1 2 90

UNIVERSIDADE D
COIMBRA

Patrícia Gaspar Ferreira

MACHINE LEARNING FOR REAL TIME DETECTION OF ANOMALOUS EVENTS AT THE ATLAS EXPERIMENT AT THE LHC

Dissertação no âmbito do Mestrado em Engenharia Física orientada pelo Professor Doutor José Ricardo Morais Silva Gonçalo e pelo Doutor Miguel Correia dos Santos Crispim Romão e apresentada ao Departamento de Física da Faculdade de Ciências e Tecnologia.

Fevereiro de 2023

1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE Đ COIMBRA

Patrícia Gaspar Ferreira

Machine Learning for real time detection of anomalous events at the ATLAS experiment at the LHC

Thesis submitted to the
University of Coimbra for the degree of
Master in Engineering Physics

Supervisors:
Prof. Dr. José Ricardo Morais Silva Gonçalo
Dr. Miguel Correia dos Santos Crispim Romão

Coimbra, 2023

Abstract

It is not possible to save in real time the huge quantity of data generated at the ATLAS experiment, which is why the trigger exists. The trigger is responsible for selecting the events to be saved for later analysis. However, there is the concern that not all interesting events are being saved. Anomaly detection methods could be a way to improve the trigger to make sure these events are selected for storage.

In this work, autoencoders were used for semi-supervised anomaly detection and their ability to discriminate between background and signal events was studied. Multiple autoencoders with different latent space dimensions were trained just with background events. The performance of the best autoencoder was compared with a supervised neural network trained for each signal and the use of just one feature for discriminating background and signal events. Simulated Trigger-Level Analysis events were used as background events and simulated $HH \to b\bar{b}b\bar{b}$, $ZH \to b\bar{b}\nu\bar{\nu}$, $B^+ \to J/\Psi$, $B^- \to J/\Psi$ events were used as signal events.

It was observed that the autoencoders that performed better were the ones with lower and higher dimensions of the latent space. The best autoencoder performed worse than the supervised neural network, for all the signals. However, for the $HH \to b\bar{b}b\bar{b}$ signal, the best autoencoder had a better performance than the use of just one feature to discriminate between background and signal events. It was also found a negative correlation between the quality of the background reconstruction by the autoencoder and the performance of the autoencoder for two of the considered signals. When put together, these facts provide an indication that anomaly detection techniques may prove a useful technique to improve the robustness of the trigger event selection.

Keywords: machine learning, anomaly detection, autoencoder, trigger, ATLAS

Resumo

Não é possível guardar em tempo real a enorme quantidade de dados gerada na experiência ATLAS, motivo pelo qual existe o trigger. O trigger é responsável por selecionar os eventos que são guardados para análise. No entanto, existe a preocupação de não estarem a ser guardados todos os eventos interessantes. A utilização de métodos de deteção de anomalias pode ser uma forma de melhorar o trigger, de forma a assegurar que estes eventos são guardados.

Neste trabalho, foram usados autoencoders para deteção de anomalias semi-supervisionada e o seu desempenho a separar eventos de fundo e de sinal foi estudado. Vários autoencoders com diferentes dimensões do espaço latente foram treinados apenas com eventos de fundo. O desempenho do melhor autoencoder para cada sinal foi comparado com uma rede neuronal supervisionada treinada para cada sinal e a utilização de apenas uma variável para distinguir entre eventos de sinal e fundo. Eventos simulados de Trigger-Level Analysis foram usados como fundo e eventos simulados de $HH \rightarrow b\bar{b}b\bar{b}$, $ZH \rightarrow b\bar{b}\nu\overline{\nu}$, $B^+ \rightarrow J/\Psi$, $B^- \rightarrow J/\Psi$ foram usados como sinais.

