigma^2$ from the generative distribution. Together these form the approximate posterior distribution $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{z}, \mathbf{x})$ using the same top-down dependency structure both in the inference and generative model.

## 4.4 LVAE with symmetric equilibrium learning

In [21] authors mention the possibility of extending the standard model to a hierarchical one. "The principles of the learning algorithm remain the same, however, refinement needs to be made in utility functions. In the general LVAE form, they look as follows:

$$\mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] \tag{4.51}$$

$$\mathcal{L}_q(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right]. \tag{4.52}$$

In general encoder and decoder factorize as

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}_m)\prod_{i=1}^{m-1} p(\mathbf{z}_i|\mathbf{z}_{i+1})p(\mathbf{x}|\mathbf{z}_1) \tag{4.53}$$

$$q(\mathbf{x}|\mathbf{x}) = q(\mathbf{z}_m|\mathbf{x})\prod_{i=1}^{m} q(\mathbf{z}_i|\mathbf{z}_{i+1}, \mathbf{x}). \tag{4.54}$$

For the two-layer latent space, the equation is simplified as

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}_2)p(\mathbf{z}_1|\mathbf{z}_2)p(\mathbf{x}|\mathbf{z}_1) \tag{4.55}$$
$$q(\mathbf{x}|\mathbf{x}) = q(\mathbf{z}_2|\mathbf{x})q(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x}). \tag{4.56}$$

Now we can plug the factorization into the utility functions and obtain

$$\mathcal{L}_p(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) + \log p_{\boldsymbol{\theta}}(\mathbf{z}_1|\mathbf{z}_2)\right] \tag{4.57}$$
$$\mathcal{L}_q(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{x}) + \log q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x})\right]. \tag{4.58}$$

## 4.5 Visualization and evaluation metrics

In this section, we discuss how to visualize data with a high number of dimensions. Dimension reduction is a technique that maps high-dimensional space $\mathcal{H}^n$ to lower dimensional space $\mathcal{L}^m$ where usually $m = 2$. The goal of the algorithm is then to perform the embedding in such a way that the internal structure of the data remains preserved as much as possible [27].

Let us start our explanation with the original method called Stochastic Neighbor Embedding. Once we understand the basics we can move to a more advanced approach. The SNE explanation is based on [28] and the t-SNE explanation on [29].

### 4.5.1 Stochastic Neighbor Embedding (SNE)

The first step of the algorithm is to compute asymmetric high-dimensional probabilities. It means for every object $i$ and its potential neighbor $j$ compute probability $p(i|j)$ defined as

$$p(i|j) = \frac{e^{-d_{ij}^2}}{\sum_{k \neq i} e^{-d_{ik}^2}}. \tag{4.59}$$

The quantity $d_{ij}^2$ is called dissimilarity and is defined as

$$d_{ij}^2 = \frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2}, \tag{4.60}$$

which is Euclidean distance between high dimensional points $\mathbf{x}_i$ and $\mathbf{x}_j$ scaled by parameter $\sigma$. The parameter is then found by computation of

$$H = -\sum p(i|j) \log_2 p(i|j) = \log_2 k, \tag{4.61}$$

where parameter $k$ is called perplexity and is chosen by hand. Once the perplexity is chosen, the $\sigma$ is found to make the entropy of the distribution equal to $\log_2 k$. The second step is to calculate low-dimensional probabilities from the high-dimensional ones. This time Gaussian neighborhood is computed with a fixed variance. The induced probability is a function of low-dimensional images $\mathbf{y}_i$ of the original points and is given as

$$q(i|j) = \frac{e^{-||\mathbf{y}_i - \mathbf{y}_j||^2}}{\sum_{k \neq i} e^{-||\mathbf{y}_i - \mathbf{y}_k||^2}}, \tag{4.62}$$

If the point is correctly placed in the low-dimensional space the difference between $p$ and $q$ is small. The similarity of the distributions is measured by the already mentioned $KL$ divergence. The cost function is therefore defined as $D_{KL}$ over all points

$$\sum_i \sum_j D_{KL}(p(i|j)||q(i|j)). \tag{4.63}$$

### 4.5.2 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is the extension of the original SNE algorithm addressing the difficulty of cost function optimization and crowding problems. The first improvement over the original version is the symmetrization of the conditional probabilities. So far the probability that a point $\mathbf{x}_i$ considers another point $\mathbf{x}_j$ as its neighbor is not the same. We therefore define pairwise probabilities in high-dimensional space as

$$p_{ij} = \frac{p(j|i) + p(i|j)}{2n}. \tag{4.64}$$

The crowding problem is related to distortion of the distances in the low-dimensional space. For example, in $n$ dimensional space we can have $n + 1$ equidistantly placed points but it certainly ca not be achieved in 2D space. The crowding problem is defined as follows: The area of the two-dimensional map that is available to accommodate moderately distant data points will not be nearly large enough compared with the area available to accommodate nearby data points. Therefore, points that should be close to each other can be placed at high distances. To solve this issue the Student t-Distribution is introduced

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l)||^2)^{-1}}. \tag{4.65}$$

At this point, both pair-wise probabilities are symmetric which is a significant advantage when computing the gradients of the loss function.

### 4.5.3 Wasserstein Distance

The Wasserstein Distance (Sometimes called Earth Mover's distance) is a tool proposed to compare probability measures and distributions on $\mathbb{R}^d$ through the knowledge of a metric on $\mathbb{R}^d$. The general definition is the following:

Given an exponent $p \geq 1$, the definition of the p-Wasserstein distance reads: For p $\in [1, \infty)$ and Borel probability measures P,Q on $\mathbb{R}^d$ with finite p-moments, their p-Wasserstein distance is

$$W_p(P,Q) = \left( \inf_{\pi \in \Gamma(P,Q)} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||X - Y||^p d\pi \right)^{1/p}, \tag{4.66}$$

where $\Gamma(P,Q)$ is the set of all joint probability measures on $\mathbb{R}^d \times \mathbb{R}^d$ whose marginals are P,Q, i.e. such that all subsets $A \in \mathbb{R}^d$ we have $\pi(A \times \mathbb{R}^d) = P(A)$ and $\pi(\mathbb{R}^d \times A) = Q(A)$ [30].

Given two 1D probability density functions and $p = 1$ (Expected value), the Wasserstein distance between the distributions is

$$W_1(P,Q) = \inf_{\pi \in \Gamma(P,Q)} \int_{\mathbb{R} \times \mathbb{R}} |x - y| d\pi(x,y), \tag{4.67}$$

where $\pi(x,y)$ is a coupling or transport plan but can also be viewed as a joint distribution that has $P$ and $Q$ as its marginals. The $|x - y|$ is a cost function corresponding to the Euclidean distance between point $x$ coming from the support of the $P$ distribution and $y$ coming from the support of the $Q$ distribution [31].

In other words, if we have two measures $P$ and $Q$ supported by $\mathbb{R}$, the Wasserstein distance is computed as a minimization process over all transport plans integrating all products of the distance between coupled points and the amount of mass that needs to be transported.

### 4.5.4 $\chi^2$ test

$\chi^2$ or chi-square is a commonly used quantity to test whether a given set of data is well described by a hypothesized function. Such a determination is called a chi-square test for goodness of fit. If $\nu$ independent variables $x_i$ are each normally distributed with mean $\mu_i$ and variance $\sigma_i^2$, then the quantity

known as $\chi^2$ is defined as follows:

$$\chi^2 = \sum_{i=1}^{\nu} \frac{(x_i - \mu_i)^2}{\sigma_i^2}. \tag{4.68}$$

If $\chi^2/DoF = 1$, where $DoF$ corresponds to a number of bins, only statistical fluctuations are present. If $\chi^2/DoF > 1$, also systematic differences in the distributions are measured [32].

# Part II

# Methodology

# Chapter 5

# Data preprocessing

Before performing an analysis in high energy physics (HEP), the data needs to be preprocessed. The goal of this chapter is to describe how to transform the output of the Monte Carlo simulations to input acceptable by machine learning frameworks. The explanation will start with an overview of methods and libraries that are needed for this purpose and the rest of the chapter will be aimed at the pre-processing itself.

## 5.1 Tools and libraries

### 5.1.1 GitHub

The GitHub platform was used as a version control system for the whole thesis. The current state of the work can be found at `https://github.com/LukVic/vae-generator-for-particle-physics`.

### 5.1.2 ROOT Framework

ROOT [33] is the object-oriented framework developed to solve the challenges of HEP. A typical application is processing both real and simulated data

consisting of many events with the same structure. For the purpose of this thesis, we are particularly interested in the way how data is stored within the framework. ROOT uses vertical data partitioning of arbitrary user-defined objects, implemented in a structure called TTree. TTrees are then partitioned into branches (TBranch) that can be accessed independently during the reading process due to separate buffers allocated for every branch. The leaves of this tree-like structure correspond to histograms of individual observables that can usually represent basic physical quantities like invariant mass or transverse momentum but also more sophisticated ones related to the geometry of the detector.

In our analysis, ROOT trees are stored separately for each year of data taking and for each reaction. The first task is therefore to extract the data from the root n-tuples to representation better suitable for machine learning applications.

### 5.1.3 ROOT RDataFrame

To efficiently transform the ROOT n-tuples TTree structure into a more suitable tabular format for use with Pandas DataFrame, the ROOT RDataFrame can be employed. RDataFrame is a high-level framework designed to analyze data produced by HEP experiments like LHC. A common struggle at the beginning of the analysis is to properly load the data that can be significantly larger than that of the available memory. The framework aims at this issue and efficiently aggregates the given entries. It means that the process of transformation, filtering, and final storage of the data can be done altogether without intermediate substeps. More details about the features of the framework can be found in [34].

### 5.1.4 Pandas

Pandas is a Python language library used for data analysis and manipulation. It offers data structures and operations for manipulating numerical tables and time series. The most important data structure for data analysis and machine learning is DataFrame.

A DataFrame consists of rows and columns. Each row typically represents a different observation or record, while each column represents a different

attribute or variable. Rows and columns are labeled for easy access and manipulation. It can be heterogeneous which means columns can contain different data types. It is also compatible with other Python libraries such as NumPy, Matplotlib, and SciPy [35].

## 5.2 Data Conversion

In this section, we will continue explaining the origin and structure of the used data. As already mentioned in section **??**, the data contains events from five reactions: $tbH^+$, $ttH$, $ttW$, $ttZ$, and $tt$, where $tbH^+$ corresponds to the signal and the rest to the background. All of them have a final state belonging to analysis channel $2lSS + 1\tau$. Table 5.1 summarizes the description.

| Name | Description | class |
|---|---|---|
| $tbH^+$ | top quark, bottom quark, charged Higgs boson | Signal |
| $ttH$ | top quark, anti-top quark, Higgs boson | Background |
| $ttW$ | top quark, anti-top quark, W boson | Background |
| $ttZ$ | top quark, anti-top quark, Z boson | Background |
| $tt$ | top quark, anti-top quark | Background |

**Table 5.1:** Summary of the reactions used for the analysis.

Next, let us have a look at how the data is represented. The data coming from the MC simulation is stored on the CERN grid servers in directories separated by the year of the run and then by type of reaction. The file IDs are shown in Table 5.2.

| Reaction | ID | Year |
|---|---|---|
| $tbH^+$ 800 GeV | $510375_{AF}$ | mc16a, mc16d, mc16e |
| $ttH$ | 346343, 346344, 346345 | mc16a, mc16d, mc16e |
| $ttW$ | 700168, 700205 | mc16a, mc16d, mc16e |
| $ttZ$ | 700309 | mc16a, mc16d, mc16e |
| $tt$ | 410470 | mc16a, mc16d, mc16e |

**Table 5.2:** ROOT n-tuples IDs corresponding to particular reactions. The year column indicates directories in which the particular dataset IDs can be found.

Each file represents an n-tuple filled with a certain number of entries that can be viewed as histograms of observables. For machine learning purposes,

it is needed to transform the n-tuple to data format with exactly defined rows and columns. The entries (events) have identical structures and are described by the same observables. Therefore, we can assign the events to be stored in rows as data samples described by observables defining the columns. Application of channel means dropping events or rows from the table. RDataFrame can do two steps at once, transform the n-tuple to columnar format, and filter unwanted events providing us with a data format conveniently prepared for Pandas DataFrame. the simplified pre-selection formula reads as

$$l2SS1tau \ \& \ nJets\_OR > 3 \ \& \ nJets\_OR\_DL1r\_70 > 0, \tag{5.1}$$

which is putting a restriction on three observables stored in the n-tuples. The full formula is shown in Appendix B. After applying the pre-selection criteria, only a fraction of the original events remain. Table 5.3 shows how many events remain after pre-selection with the corresponding percentage.

| **Reaction** | **Σ simulated** | **Σ pre-selected** | % |
|---|---|---|---|
| $tbH^+$ 800 GeV | 132130 | 10449 | 7.908 |
| $ttH$ | 1477351 | 15567 | 1.054 |
| $ttW$ | 730528 | 3187 | 0.436 |
| $ttZ$ | 2532296 | 8819 | 0.348 |
| $tt$ | 494461 | 38 | 0.008 |
| All | 5366766 | 38060 | 0.709 |

**Table 5.3:** Number and percentage of remaining events after preselection.

The reaction datasets belonging to the background are then concatenated together to provide one unified background dataset. This means we will need a separate trained VAE model for signal and background datasets.

# Chapter 6

# Classification

This chapter describes the practical aspects of the classification process used in the thesis which is used for the final evaluation of the generated data. The first part of the chapter aims to explain the implementations of the used classifiers. The second part then outlines the metrics implemented to assess the separation power of the classifier.

## 6.1 Data preparation

The data coming from the pre-processing part is separated into files based on corresponding reactions and is described by 71 features. For the generative purposes is this number too high since we would need to check each histogram individually. To avoid this complication we proceed as follows. At first, the separation is performed with all the features. Feature importance is computed afterward to determine which features strongly affect the classification results. Once we possess such a piece of information we choose the 10 most important features. The data distribution represented by these features is then learned by VAEs. The simulated dataset can be augmented at this point. This process is supported by [36] where the author noticed only small reduction in performance (5% on average) if 5 most important features are selected.

## 6.2 Classifiers

The theory behind the chosen classifiers was described in chapter 3 which discusses multilayer perceptrons and gradient-boosting decision trees. This section describes implementation details.

### 6.2.1 Multilayer perceptron

The data augmentation provides us with ability to significantly increase the size of the training dataset. It means that neural networks with a proportionally higher number of parameters and depth can be utilized. Keeping this in mind two architectures were chosen for the analysis. Their architectures are the following:

1. Parameters: 962

   $$\mathbf{x} \in \mathcal{X}^D \rightarrow \text{Linear}[D, 64] \rightarrow \text{BatchNorm} \rightarrow ReLU \rightarrow \text{Dropout}(0.1) \rightarrow \text{Linear}[64, M],$$

2. Parameters: 70146

   $$\mathbf{x} \in \mathcal{X}^D \rightarrow \text{Linear}[D, 256] \rightarrow \text{BatchNorm} \rightarrow ReLU \rightarrow \text{Dropout}(0.1) \rightarrow \text{Linear}[256, 256], \rightarrow \text{BatchNorm} \rightarrow ReLU \rightarrow \text{Dropout}(0.1) \rightarrow \text{Linear}[256,\ \text{M}].$$

Other important hyperparameters are:

- Optimizer: Adam
- Learning rate: 1e-3, 1e-4
- Number of Epochs: 1000
- Early stopping: 20
- Batch Size: 2048

### 6.2.2 XGBoost

The scikit-learn library implementation of XGBoost was used for the analysis. The following parameters can affect the performance the most:

- n_estimators: Specifies the number of decision trees to be boosted.
- max_depth: It limits how deep each tree can grow.
- min_child_weight: minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning.
- reg_alpha: Is the L1 regularization parameter, increasing its value makes the model more conservative.
- gamma: Is the regularization parameter for tree pruning. It specifies the minimum loss reduction required to grow a tree.
- learning_rate: Is a regularization parameter that shrinks feature weights in each boosting step.
- colsample_bytree: The algorithm will randomly select a subset of features on each iteration (tree).
- subsample: It represents the subsample ratio of the training sample.
- scale_pos_weight: his parameter is useful in case we have an imbalanced dataset, particularly in classification problems, where the proportion of one class is a small fraction of total observations.