Observou-se que os autoencoders com dimensões do espaço latente mais baixas e mais altas tiveram melhor desempenho. O melhor autoencoder teve pior desempenho que a rede neuronal supervisionada, para todos os sinais. No entanto, para o sinal $HH \rightarrow b\bar{b}b\bar{b}$, o melhor autoencoder foi melhor do que a utilização de apenas uma variável para distinguir entre eventos de fundo e sinal. Observou-se também a existência de uma correlação negativa entre a qualidade da reconstrução do fundo pelo autoencoder e o desempenho do autoencoder para dois dos sinais considerados. Juntas, estas observações dão uma indicação inicial de que as técnicas de deteção de anomalias têm o potencial para ser muito úteis, melhorando a robustez da seleção de eventos em tempo real feita pelo trigger.

Palavras-chave: aprendizagem automática, deteção de anomalias, autoencoder, trigger, ATLAS

Acknowledgments

I would like to thank my supervisors Ricardo Gonçalo and Miguel Romão for all their help, guidance and encouragement throughout this project.

I would also like to thank Filipe Veloso and Nuno Castro for their help and advice.

I am thankful to Ricardo Barrué for the generation of the samples used in this work. I am also thankful to my colleague Céu for her availability to help.

Finally, I am grateful to my family for all their support, specially to my father for believing in me.

Contents

List of Figures

List of Tables

List of Acronyms

AE	Autoencoder
ALICE	A Large Ion Collider Experiment
ATLAS	A Toroidal LHC Apparatus
AUC	Area Under the Curve
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
CP	Cluster Processor
CTP	Central Trigger Processor
FE	Front-End
FN	False Negative
FP	False Positive
FPR	False Positive Rate
HLT	High-Level Trigger
JEP	Jet/Energy-sum Processor
L1	Level-1
L1Calo	Level-1 Calorimeter
L1Muon	Level-1 Muon
L1Topo	Level-1 Topological
L2	Level-2
LAr	Liquid Argon

LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
MUCTPI	Muon Central Trigger Processor Interface
NN	Neural Network
ROC	Receiver Operator Characteristic
ReLU	Rectified Linear Unit
ROB	Readout Buffers
ROS	Readout Systems
RoI	Regions of Interest
TDAQ	Trigger and Data Acquisition
TLA	Trigger-Level Analysis
TN	True Negative
TP	True Positive
TPR	True Positive Rate

Chapter 1

Introduction

Particle physics is the area that studies the subatomic world in order to have a better understanting of the universe. To do that, colliders are built. The Large Hadron Collider (LHC) collides particles at high energies to expand our knowledge of subatomic particles.

The ATLAS experiment at the LHC generates a high amount of data as a result of the proton collisions. This data can not be all saved and the decision of which events are saved is made by the trigger.

It is important that the events saved are chosen well, since events not selected are lost forever. There might be interesting events occurring, but if the trigger does not select them to be saved, we will never know they are happening.

Anomaly detection methods trained only with known events can be used for the search of interesting events without depending on a theory. And using these methods at the trigger level is a way to maybe find interesting events directly at the the trigger level and assure they are being saved.

In this work, autoencoders, a method used for anomaly detection, trained only with background events, are used to separate background and signal events and their performance is studied.

This thesis is organized as follows. In Chapter 2 a brief description of the ATLAS detector and the trigger system is given. Chapter 3 presents a review of machine learning concepts needed for the understanding of this work and chapter 4 presents the data used. Chapter 5 contains the benchmarks used to compare with the anomaly detection method presented in chapter 6 and chapter 7 presents the conclusions of this work.

Chapter 2

ATLAS

The Large Hadron Collider (LHC) of the European Organization for Nuclear Research (CERN) is the most powerful particle accelerator in the world. It collides bunches of protons at a rate of 40 million per second with a center of mass energy of 13 TeV.

Along the LHC, there are four larger detectors designed to detect new physics events. A Toroidal LHC Apparatus (ATLAS) and Compact Muon Solenoid (CMS) are two general purpose detectors. The other two detectors, A Large Ion Collider Experiment (ALICE) and Large Hadron Collider beauty (LHCb), are specialized detectors [7].