We chose two XGBoost instances based on the solution found by the hyperparameter optimization framework Optuna [37]. Their parameters are:

1. XGB 0: default parameters
2. XGB 1:
   - n_estimators: 1102
   - learning_rate: 0.03
   - colsample_bytree: 0.9
   - gamma: 0.0005
   - subsample: 1.0
   - reg_alpha: 0.0029
   - reg_lambda: 0.00032

## 6.3 Evaluation Metrics

In the case of a binary classifier, the learned model predicts probabilities for every event $\mathbf{x}$ being assigned to signal $P(\mathbf{x} \in \mathcal{S})$ and background $P(\mathbf{x} \in \mathcal{B})$. This section discusses the output probabilities utilized for computing significance and explores how this metric relates to more familiar metrics commonly employed in the machine learning community.

Evaluation metrics are usually based on four building blocks called True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Their meaning is the following:

- TP: Is an outcome where the model correctly predicts the positive class.
- TN: Is an outcome where the model correctly predicts the negative class.
- FP: Is an outcome where the model incorrectly predicts the positive class.
- FN: Is an outcome where the model incorrectly predicts the negative class.

### 6.3.1 Weights

Significance, apart from other metrics like Accuracy or Precision needs to be in our case computed with the use of specific weights to reflect a true number of particles appearing in the detector. In practice, the validation data can be weighted based on the type of reaction ($tbH^+$, $ttH$, ...) or individually for each event. In this thesis, individual events are weighted for the purpose of analysis. To compute the weights the following formula is used:

$$w = \frac{Y(e_{f_0}) \cdot e_{f_1} \cdot e_{f_2} \cdot e_{f_3} \cdot e_{f_4} \cdot e_{f_5} \cdot e_{f_6} \cdot e_{f_7} \cdot e_{f_8} \cdot e_{f_9}}{e_{f_7}}, \tag{6.1}$$

where $w$ is the computed weight and $Y$ is the luminosity corresponding to the year of production defined as

$$Y(e_{f_0}) = \begin{cases} 36205.66, & \text{if } e_{f_0} = 2015 \ \vee e_{f_0} = 2016 \\ 44307, & \text{if } e_{f_0} = 2017 \\ 58450, & \text{if } e_{f_0} = 2018. \end{cases} \tag{6.2}$$

Particular factors $e_{f_i}$ from equation 6.1 are described in Table 6.1.

| Index | Feature name |
|---|---|
| $f_0$ | RunYear |
| $f_1$ | custTrigSF_LooseID_FCLooseIso_DLT |
| $f_2$ | weight_pileup |
| $f_3$ | jvtSF_customOR |
| $f_4$ | bTagSF_weight_DL1r_70 |
| $f_5$ | weight_mc |
| $f_6$ | xs |
| $f_7$ | totalEventsWeighted |
| $f_8$ | lep_SF_CombinedTight_0 |
| $f_9$ | lep_SF_CombinedTight_1 |

**Table 6.1:** Feature names corresponding to the IDs from the formula 6.1.

As already mentioned, the meaning of the weights is to obtain the correct number of particles corresponding to the real detected collision. For the purpose of the thesis, the $xs$ feature value was manually set to 0.1[1] for the signal since the value is unknown in reality. Moreover, weights have to be multiplied by filtering efficiency which is 40.3%[2] for tbH$^+$ 800 GeV. The final formula for the signal weight vector is

$$\mathbf{w}^s = \mathbf{w} \cdot \frac{0.1}{e_{f_0}} \cdot 0.403. \tag{6.3}$$

Table 6.2 shows the number of weighted events after pre-selection for all considered reactions.

| Reaction | Σ simulated | Σ pre-selected | Σ weighted |
|---|---|---|---|
| $tbH^+$ 800 GeV | 132130 | 10449 | 38.861 |
| $t\bar{t}H$ | 1477351 | 15567 | 12.649 |
| $t\bar{t}W$ | 730528 | 3187 | 9.046 |
| $t\bar{t}Z$ | 2532296 | 8819 | 12.561 |
| $t\bar{t}$ | 494461 | 38 | 4.605 |
| All | 5366766 | 38060 | 77.720 |

**Table 6.2:** Number of weighted remaining events after pre-selection for corresponding reactions.

[1] Representing a production crosssection of 0.1 pb.
[2] This percentage is the filter efficiency in the MC event generation.

### 6.3.2 Significance

The theoretical derivation of the significance was provided in section 3.3.3. The derived formulas are

$$Z_1 = \sqrt{2\left((S+B)\ln\left(1+\frac{S}{B}\right)-S\right)}, \tag{6.4}$$

$$Z_2 = \frac{S}{\sqrt{B}}. \tag{6.5}$$

As the output of the classification, we obtain probabilities for each event $\mathbf{x}$ belonging to either signal $\mathcal{S}$ or background $\mathcal{B}$ where $\mathcal{S} \cup \mathcal{B} \subseteq \mathcal{V}$ and $\mathcal{V}$ is the validation dataset. To perform the assignment we take all the events and decide where they belong based on a given threshold $\xi$. For single event $\mathbf{x}$ we can write $P(\mathbf{x} \in \mathcal{S}) \geq \xi \rightarrow \mathbf{x} \in \mathcal{S}$ and $P(\mathbf{x} \in \mathcal{S}) < \xi \rightarrow \mathbf{x} \in \mathcal{B}$. Now let us split the events based on their true labeling $\mathcal{V}_s \cup \mathcal{V}_b \subseteq \mathcal{V}$. Based on the chosen threshold a certain proportion of events from both of these subsets will be assigned to $\mathcal{S}$. We can define $S = \sum_{i=0}^{|\mathcal{V}_s \subset \mathcal{S}|-1} w_i$ and $B = \sum_{i=0}^{|\mathcal{V}_b \subset \mathcal{S}|-1} w_i$, where $w_i$ are weights defined in 6.3.1. These quantities can be plugged into equations 6.4 or 6.5. We can also state that $S = TP(\xi)$ and $B = FP(\xi)$.

Figure 6.1 shows the separate distributions for signal and background labeled testing events based on the output probabilities $P(\mathbf{x} \in \mathcal{S})$. Due to the binning on the output we can see correspondence with the definition 3.3.3 that applies on a binned histogram.

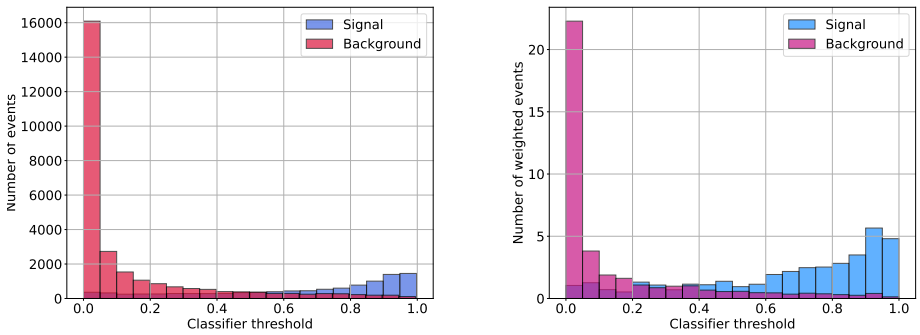



**Figure 6.1:** Left: Number of unweighted events for a given output probability. Right: Number of weighted events for a given output probability.

If $\xi = 0$ we obtain only true positives and false positives since for each $\mathbf{x}$ holds $P(\mathbf{x} \in \mathcal{S}) > 0$. Once we start increasing the $\xi$, the number of false positives decreases rapidly since a majority of true background events have a

low probability of being assigned as signal events (We are summing over bins of the weighted histogram from $\xi$ to 1). At a certain point, however, we start to detect false negatives since for higher thresholds signal events start to be misclassified as background. Figure 6.2 shows the summed weighted events from threshold to 1 and highlights the maximum w.r.t. the given significance formula. Four significance definitions are evaluated. $Z_1$ and $Z_2$ correspond to the definition 6.4 and 6.5. The most reliable result provides the $Z_1$ definition.

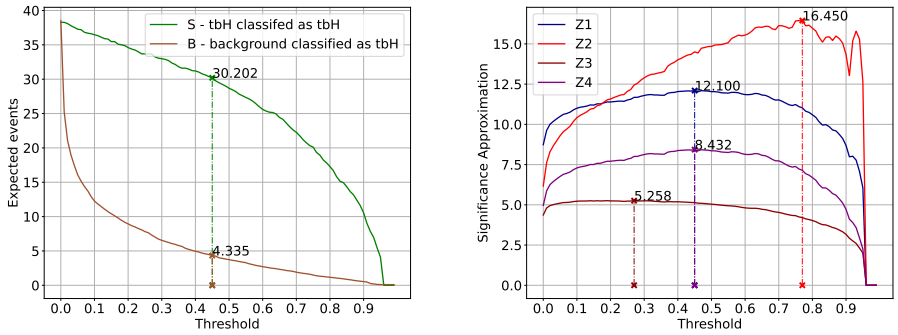



**Figure 6.2:** Left: Number of signal and background events for all considered thresholds. The best ratio is highlighted by a dashed line. A cut is applied if a number of weighted events is too low to prevent noticeable fluctuations of significance $Z_2$. Right: Four possible definitions of significance and their corresponding optima. $Z_1 : \sqrt{2\left((S+B)\ln\left(1+\frac{S}{B}\right)-S\right)}, Z_2 : S/\sqrt{B}, Z_3 : S/\sqrt{S+B}, Z_4 : S/\sqrt{B+3/2}$.

### 6.3.3 Accuracy

Accuracy is one of the standard evaluation metrics. Its formal definition is the following:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

For binary classifiers which is our case, the Accuracy can be defined in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is a good indicator of model performance. However, it can give misleading results for datasets with imbalanced classes.

### 6.3.4 Precision

By defining the Precision we are asking what proportion of positive identifications was actually correct. The formal definition is following:

$$\text{Precision} = \frac{TP}{TP + FP}.$$

From the definition, it can be inferred that the Precision metric depends only on the classification of the positive class. This attribute is common with (for us) the most important metric, significance, which was already discussed.

# Chapter 7

# Data augmentation

In chapter 4 we described the theory behind the models and learning algorithms that were implemented to perform data augmentation. This segment will be focused on the practical side of the matter. In the forthcoming sections, we will underline the implementation of the specific models and the associated challenges.

## 7.1 Standard models

In section 4.1 and 4.2 we described the theoretical basics behind the standard ELBO learning and SEL VAE respectively. The model consists of two neural networks, the encoder which transforms input data into latent representation, and the decoder giving the data back its original meaning. The parameters of the networks are then learned by ELBO maximization or by finding Nash equilibrium. Architectures were kept the same for both learning algorithms to ensure the objectivity of the results. The only difference was the use of Layer normalization in the case of ELBO and the Batch normalization in the case of SEL.

### 7.1.1 The encoder and decoder architecture

We chose the same architecture for both the encoder and decoder, with only one difference that will be explained later. The architecture is following:

$$\begin{aligned}\mathbf{x} \in \mathcal{X}^D &\to \text{Linear}[D, 4096] \to \text{LayerNorm} \to \mathit{ReLU} \\ &\to \text{Linear}[4096, 1024] \to \text{LayerNorm} \to \mathit{ReLU} \\ &\to \text{Linear}[1024, 2 \cdot M] \to \text{split} \to \boldsymbol{\mu} \in \mathbb{R}^M, \log \boldsymbol{\sigma}^2 \in \mathbb{R}^M,\end{aligned}$$

where $D$ is a number of features and $M$ is the size of the latent space. Each linear layer is coupled with batch normalization and ReLU activation function. The last layer output is split into two parts to produce $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}^2$ for $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ which is modeled as the normal distribution.

The decoder architecture is very similar up to one difference. The structure looks as follows:

$$\begin{aligned}\mathbf{z} \in \mathcal{Z}^M &\to \text{Linear}[M, 1024] \to \text{LayerNorm} \to \mathit{ReLU} \\ &\to \text{Linear}[1024, 4096] \to \text{LayerNorm} \to \mathit{ReLU} \\ &\to \text{Linear}[4096, 2 \cdot D + 1] \to \text{split} \to \boldsymbol{\mu} \in \mathbb{R}^D, \log \boldsymbol{\sigma}^2 \in \mathbb{R}^D, \text{sigmoid}(\mathbf{p}) \in [-1, 1].\end{aligned}$$

The structure looks very similar except for parameters being produced for $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ distribution. The difference is hidden in the $\mathbf{p}$ parameter which is a parameter of the Bernoulli distribution. If we want to learn a different distribution than the normal one the decoder is the place where the parameters need to be separated. It is also important to mention that the loss function needs to be enhanced by the Binary cross-entropy term. The approach would be very similar for categorical features as for the binary ones since we need to one-hot encode them.

### 7.1.2 ELBO Loss function

To compute ELBO based on the theory, we require a prior distribution $p(\mathbf{z})$, approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Prior distribution is in our case standard normal distribution therefore sampling from it is straightforward. The encoder and decoder neural networks learn the parameters for the approximate posterior and likelihood, which are then utilized to construct the respective distributions. A reparametrization trick needs to be used to allow backpropagation through the sampling part of the algorithm. The order of actions looks as follows:

$$\begin{aligned}\mathbf{x} \in \mathcal{X}^D &\to \text{Encode}(\mathbf{x}) \to q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \to \text{RSample} \to \mathbf{z} \in \mathcal{Z}^M \to \text{Decode}(\mathbf{z}) \\ &\to p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}),\end{aligned}$$

where RSample stands for reparametrized sampling. At this point all the necessary components of ELBO are accessible and we can easily compute $D_{KL}(p(\mathbf{z})||q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))$ and $\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. For the binary part of the decoder output, binary cross-entropy is used and added to the common loss formula.

### 7.1.3 SEL Loss function

The algorithm consists of two steps, thus the loss function is separated into two parts that are learned separately. the first one reads as follows:

$$\mathbf{x} \in \mathcal{X}^D \rightarrow \text{Encode}(\mathbf{x}) \rightarrow q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \rightarrow \text{Sample} \rightarrow \mathbf{z} \in \mathcal{Z}^M \rightarrow \text{Decode}(\mathbf{z}) \rightarrow p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}),$$

which is very similar to ELBO learning. There is just one difference which is an omission of parametrization for sampling since the expected value does not depend on $\boldsymbol{\theta}$. In the case of discrete distributions, this step also contains an additional binary cross-entropy term.

The second step is similar:

$$\mathbf{z} \in \mathcal{Z}^M \rightarrow \text{Decode}(\mathbf{x}) \rightarrow p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \rightarrow \text{Sample} \rightarrow \mathbf{x} \in \mathcal{X}^D \rightarrow \text{Encode}(\mathbf{x}) \rightarrow q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}),$$

this time samples are coming from likelihood and approximate posterior is learned. At this point, the log-likelihoods for the losses can be easily computed.

### 7.1.4 Observations

- Smaller size of the latent space $\mathcal{Z}^M$ improves the learning capabilities.
- It is better to build architecture with a lower number of layers but a high number of neurons.
- Layer normalization leads to better stability compared to Batch normalization for ELBO learning.
- Learning of Symmetric learning algorithm is more robust and allows higher learning rates.

## 7.2 Ladder models

The theory behind LVAE with ELBO learning is explained in section 4.3

### 7.2.1 Architecture

The architecture of the Ladder VAE is significantly more complex than that of the standard VAE. In the case of two-level LVAE shown in Figure 7.1 every arrow corresponds to a neural network, meaning already six neural networks are needed. It was already mentioned in the theoretical part that architecture can be split into two parts. The first part is the bottom-up deterministic path and is shown on the left in Figure 7.1. It consists of two deterministic neural networks with outputs $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{Y}^{D_2}$ and two networks that we could call encoders that produce parameter shifts for learned distribution. The second part is called top-down and contains two networks. The upper one resembles encoder in architecture and is based on $\mathbf{z}_2$ samples. it supplies parameters for $\mathbf{z}_1$ sampling. The lower network corresponds to the decoder in architecture.

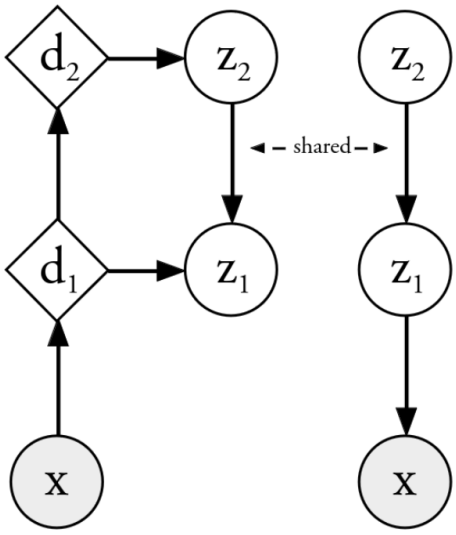



**Figure 7.1:** Schematic visualization of LVAE architecture.

The architectures of encoder and decoder networks is kept the same as for the standard VAEs. The deterministic networks are new in this case and

their architecture is following:

$$\begin{aligned}\mathbf{x} \in \mathcal{X}^{D_1} &\to \text{Linear}[D_1, 2048] \to \text{LayerNorm} \to \mathit{ReLU} \\ &\to \text{Linear}[2048, 1024] \to \text{LayerNorm} \to \mathit{ReLU} \\ &\to \text{Linear}[1024, D_2] \to \text{split} \to \mathbf{y} \in \mathcal{Y}^{D_2}.\end{aligned}$$

### 7.2.2 ELBO loss function

The loss function of LVAE is modeled similarly to the case of standard 7.1.2. However, the path to the goal is slightly longer. The chain of deterministic steps looks as follows:

$$\mathbf{x} \in \mathcal{X}^{D_1} \to \text{Transform}(\mathbf{x}) \to \mathbf{y}_1 \in \mathcal{Y}^{D_2} \to \text{Transform}(\mathbf{y}) \to \mathbf{y}_2 \in \mathcal{Y}^{D_2}.$$

At this point the algorithm branches to produce parameters for $q_{\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{x})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x})$. We proceed with

$$\begin{aligned}\mathbf{y}_2 \in \mathcal{Y}^{D_2} &\to \text{Encode}(\mathbf{y}_2) \to \Delta\boldsymbol{\mu}_2, \Delta \log \boldsymbol{\sigma}_2^2 \\ \mathbf{y}_1 \in \mathcal{Y}^{D_2} &\to \text{Encode}(\mathbf{y}_1) \to \Delta\boldsymbol{\mu}_1, \Delta \log \boldsymbol{\sigma}_1^2.\end{aligned}$$

At this point, we have prepared the parameter shifts. To obtain $\mathbf{z}_2$ and $\mathbf{z}_1$ recomputation described in 4.3.3 is performed. The parameter shift could be explained by

$$\left.\begin{matrix}0, 1 \\ \Delta\boldsymbol{\mu}_2, \Delta \log \boldsymbol{\sigma}_2^2\end{matrix}\right\} q_{\boldsymbol{\phi}}(\mathbf{z}_2|\mathbf{x}) \to \text{RSample} \to \mathbf{z}_2 \in \mathcal{Z}^M$$

$$\left.\begin{matrix}\boldsymbol{\mu}_1, \log \boldsymbol{\sigma}_1^2 \\ \Delta\boldsymbol{\mu}_1, \Delta \log \boldsymbol{\sigma}_1^2\end{matrix}\right\} q_{\boldsymbol{\phi}}(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x}) \to \text{RSample} \to \mathbf{z}_1 \in \mathcal{Z}^M,$$

where the upper path shifts the standard normal distribution. The final step is then to compute the likelihood as

$$\mathbf{z}_1 \in \mathcal{Z}^M \to \text{Decode}(\mathbf{z}_1) \to p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1). \tag{7.1}$$

At this point, we have all the terms needed to construct the extended ELBO loss.

### 7.2.3 SEL loss function

The process again consists of two parts. This time we avoid detailed step-by-step explanation since it is similar to 7.2.2. Let us focus more on the idea

behind the approach.

At the beginning of the first step, we have our data $\mathbf{x}$. It is ten used to obtain $\mathbf{z}_1$ and $\mathbf{z}_2$ in the following way:

$\mathbf{x} \in \mathcal{X}^D \to q_\phi(\mathbf{z}_2|\mathbf{x}) \to$ Sample $\to \mathbf{z}_2 \in \mathcal{Z}^M \to q_\phi(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x}) \to$ Sample $\to \mathbf{z}_1 \in \mathcal{Z}^M$ $\to \{\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2\}$.

Since we have $\mathbf{x}$, $\mathbf{z}_1$ and $\mathbf{z}_2$ in our hands we can compute

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) = \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) + \log p_{\boldsymbol{\theta}}(\mathbf{z}_1|\mathbf{z}_2).$$

The second step is similar, but we start with $\mathbf{z}_0$ sampled from the prior distribution. The process is following:

$$p(\mathbf{z}_2) \to \text{Sample} \to \mathbf{z}_2 \in \mathcal{Z}^M \to p_{\boldsymbol{\theta}}(\mathbf{z}_1|\mathbf{z}_2) \to \text{Sample} \to \mathbf{z}_1 \in \mathcal{Z}^M$$
$$\to p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1) \to \text{Sample} \to \mathbf{x} \in \mathcal{X}^D \to \{\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2\}.$$

The learning phase then reads as

$$\log q_\phi(\mathbf{z}_1|\mathbf{x}) = \log q_\phi(\mathbf{z}_2|\mathbf{x}) + \log q_\phi(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x}).$$

### 7.2.4 SEL Implementation issues

Let us describe a complication that occurred during the implementation of the first part. Particularly, the second term in the first loss function 7.2 that reads

$$\log p_{\boldsymbol{\theta}}(\mathbf{z}_1|\mathbf{z}_2) \tag{7.2}$$

is included to learn the parameters of neural network producing parameters for $p_{\boldsymbol{\theta}}(\mathbf{z}_1|\mathbf{z}_2)$ distribution. Unfortunately, learning these parameters always led to divergence of the overall loss function. For smaller network architectures, the diverging tendencies were weaker but the best solution was to remove the term from the loss formula. Based on these complications it was decided to omit the Ladder SEL from evaluation experiments.

## 7.3 Generative pipeline

- The data is transformed into a tabular format where columns correspond to chosen observables (features) and rows correspond to simulated events.

- The chosen VAE model is trained on the prepared dataset. A separate VAE is prepared for signal and background.
- Using the learned model arbitrary number of new events can be generated.
- The quality of the newly generated data is assessed by both qualitative and quantitative metrices.

# Part III

# Experimental results

# Chapter 8

# VAEs data quality comparison

The idea behind the first part of the generated data quality assessment is an evaluation with well-known qualitative and quantitative metrices. Data from all three models is evaluated and compared at the end. The final evaluation is performed using the best-scoring model, which is consequently employed for the analysis of charged Higgs boson separation.

## 8.1 Feature selection

Section 6.1 mentions that the number of features used for the data augmentation has been reduced from 71 to 10 which significantly helps with data quality evaluation and the decrease in performance is negligible. This section describes the process in more detail.

### 8.1.1 Feature importance

It is important to point out that it is complicated to choose $n$ the most important features, where $n$ stands for an arbitrary number of features. Importance is usually significantly different for GBDTs and MLPs. It also depends on chosen hyperparameters and architecture. Our objective was to select features that were frequently evaluated as highly important, but also

ensure that at least one feature is non-real valued. The approach was the following:

- MLP and XGBoost were used as classifiers.
- From the simulated dataset 0.2, 0.5 and 1.0 fractions were extracted.
- 20 different seeds were used for extraction. It means 20 different datasets for each extracted fraction.
- Feature importance was computed for all datasets.
- Features were ranked by achieved position in each run.
- Individual scores were summed to obtain overall results.
- Features that obtained the highest score were chosen.

To compute feature importance for the MLP, gradients of the output w.r.t. the input features were calculated. Once the MLP model is learned, the gradient of the output is computed with respect to the particular feature. The gradients are averaged over all data instances. The features with larger corresponding gradient values are the most important [38].

Regarding the XGBoost, feature importance for a single decision tree is determined by the improvement in the performance achieved by the split point of each attribute, weighted by the number of observations done by the node. It can also be interpreted as how much splitting on each feature allows to reduce the impurity across all the splits in the tree. The feature importances are then averaged across all of the decision trees [39]. The scikit-learn function was used to compute feature importance.

The list of chosen features is:

- total_charge: The sum of the electric charges of all light leptons in the event.
- MtLepMet: The transverse mass of a lepton and the missing transverse energy vector.
- DRll01: The $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ distance between the two leading leptons.
- HT_lep: The scalar sum of the transverse momenta of all leptons in the event.

- MLepMet: The invariant mass of a lepton and the missing transverse energy vector.
- jets_pt_0: The transverse momentum of the leading jet.
- taus_pt_0: The transverse momentum of the leading tau.
- met_met: The missing transverse energy in the event.
- minDeltaR_LJ_0: The minimum $\Delta$R distance between the leading lepton and a jet in the event.
- HT: The scalar sum of the transverse momenta of all jets in the event.

## 8.2 Generation time

In this experiment, we measured how long it takes to generate a certain amount of data. Since the time to generate each sample is constant we expect $\mathcal{O}(n)$ time complexity. Figure 8.1 shows the experimental results. It can be inferred that $10^7$ samples can be generated in $\approx 26$ seconds. After this number of samples, the generative process started to be very memory-demanding. Still, from the obtained data we can extrapolate the run time even for the higher number of samples.

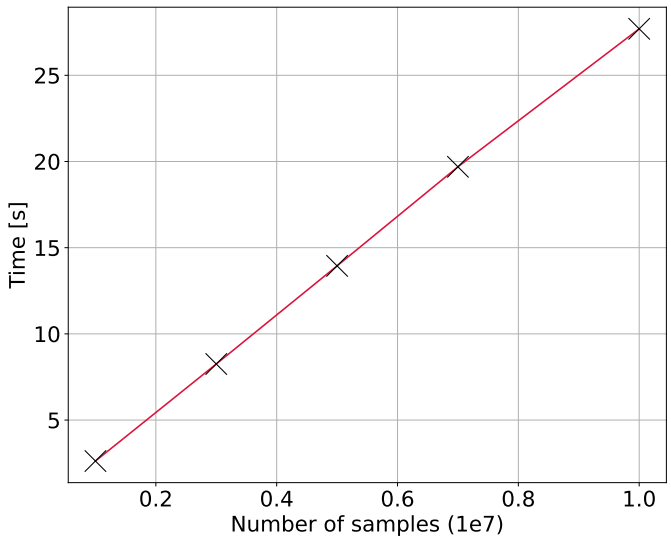



**Figure 8.1:** Processing time as a function of generated data.

## 8.3 Used metrics

To evaluate the data quality, a variety of metrices were implemented. These metrics can be categorized as follows:

- Joint: The metrics evaluate the dataset as a whole.
- Marginal: The distribution of the dataset is marginalized into separated feature histograms which are subsequently evaluated.
- Qualitative: The data is evaluated subjectively by the search for patterns in usually visualized output.
- Quantitative: Precise numbers are output to accurately assess the result.

### 8.3.1 Feature histograms

This method is straightforward. After the data is generated, individual feature vectors are used to plot histograms. For this experiment, the same number of events is generated as present in the simulated dataset. Normalized histograms of simulated and generated data are then plotted together to visually compare the shapes. For each model, the three randomly chosen features, as well as the binary feature, are shown. Figures 8.2 and 8.3 show the signal and background features comparisons. The rest can be found in appendices A.1 and A.2.

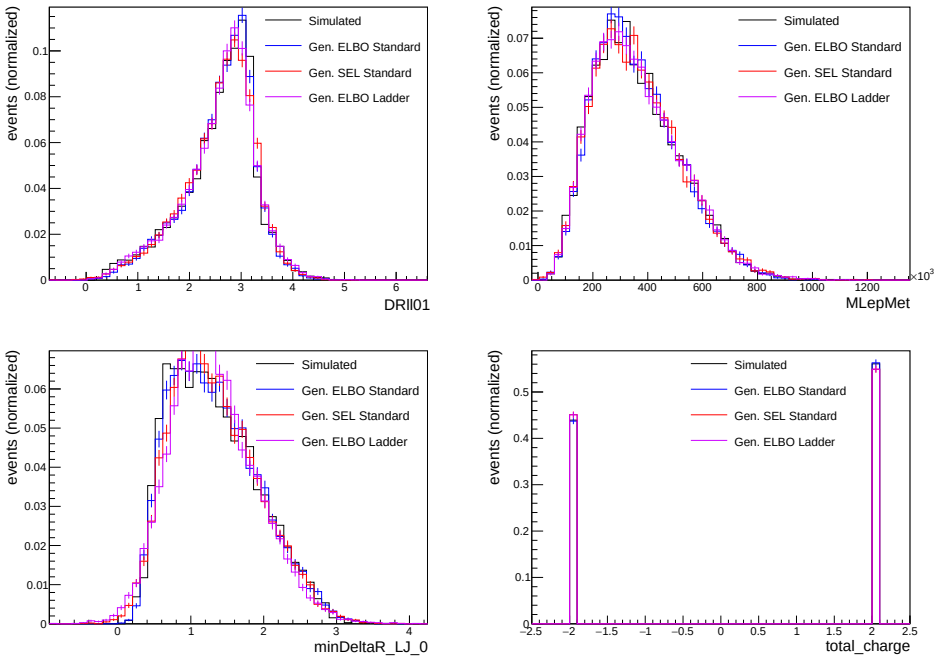



**Figure 8.2:** Comparison of four chosen signal features for all the models.

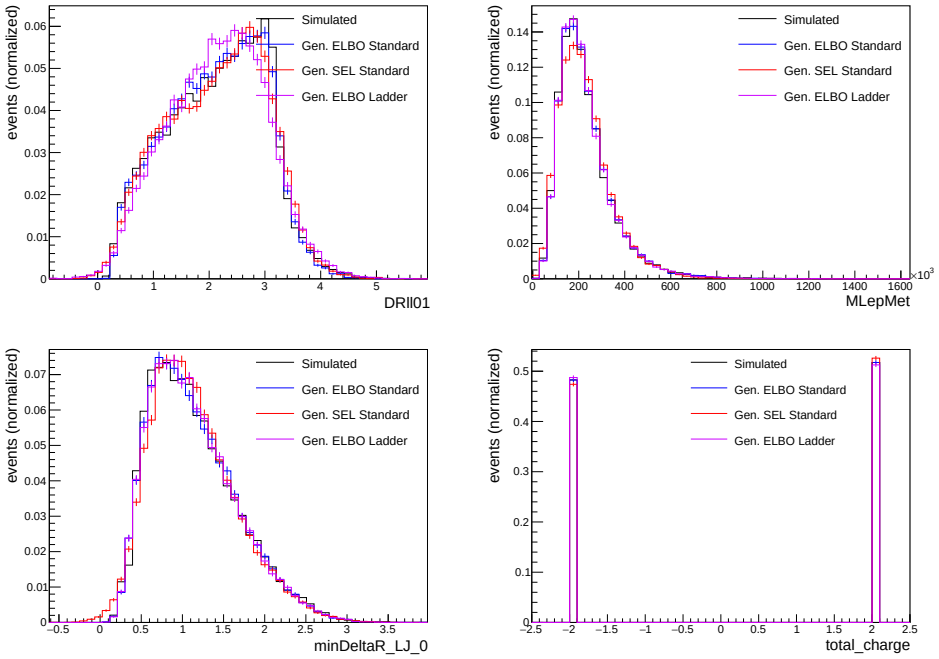



**Figure 8.3:** Comparison of four chosen background features for all the models.

### 8.3.2 $\chi^2$ test

$\chi^2$ test is a metric well-known in particle physics for evaluating 1D feature histograms. The goal is to compare the histogram of the simulated feature with the generated one. The theory behind the metric is described in section 4.5.4. The results for signal histograms are summarized in Table 8.1. The results for the background histograms are shown in Table 8.2. The ROOT framework function [40] was used to produce the results.

| Feature name | Std. ELBO | Std. SEL | Ldr. ELBO |
|---|---|---|---|
| $\text{taus}_{\text{pt}_0}$ | 1.491 | 12.158 | 1.625 |
| MtLepMet | 1.133 | 1.24 | 1.126 |
| $\text{met}_{\text{met}}$ | 1.595 | 1.515 | 1.46 |
| DRll01 | 3.528 | 3.744 | 2.838 |
| MLepMet | 1.431 | 1.432 | 0.994 |
| $\text{minDeltaR}_{\text{LJ}_0}$ | 1.925 | 4.438 | 8.491 |
| $\text{jets}_{\text{pt}_0}$ | 1.249 | 2.505 | 0.941 |
| HT | 1.728 | 1.683 | 0.976 |
| $\text{HT}_{\text{lep}}$ | 1.892 | 2.07 | 0.86 |
| $\text{total}_{\text{charge}}$ | 0.103 | 2.928 | 2.744 |

**Table 8.1:** Chosen features with corresponding $\chi^2$ values for the three generative models (signal sample).

| Feature name | Std. ELBO | Std. SEL | Ldr. ELBO |
|---|---|---|---|
| $\text{taus}_{\text{pt}_0}$ | 1.638 | 5.320 | 8.148 |
| MtLepMet | 1.112 | 2.555 | 1.269 |
| $\text{met}_{\text{met}}$ | 1.078 | 1.959 | 1.255 |
| DRll01 | 1.572 | 3.859 | 2.291 |
| MLepMet | 1.066 | 2.293 | 1.226 |
| $\text{minDeltaR}_{\text{LJ}_0}$ | 1.344 | 2.203 | 6.799 |
| $\text{jets}_{\text{pt}_0}$ | 0.997 | 2.684 | 1.410 |
| HT | 0.793 | 1.870 | 0.927 |
| $\text{HT}_{\text{lep}}$ | 1.027 | 2.444 | 1.002 |
| $\text{total}_{\text{charge}}$ | 0.144 | 1.473 | 0.152 |

**Table 8.2:** Chosen features with corresponding $\chi^2$ values for all three generative models (background sample).

On average, the standard ELBO performs the best. While the Ladder ELBO learning shows better results for some features, it performs significantly worse for a few others. From performed experiments, we could assess that this is probably caused by the interplay of two KL divergence terms. Results for

the *total_charge* feature are not necessarily very precise since it is computed only for 2 bins and $\chi^2$ can significantly fluctuate under such circumstances.

### 8.3.3 Wasserstein distance

The purpose of the Wasserstein distance is similar to that of the $\chi^2$ test but is used more broadly. The theory explaining the metric can be seen in section 4.5.3. Table 8.3 and 8.4 show the distance for all three models. The SciPy function [41] was used.

| Feature name | Std. ELBO | Std. SEL | Ldr. ELBO |
|---|---|---|---|
| $\text{taus}_{\text{pt}_0}$ | 4.309e-07 | 1.736e-07 | 4.221e-07 |
| MtLepMet | 1.206e-07 | 2.051e-07 | 1.539e-07 |
| $\text{met}_{\text{met}}$ | 1.031e-07 | 2.380e-07 | 2.442e-07 |
| DRll01 | 0.021 | 0.024 | 0.043 |
| MLepMet | 1.439e-07 | 2.180e-07 | 1.624e-07 |
| $\text{minDeltaR}_{\text{LJ}_0}$ | 0.027 | 0.062 | 0.013 |
| $\text{jets}_{\text{pt}_0}$ | 9.411e-08 | 5.127e-08 | 9.608e-08 |
| HT | 6.678e-08 | 1.057e-07 | 5.125e-08 |
| $\text{HT}_{\text{lep}}$ | 1.393e-07 | 1.539e-07 | 6.781e-08 |
| $\text{total}_{\text{charge}}$ | 0.0081 | 0.0085 | 0.0015 |

**Table 8.3:** Chosen features with corresponding Wasserstein distance values for all three generative models (Signal).

| Feature name | Std. ELBO | Std. SEL | Ldr. ELBO |
|---|---|---|---|
| $\text{taus}_{\text{pt}_0}$ | 3.884e-07 | 7.076e-07 | 7.278e-07 |
| MtLepMet | 2.118e-07 | 2.690e-07 | 1.052e-07 |
| $\text{met}_{\text{met}}$ | 5.171e-08 | 2.750e-07 | 1.706e-07 |
| DRll01 | 0.013 | 0.028 | 0.034 |
| MLepMet | 2.122e-07 | 2.611e-07 | 1.093e-07 |
| $\text{minDeltaR}_{\text{LJ}_0}$ | 0.011 | 0.027 | 0.011 |
| $\text{jets}_{\text{pt}_0}$ | 6.325e-08 | 4.987e-08 | 4.304e-08 |
| HT | 4.102e-08 | 1.264e-07 | 1.746e-08 |
| $\text{HT}_{\text{lep}}$ | 3.060e-07 | 2.472e-07 | 1.116e-07 |
| $\text{total}_{\text{charge}}$ | 0.0023 | 0.0065 | 0.0028 |

**Table 8.4:** Chosen features with corresponding Wasserstein distance values for all three generative models (Background).

The results from this experiment are very similar to each other for the

three evaluated models. An interesting observation is a significantly higher score for $DRll01$ and $minDeltaR_{LJ_0}$ features.

### 8.3.4 Feature corelation matrix

The idea behind this experiment is the following. The features of the dataset can be analyzed by computing the correlation between any pair of features. The usually used correlation coefficients are Pearson, Spearman's Rank, Kendall Rank, and Point Biserial coefficient. In our case, we chose the Pearson correlation coefficient [42]. The correlation matrix is computed separately for the simulated and the generated dataset. The matrices are compared to assess whether the pairwise correlations remain consistent in the generated data. Figures 8.4 and 8.5 show results generated by standard ELBO learning for signal and background, respectively. Color closer to red indicates a high correlation. Color closer to blue indicated a lower or negative correlation. The remaining results can be found in appendix A.3.

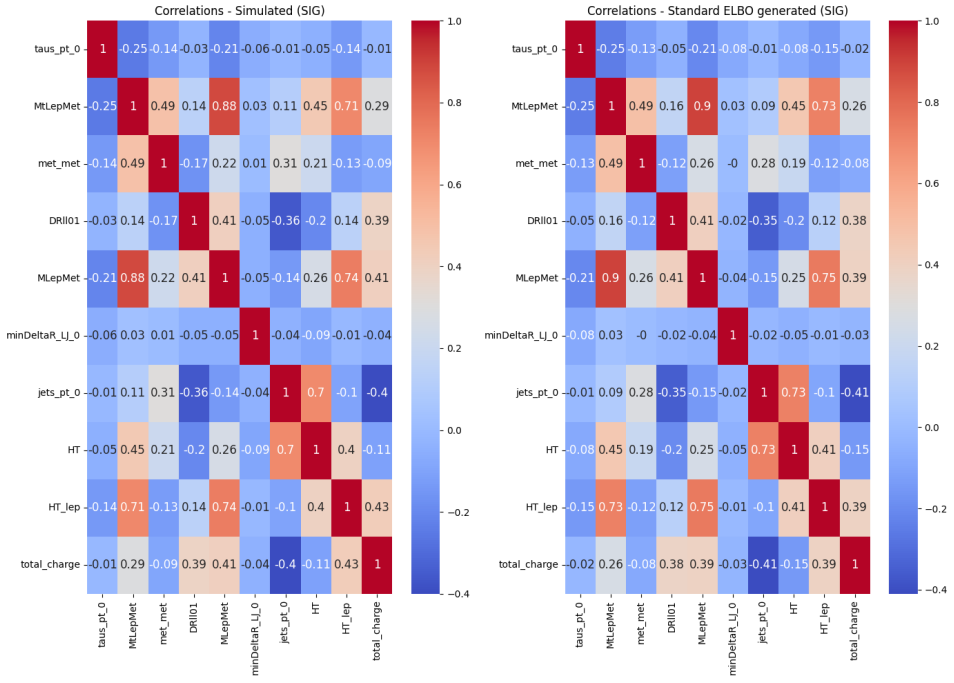



**Figure 8.4:** Feature correlation matrix for signal. Left: Simulated data, Right: Generated data by standard ELBO learning model (signal sample)

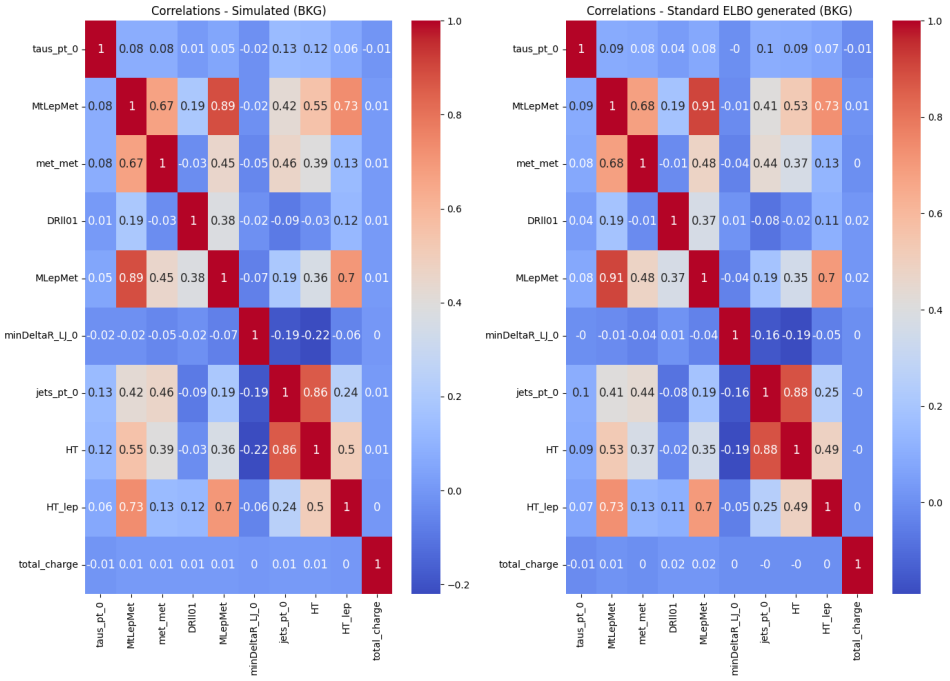



**Figure 8.5:** Feature correlation matrix for background. Left: Simulated data, Right: Generated data by standard ELBO learning model (background sample)

From the Figures, we can see good preservation of feature correlations. From a classification perspective, one interesting fact can be noted. The *total_charge* feature shows strong correlations with certain features in the signal sample but shows a very neutral correlation with all features in the background sample. This behavior may explain the high importance of the feature.

### 8.3.5 t-SNE visualization

The t-SNE algorithm is used to qualitatively compare the similarity of the whole dataset distributions. It is impossible to visually compare features in 10-dimensional space thus embedding into lower-dimensional space is needed. A deeper explanation of the theory behind the embedding is given in section 4.5.2. Figure 8.6 shows the comparison of simulated and generated data in the data space for 3000 samples. All the point clouds seem to be well aligned.

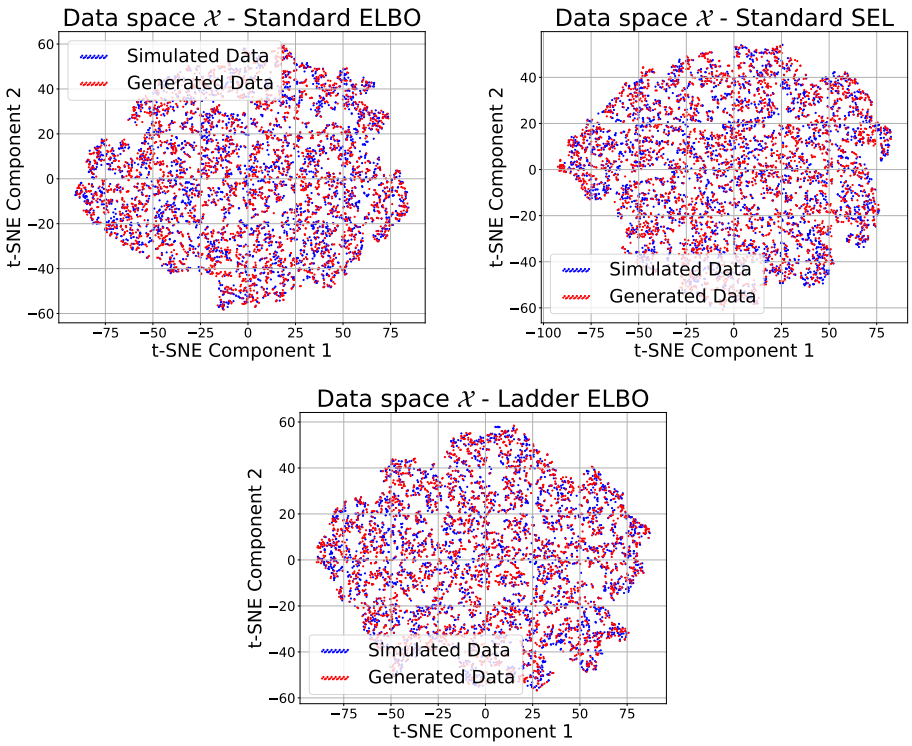



**Figure 8.6:** Comparison of simulated and generated data for all three implemented models.

### 8.3.6 Cross validation

This experiment aims to investigate the difference between the simulated and the generated data. Both datasets are split into training and validation subsets. The following experiments are performed for simple MLP with 962 parameters:

- Train: Simulated × Validate: Simulated
- Train: Simulated × Validate: Generated
- Train: Generated × Validate: Generated
- Train: Generated × Validate: Simulated

Table 8.5 shows the mean values after 20 repetitions. There is almost no difference between simulated and generated data by standard and Ladder

ELBO learning. The discrepancy is noticeable for the standard SEL which is expected since the algorithm focuses more on the balance between latent and data space.

| **Standard ELBO** | **Simulated validate** | **Generated validate** |
|---|---|---|
| Simulated train | 0.862 | 0.861 |
| Generated train | 0.860 | 0.864 |
| **Standard SEL** | **Simulated validate** | **Generated validate** |
| Simulated train | 0.862 | 0.841 |
| Generated train | 0.853 | 0.856 |
| **Ladder ELBO** | **Simulated validate** | **Generated validate** |
| Simulated train | 0.862 | 0.859 |
| Generated train | 0.858 | 0.862 |

**Table 8.5:** Cross validation results for all the three implemented models. Standard ELBO learning shows the best results.

### 8.3.7 Conclusion

Based on the performed experiments, the standard ELBO learning approach was chosen to perform the data augmentation experiment for the signal and background analysis.

# Chapter 9

# Augmented analysis

This chapter concludes the thesis by integrating data augmentation and data analysis.

## 9.1 Simulated data augmentation

To properly investigate the impact of data augmentation, the following steps were carefully considered:

- MLP and XGBoost classifiers were employed to examine the effect of the data augmentation on different models.
- First, models were trained on 0.2, 0.4, 0.6, 0.8, and 1.0 fractions of the simulated data to determine a baseline performance.
- 0.2, 0.4, 0.6, 0.8, and 1.0 were extracted from the generated dataset, which is five times larger than the simulated dataset.
- This process was repeated 20 times with different train-validation splits each time to reduce fluctuations in results.
- Mean values after 20 repetitions were visualized and standard deviation between them was noted.

In the following sections, MLP and XGBoost classifiers results are shown separately due to differences in the analysis process and results.

## 9.2 Gradient boosting decision tree

Two different configurations of XGBoost parameters were chosen for the final experiment (section 6.2.2). Four metrics are monitored to assess the effect of the data augmentation. Figure 9.1 shows precision on the left and accuracy on the right. The green band indicates simulated data and the purple region corresponds to generated data addition. Both matrices show a similar increasing tendency with added generated data.

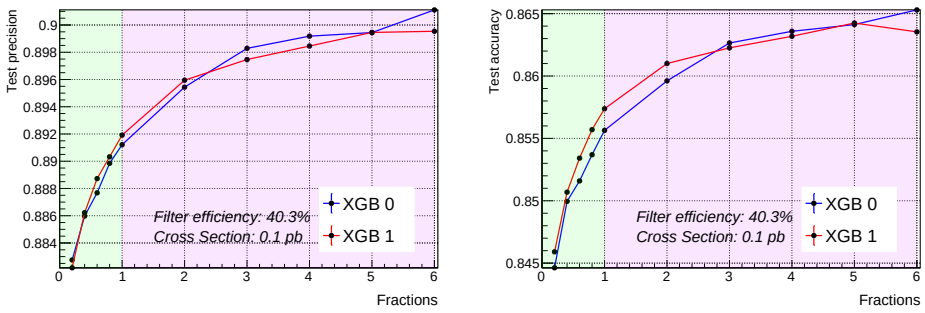



**Figure 9.1:** Left: Precision comparison for two XGBoost configurations. Standard deviation 0.0035 for 20 repetitions. Right: Accuracy comparison for two XGBoost configurations. Standard deviation 0.0034 for 20 repetitions.

The second pair of plots in Figures 9.2 shows the significance of $Z_1$ and $Z_2$ metrics. The XGB 1 classifier has optimized hyperparameters (section 6.2.2) therefore it saturates earlier but both classifiers achieve similar performance with a sufficient amount of generated data.

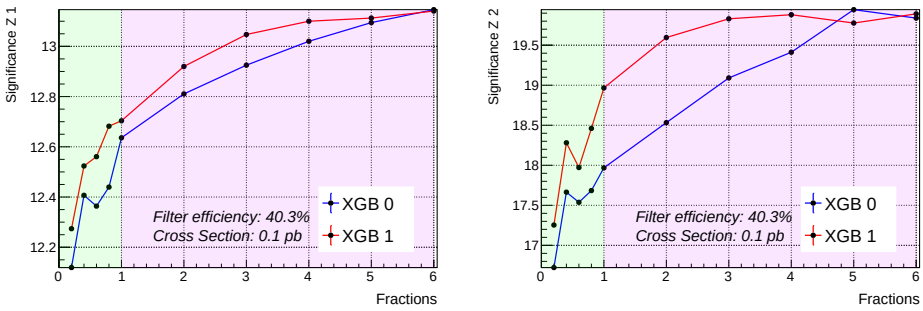



**Figure 9.2:** Left: Significance $Z_1$ comparison for two XGBoost configurations. Standard deviation 0.9009 for 20 repetitions Right: Significance $Z_2$ comparison for two XGBoost configurations. Standard deviation 2.7141 for 20 repetitions.

It is visible that the data augmentation causes improvement of all the measured quantities. The reason is probably a strong overfitting of the algorithm that is difficult to overcome. However, a higher volume of data helps with its reduction. This claim is supported by the performance measured on the training dataset shown in A.4.

## 9.3 Multilayer perceptron

In this section, we do the same comparison as in the previous one just with MLPs instead of XGBoost. Figure 9.3 shows results for precision and accuracy and Figure 9.4 shows results for the significance.

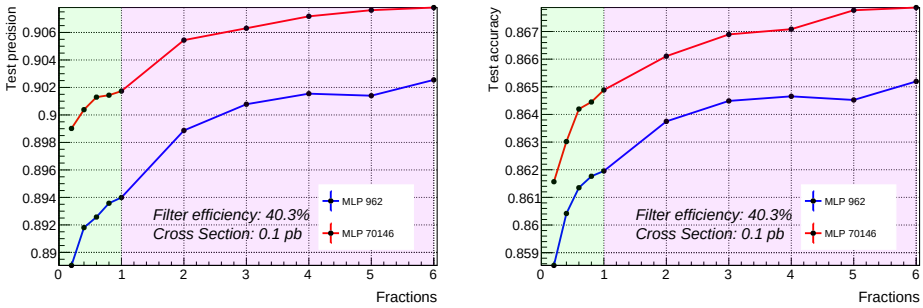



**Figure 9.3:** Left: The precision comparison for two MLP configurations. Standard deviation 0.0035 for 20 repetitions. Right: The accuracy comparison for two MLP configurations. Standard deviation 0.0038 for 20 repetitions.

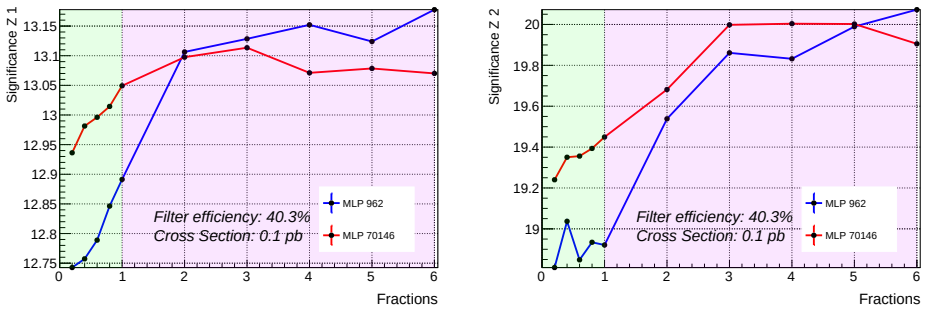



**Figure 9.4:** Left: Significance $Z_1$ comparison for two MLP configurations. Standard deviation 0.9233 for 20 repetitions Right: Significance $Z_2$ comparison for two MLP configurations. Standard deviation 2.9086 for 20 repetitions.

The results are not as good as the ones presented in the previous section. Precision and accuracy slightly increase but significance stays almost constant. We can also see that the difference between 0.2 and 1.0 of simulated data is negligible. This time overfitting is not present which can be the reason for a reduction in improvements A.5. The best-achieved scores are however very similar to XGBoost which could indicate certain performance boundaries given by the expressiveness of either simulated or generated data.

## 9.4 Generated data only

This section demonstrates the use of only generated data for chosen XGBoost and MLP implementations. Figures 9.5 and 9.6 show the result. Standard deviations between runs were similar to ones from the previous sections.

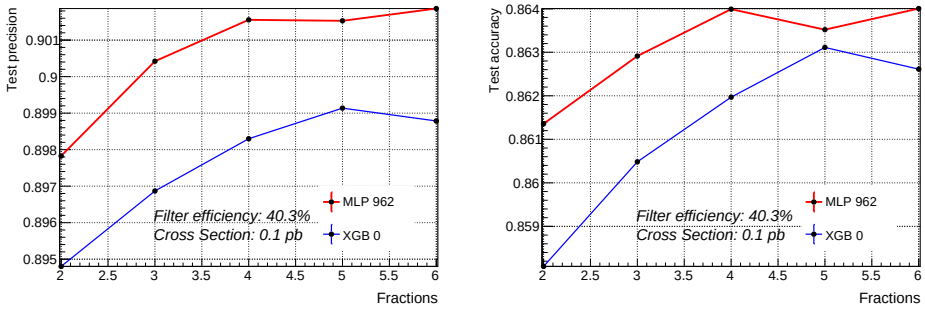



**Figure 9.5:** Left: The precision comparison for basic XGBoost and MLP with 962 parameters. Right: The accuracy comparison for basic XGBoost and MLP with 962 parameters.

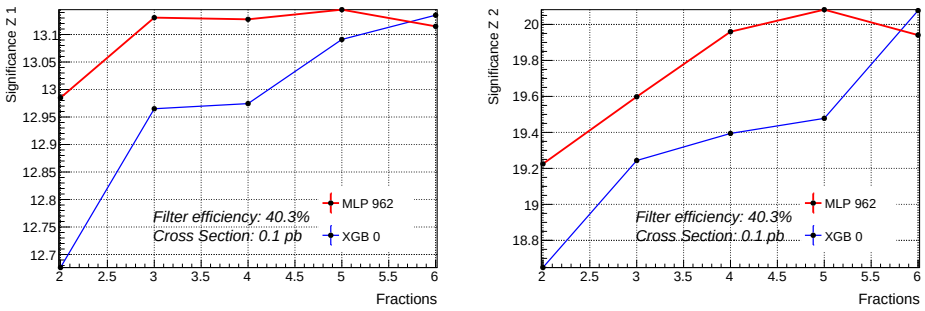



**Figure 9.6:** Left: Significance $Z_1$ comparison for basic XGBoost and MLP with 962 parameters. Right: Significance $Z_2$ comparison for basic XGBoost and MLP with 962 parameters.

The results suggest that the performance closely resembles that achieved by a combination of simulated and generated data. Hence, there is no need to include simulated data to improve performance. Simulated data can be used to cross-check and estimate the related systematic uncertainties by using data augmentation.

# Chapter 10

# Conclusion

A method for the augmentation of charged Higgs boson simulated data belonging to the analysis channel $2lSS + 1\tau_{had}$ based on a variational autoencoder model was developed. Two learning algorithms, Evidence lower bound maximization and Symmetric Equilibrium Learning were implemented with standard architectures. Effects of hierarchical architecture in the form of Ladder VAE were investigated. All three models were evaluated by individual features histograms, Wasserstein distance, $\chi^2$ test, t-SNE algorithm, and cross-validation. Standard ELBO learning showed the best results and the fastest learning.

The generator chosen by metrics evaluation was used to generate an artificial dataset defined by 10 most important features with a size five times larger than the simulated one. Multilayer perceptron and XGBoost algorithm configurations were trained on simulated, generated, and mixed data. The accuracy, precision, and significance metrics were used to monitor the performance of the classifier. It was observed that the generated data can increase the classifier performance and can to a certain extent replace the simulated data. In the future, the impact of systematic uncertainties on the data analysis needs to be studied.

# Bibliography


[1] ATLAS. *Cross section and luminosity*. May 2024. URL: https://cds.cern.ch/record/2800578/files/CrossSectionandLuminosity%5CPhysics%20CheatSheet.pdf.

[2] W. Herr and B. Muratori. "Concept of luminosity". In: (Nov. 2006). DOI: 10.5170/CERN-2006-002.361. URL: http://cds.cern.ch/record/941318.

[3] E. Daw. "Lecture 7 - Rapidity and Pseudorapidity". In: (Mar. 2012). URL: https://www.hep.shef.ac.uk/edaw/PHY206/Site/2012_course_files/phy206rlec7.pdf.

[4] CMS Wiki Pages. *cms_coordinate_system.png*. July 2017. URL: https://wiki.physik.uzh.ch/cms/_detail/latex:cms_coordinate_system.png?id=latex%3Atikz.

[5] CERN. *About CERN*. May 2024. URL: https://home.cern/about.

[6] CERN. *The Large Hadron Collider*. May 2024. URL: https://home.cern/science/accelerators/large-hadron-collider.

[7] ATLAS. "The Inner Detector". In: (May 2024). URL: https://atlas.cern/Discover/Detector/Inner-Detector.

[8] ATLAS Collaboration. *The ATLAS Experiment at the CERN Large Hadron Collider: A Description of the Detector Configuration for Run 3*. May 2023. arXiv: 2305.16623 [physics.ins-det].

[9] G. Weiglein. *Standard Model and Higgs Physics*. Sept. 2014. URL: https://www.desy.de/~weiglein/mlaach_smhiggs_14.pdf.

[10] K. Nguyen. *The Higgs Mechanism*. July 2009. URL: https://www.theorie.physik.uni-muenchen.de/lsfrey/teaching/archiv/sose_09/rng/higgs_mechanism.pdf.

[11] J. Eysermans, M. I. P. Morales, and on behalf of the CMS Collaboration. "Charged Higgs Analysis in CMS". In: *Journal of Physics: Conference Series* 761.1 (Oct. 2016), p. 012030. DOI: 10.1088/1742-6596/761/1/012030. URL: https://iopscience.iop.org/article/10.1088/1742-6596/761/1/012030/pdf.

[12] Ch. M. Bishop and H. Bishop. *Deep Learning - Foundations and Concepts*. Ed. by Springer Cham. 1st ed. 2023. ISBN: 978-3-031-45468-4. DOI: https://doi.org/10.1007/978-3-031-45468-4.

[13] L. Vicenik. "Machine Learning for the Leptoquark Search Using CERN ATLAS Data". Presented Jun 2022. Prague, Tech. U, 2022. URL: https://cds.cern.ch/record/2812370.

[14] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014, pp. I–XVI, 1–397. ISBN: 978-1-10-705713-5. URL: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf.

[15] J. Brownlee. *A Gentle Introduction to Ensemble Learning Algorithms*. Apr. 2021. URL: https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/.

[16] J. Starmer. *Statquest An epic journey through statistics and machine learning*. May 2024. URL: https://statquest.org/video-index/.

[17] T. Chen and C. Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145%5C%2F2939672.2939785.

[18] R.J. Barlow. "Practical statistics for particle physics". en. In: *CERN Yellow Reports: School Proceedings* Vol. 5 (2020 2020), 12–25 September 2018. DOI: 10.23730/CYRSP-2020-005.149. URL: https://e-publishing.cern.ch/index.php/CYRSP/article/view/1124.

[19] Glen C. et al. "Asymptotic formulae for likelihood-based tests of new physics". In: *The European Physical Journal C* 71.2 (Feb. 2011). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-011-1554-0. URL: http://dx.doi.org/10.1140/epjc/s10052-011-1554-0.

[20] D. P. Kingma and M. Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (June 2019), pp. 307–392. ISSN: 1935-8245. DOI: 10.1561/2200000056. URL: http://dx.doi.org/10.1561/2200000056.

[21] B. Flach, D. Schlesinger, and A. Shekhovtsov. *Symmetric Equilibrium Learning of VAEs*. July 2024. arXiv: 2307.09883 [cs.LG].

[22] C. K. Sønderby et al. *Ladder Variational Autoencoders*. May 2016. arXiv: 1602.02282 [stat.ML].

[23] J. Morris. *Introduction to variational autoencoders*. Oct. 2021. URL: https://jxmo.io/posts/variational-autoencoders.

[24] L. Weng. *From Autoencoder to Beta-VAE*. Aug. 2018. URL: https://lilianweng.github.io/posts/2018-08-12-vae/.

[25] J. M. Tomczak. *Deep Generative Modeling*. Springer Nature, Feb. 2022. URL: https://jmtomczak.github.io/dgm_book.html.

[26] Michael I. J. *The exponential family: Basics*. May 2024. URL: https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter8.pdf.

[27] F. Castillo. *Reduction of Dimensionality: Top Techniques Overview – SNE, t-SNE, and UMAP*. Sept. 2023. URL: https://arize.com/blog-course/reduction-of-dimensionality-top-techniques/.

[28] G. E. Hinton and S. Roweis. "Stochastic Neighbor Embedding". In: *Advances in Neural Information Processing Systems*. Ed. by S. Becker, S. Thrun, and K. Obermayer. Vol. 15. MIT Press, Jan. 2002. URL: https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf.

[29] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (Nov. 2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[30] A. Ramdas, N. Garcia, and M. Cuturi. *On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests*. Sept. 2015. arXiv: 1509.02237 [math.ST].

[31] Z. Goldfeld. *Scaling Wasserstein Distances to High Dimensions via Smoothing*. Feb. 2021. URL: http://people.ece.cornell.edu/zivg/Scaling_Talk2021.pdf.

[32] P. Scott. *Chi-square: testing for goodness of fit*. May 2024. URL: http://maxwell.ucsc.edu/~drip/133/ch4.pdf.

[33] R. Brun and F. Rademakers. "ROOT — An object oriented data analysis framework". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 389.1 (Apr. 1997). New Computing Techniques in Physics Research V, pp. 81–86. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(97)00048-X. URL: https://www.sciencedirect.com/science/article/pii/S016890029700048X.

[34] Piparo, D. et al. "RDataFrame: Easy Parallel ROOT Analysis at 100 Threads". In: *EPJ Web Conf.* 214 (Sept. 2019), p. 06029. DOI: 10.1051/epjconf/201921406029. URL: https://doi.org/10.1051/epjconf/201921406029.

[35] W. McKinney et al. "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. Jan. 2010, pp. 51–56. URL: http://conference.scipy.org.s3-website-us-east-1.amazonaws.com/proceedings/scipy2010/pdfs/mckinney.pdf.

[36] M. Rames. "Search for tbH$^+$($\tau\tau$) with Performance Optimisation for Signal and Background Separation Using Machine Learning with ATLAS Data". Presented Jun 2023. Prague, Tech. U., June 2023. URL: https://cds.cern.ch/record/2862250.

[37] Takuya A. et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 2019. URL: https://arxiv.org/abs/1907.10902.

[38] S. Kabir and L. Farrokhvar. "Non-Linear Feature Selection for Prediction of Hospital Length of Stay". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. Dec. 2019, pp. 945–950. DOI: 10.1109/ICMLA.2019.00162.

[39] E. K. Marsh. *Calculating XGBoost Feature Importance*. Jan. 2023. URL: https://medium.com/@emilykmarsh/xgboost-feature-importance-233ee27c33a4.

[40] L. Moneta N. Gagunashvili D. Haertl. *chi2test.C File Reference*. May 2024. URL: https://root.cern/doc/master/chi2test_8C.html.

[41] Scipy. *scipy.stats.wasserstein_distance*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wasserstein_distance.html.

[42] M. Stojiljković. *NumPy, SciPy, and pandas: Correlation With Python*. May 2024. URL: https://realpython.com/numpy-scipy-pandas-correlation-python/.

# Appendices

# Appendix A

# Generative model evaluation results

## A.1 Signal histograms

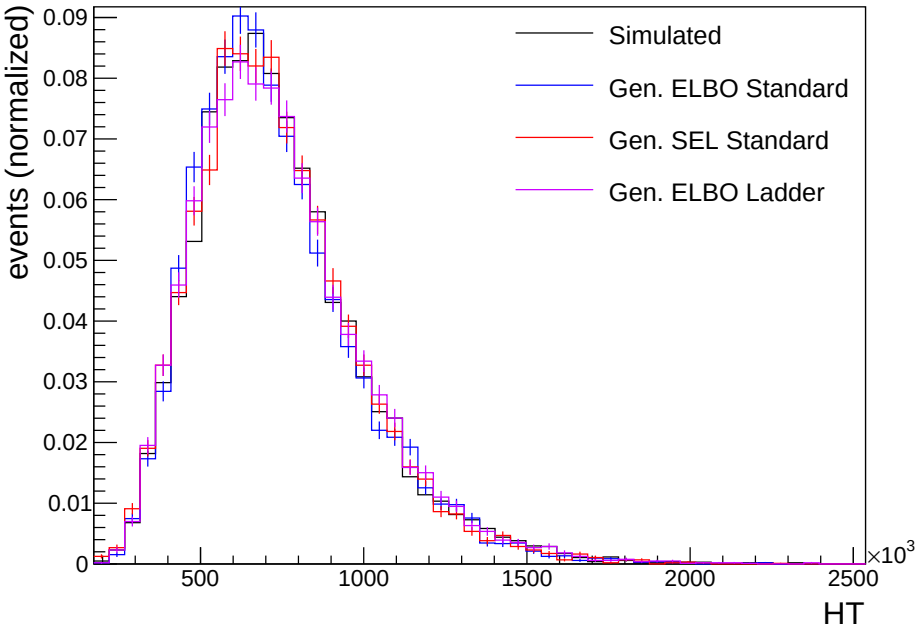



**Figure A.1:** Feature: HT

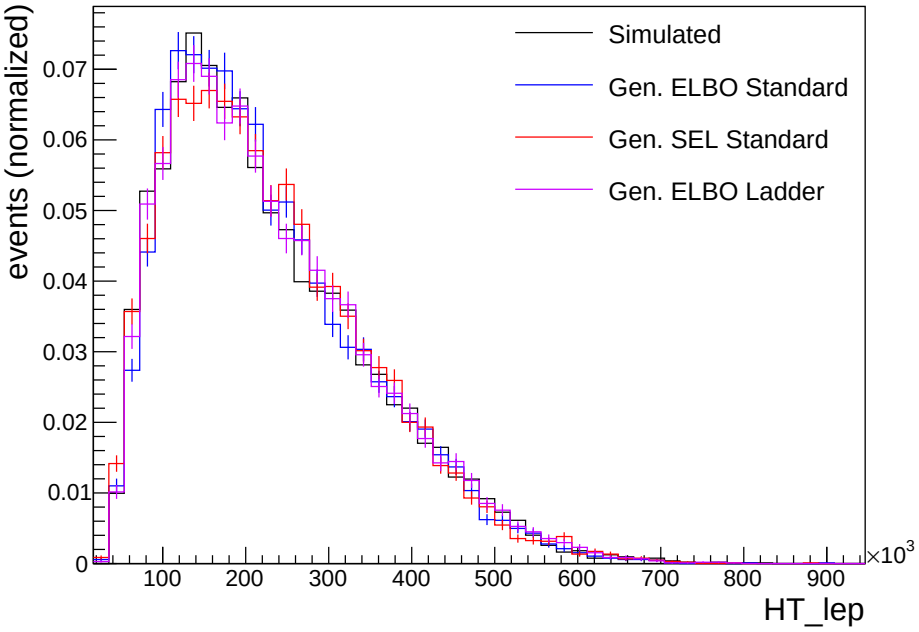



**Figure A.2:** Feature: HT_lep

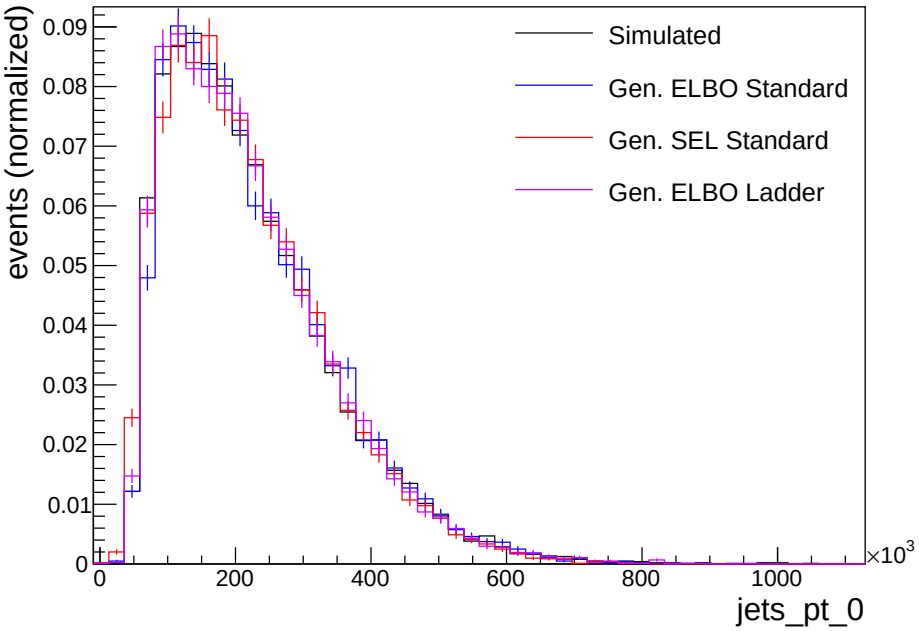



**Figure A.3:** Feature: jets_pt_0

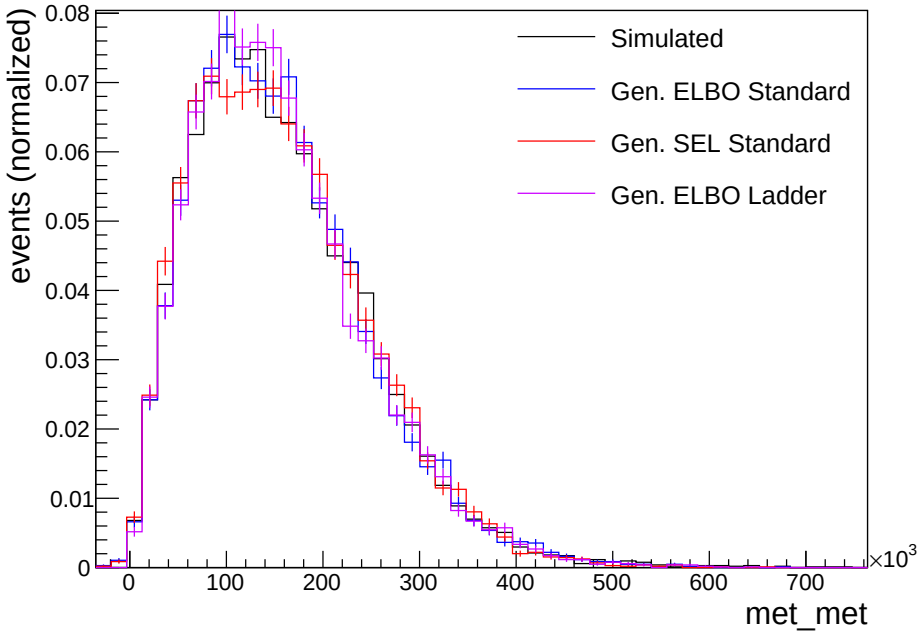



**Figure A.4:** Feature: met_met

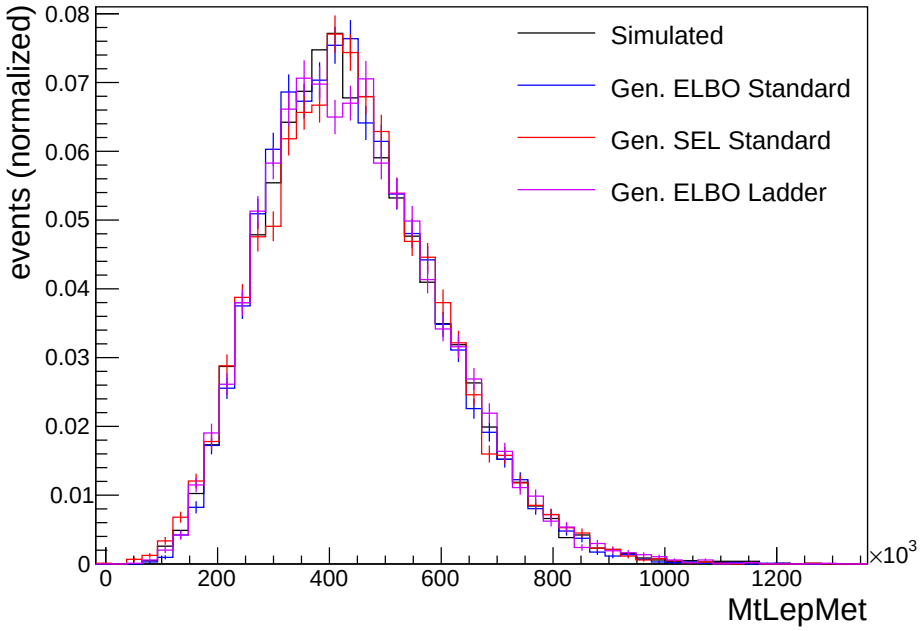



**Figure A.5:** Feature: MtLepMet

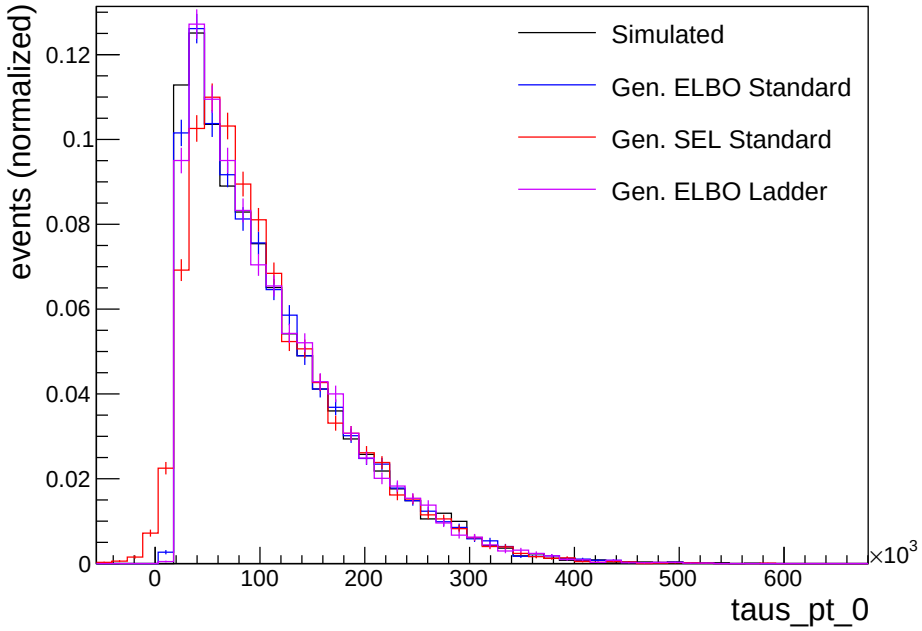



**Figure A.6:** Feature: taus_pt_0

## A.2 Background histograms

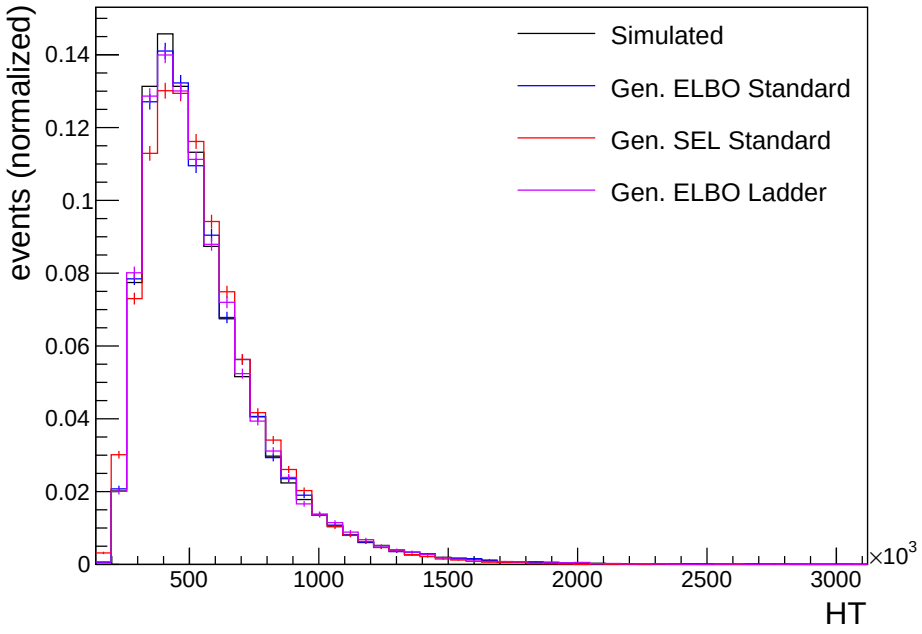



**Figure A.7:** Feature: HT

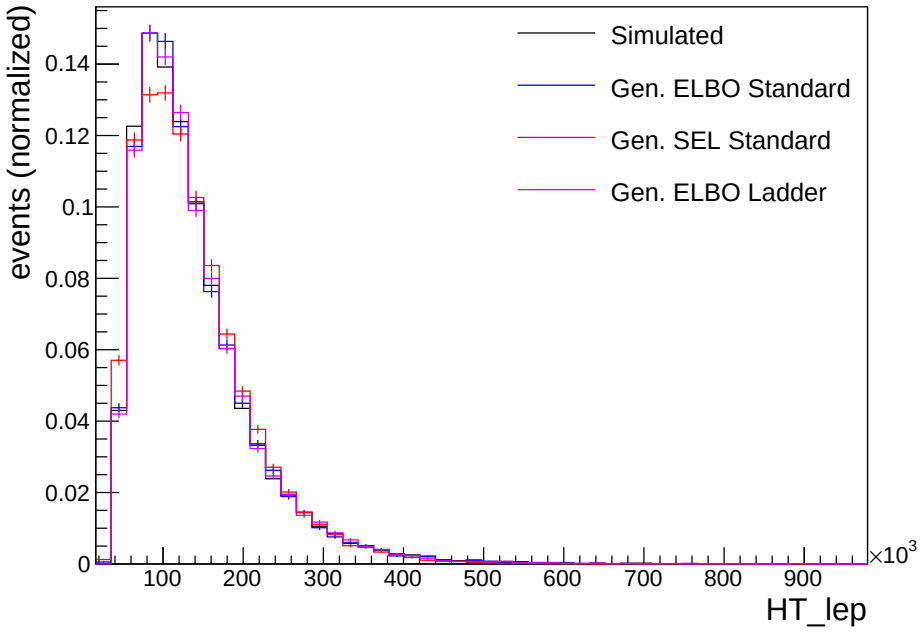



**Figure A.8:** Feature: HT_lep

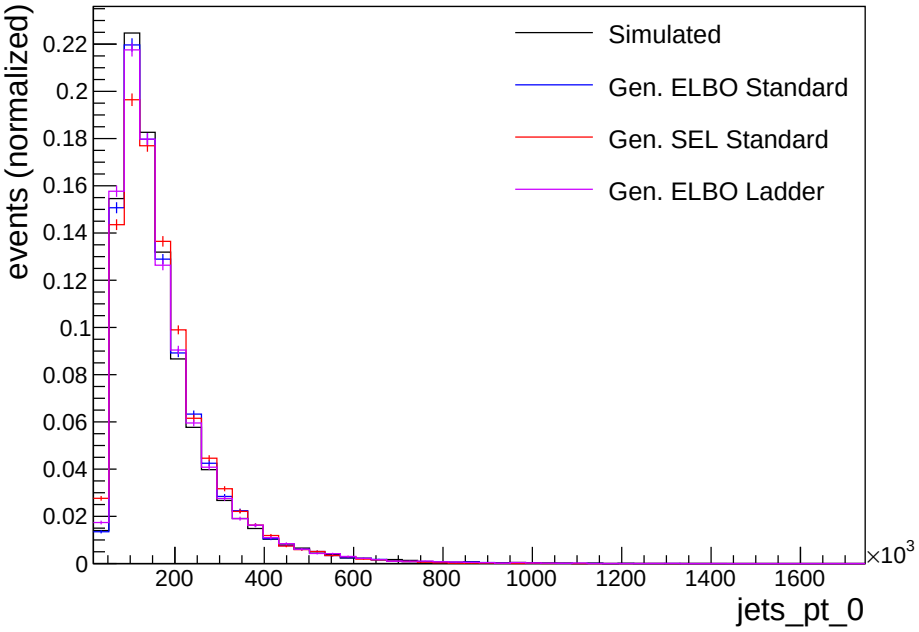



**Figure A.9:** Feature: jets_pt_0

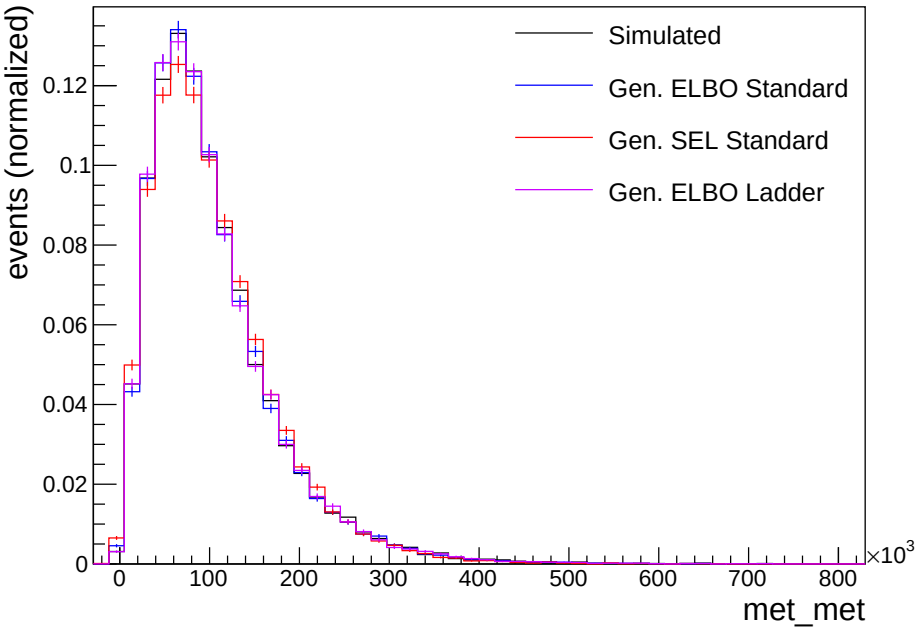



**Figure A.10:** Feature: met_met

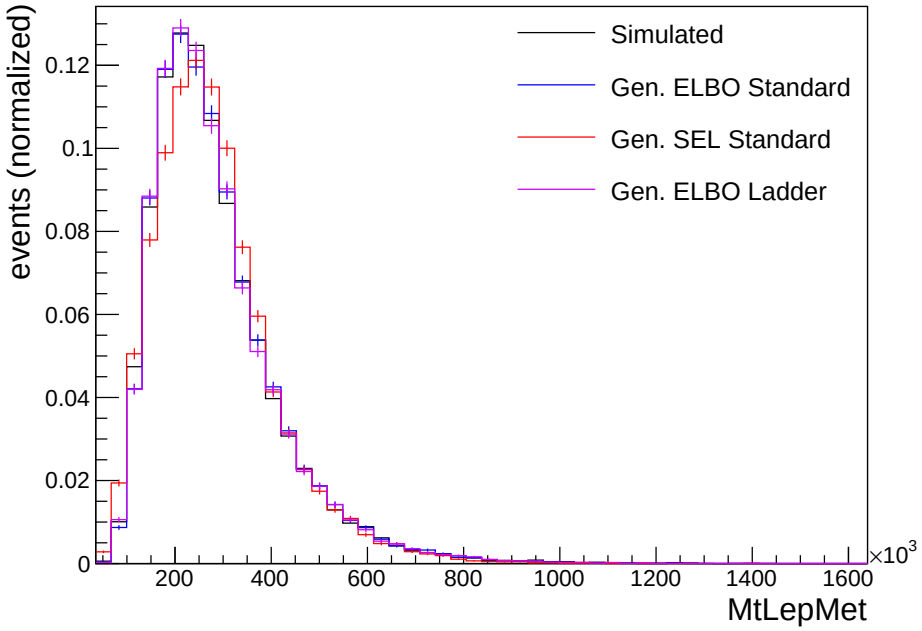



**Figure A.11:** Feature: MtLepMet

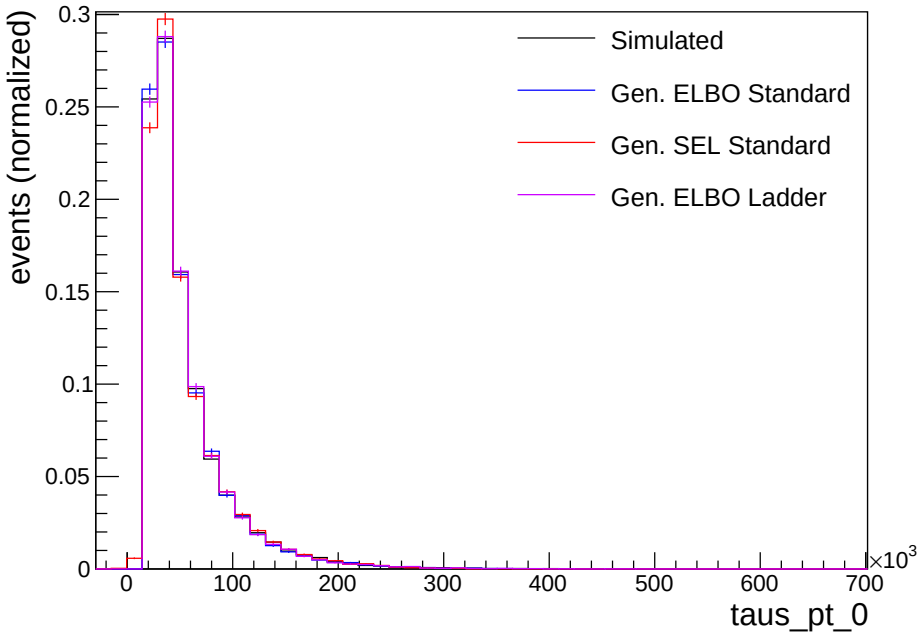



**Figure A.12:** Feature: taus_pt_0

## A.3 Corelation matrices

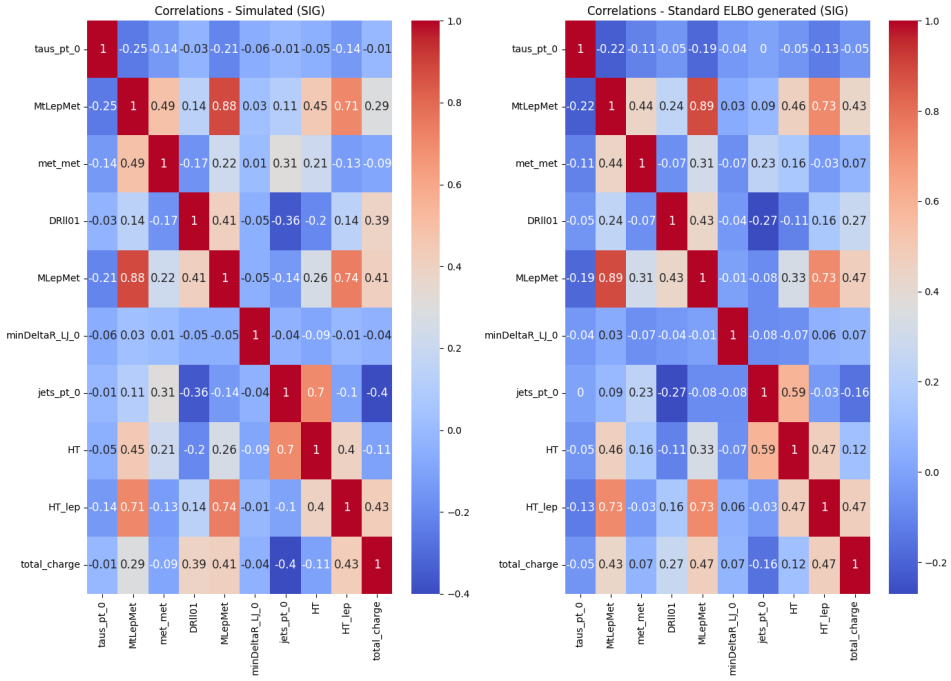



**Figure A.13:** Standard Symmetric Equilibrium Learning - Signal

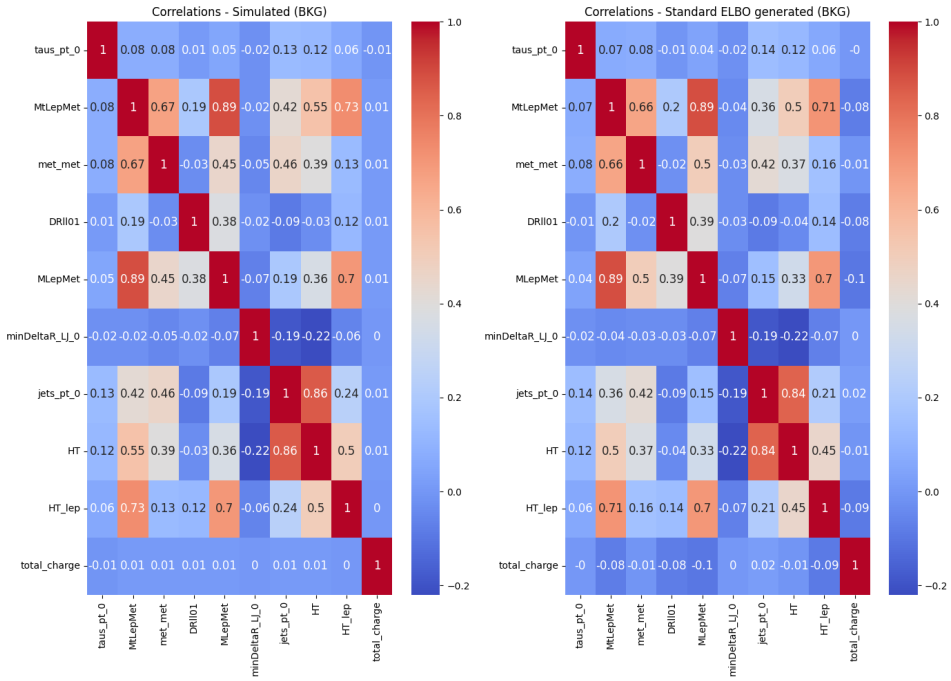



**Figure A.14:** Standard Symmetric Equilibrium Learning - Background

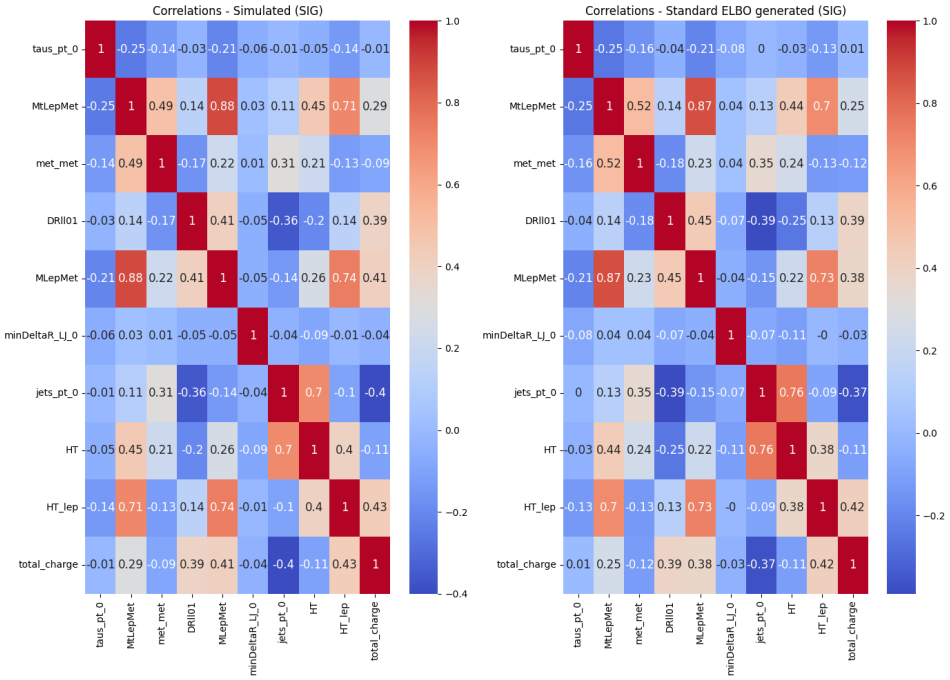



**Figure A.15:** Ladder Evidence lower bound learning - Signal

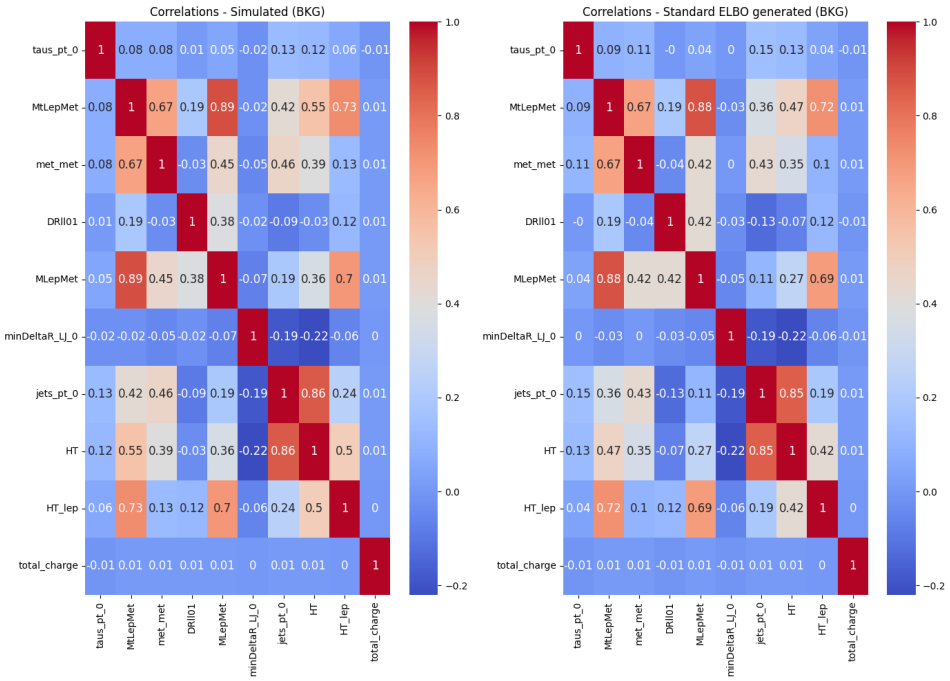



**Figure A.16:** Ladder Evidence lower bound learning - Background

## A.4 Training set results XGB

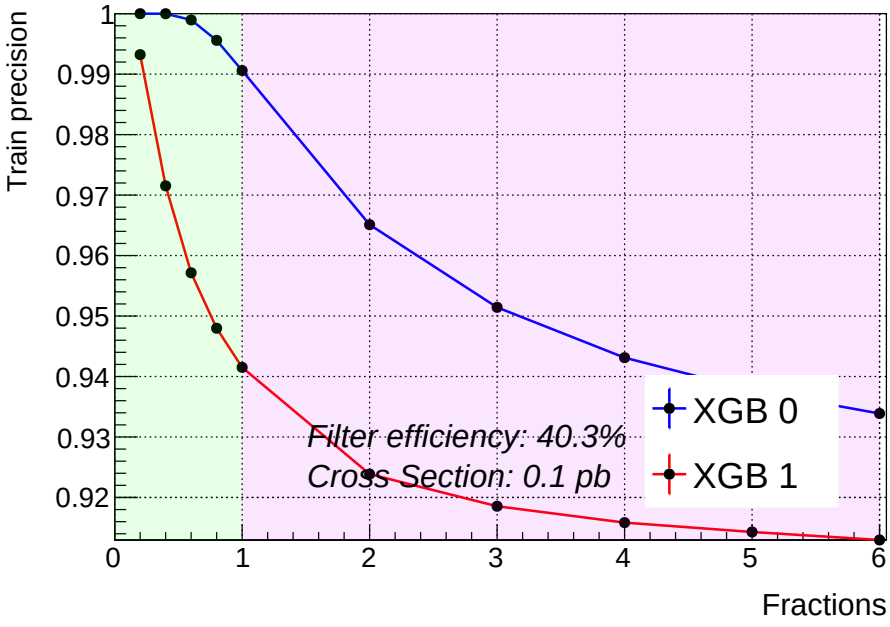



**Figure A.17:** Precision measured on the training dataset for XGB.

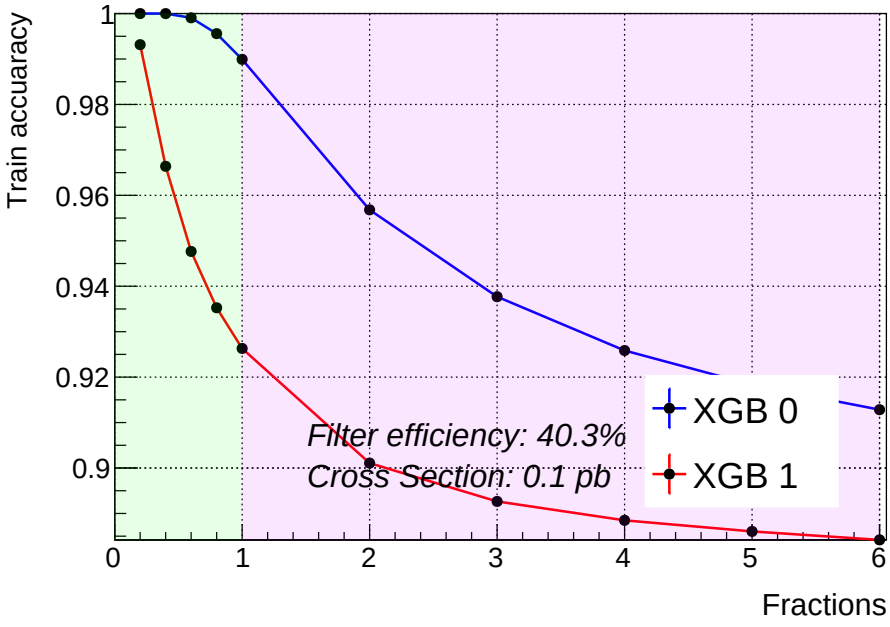



**Figure A.18:** Accuracy measured on the training dataset for XGB.

## A.5 Training set results MLP

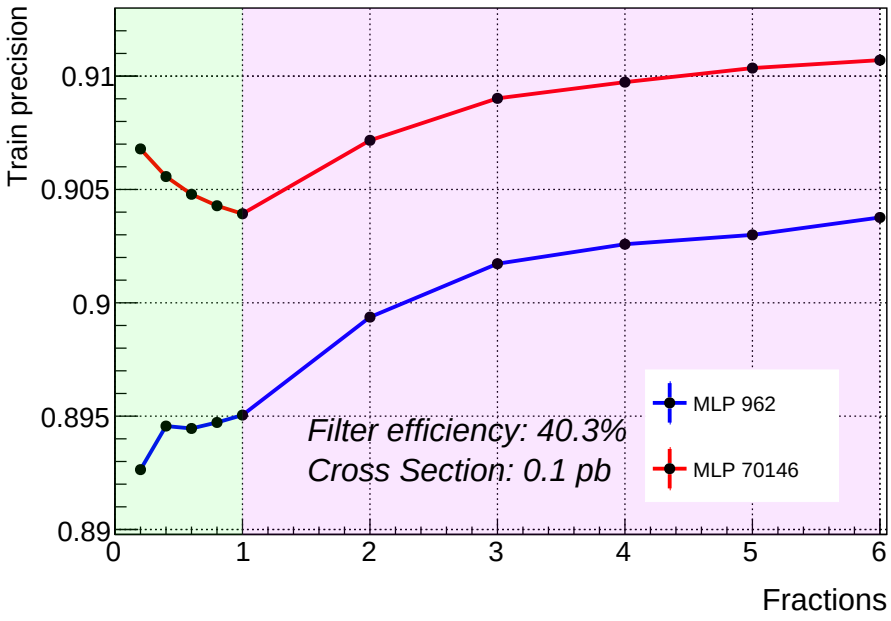



**Figure A.19:** Precision measured on the training dataset for MLP.

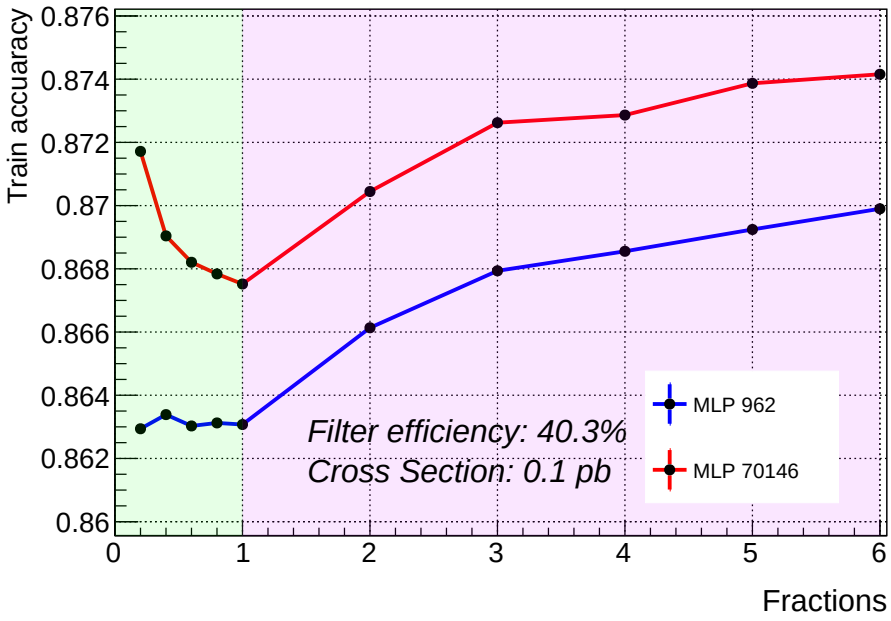



**Figure A.20:** Accuracy measured on the training dataset for MLP.

# Appendix B

## Pre-selection formula

```
(((abs(df.lep_ID_0)==13) & (df.lep_isMedium_0>0) & (df.lep_isolationFCLoose_0>0)&
(df.passPLIVVeryTight_0>0)) | ((abs(df.lep_ID_0)==11) & (df.lep_isolationFCLoose_0>0)&
(df.lep_isTightLH_0>0) & (df.lep_chargeIDBDTResult_recalc_rel207_tight_0>0.7) &
(df.passPLIVVeryTight_0>0)))&(((abs(df.lep_ID_1)==13) & (df.lep_isMedium_1>0) &
(df.lep_isolationFCLoose_1>0) & (df.passPLIVVeryTight_1>0)) | ((abs(df.lep_ID_1)==11) &
(df.lep_isolationFCLoose_1>0) & (df.lep_isTightLH_1>0) &
((df.lep_chargeIDBDTResult_recalc_rel207_tight_1>0.7)) &
(df.passPLIVVeryTight_1>0)))&((((abs(df.lep_ID_0) == 13)) | ( (abs( df.lep_ID_0 ) == 11)&
(df.lep_ambiguityType_0 == 0) & (~(((df.lep_Mtrktrk_atPV_CO_0<0.1) &
(df.lep_Mtrktrk_atPV_CO_0>0))& ~((df.lep_RadiusCO_0>20)  &((df.lep_Mtrktrk_atConvV_CO_0<0.1)&
(df.lep_Mtrktrk_atConvV_CO_0>0))))&~((df.lep_RadiusCO_0>20)&((df.lep_Mtrktrk_atConvV_CO_0<0.1)&
(df.lep_Mtrktrk_atConvV_CO_0>0)))))) & (((abs( df.lep_ID_1 ) == 11) & (df.lep_ambiguityType_1 == 0)&~
(((df.lep_Mtrktrk_atPV_CO_1<0.1)&(df.lep_Mtrktrk_atPV_CO_1>0)) & ~((df.lep_RadiusCO_1>20)&
((df.lep_Mtrktrk_atConvV_CO_1<0.1)&(df.lep_Mtrktrk_atConvV_CO_1>0))))&~((df.lep_RadiusCO_1>20)&
((df.lep_Mtrktrk_atConvV_CO_1<0.1)&(df.lep_Mtrktrk_atConvV_CO_1>0))))|((abs(df.lep_ID_1) == 13))))&\
(df.nTaus_OR==1)&((df.nJets_OR_TauOR>2) &
(df.nJets_OR_DL1r_70>0))&((df.dilep_type>0) & ((df.lep_ID_0*df.lep_ID_1)>0))
```

**Figure B.1:** The full version of the pre-selection $2lSS + 1\tau$ in Python code.