2.1 Detector

The ATLAS detector, represented in figure 2.1, has a height of 25 m and a length of 44 m, and weights approximately 7000 tonnes. It is constituted by the magnet system, the inner detector, the calorimeter and the muon spectrometer.

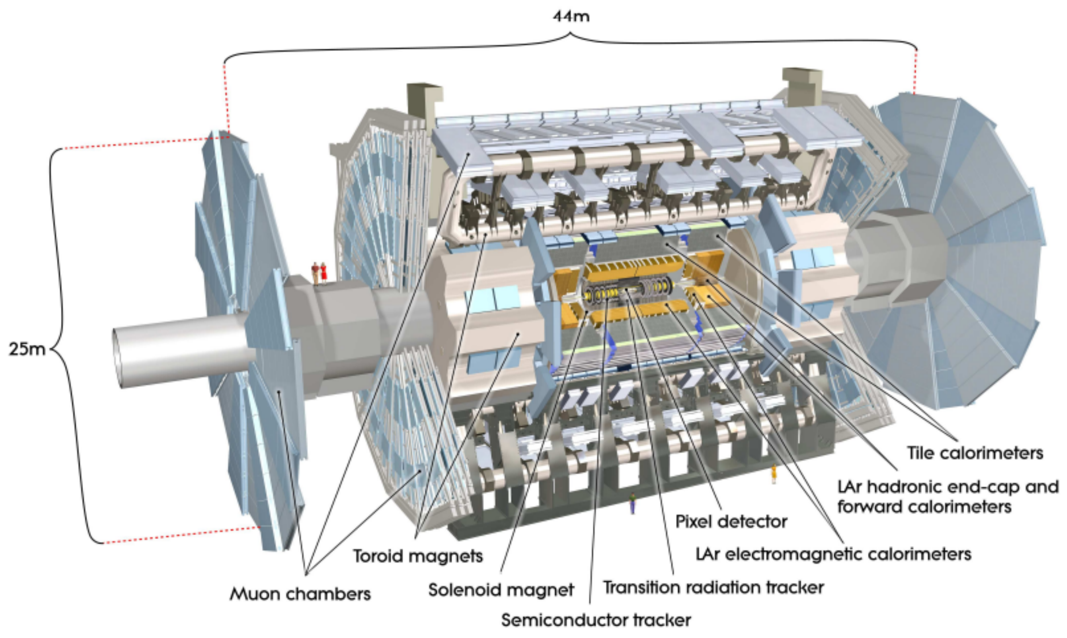
44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
Toroid magnets
LAr electromagnetic calorimeters
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

Figure 2.1: ATLAS detector [1].

The ATLAS coordinate system is a right-handed coordinate system, where the x-axis points from the collision point to the center of the LHC, the y-axis points upwards and the z-axis points in the direction of the beam. The angle θ is measured from the z-axis and the angle ϕ is measured around the beam axis. The pseudorapidity η is defined as

$$\eta = -ln\, tan\left(\frac{\theta}{2}\right) \tag{2.1}$$

Figure 2.2 shows the correspondence between a few values of the pseudorapidity and the polar angle.

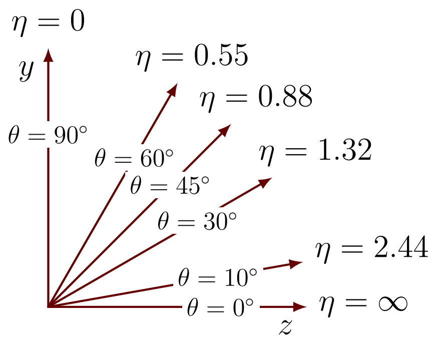

Figure 2.2: Correspondence between pseudorapidity (η) and the polar angle (θ) [2].

2.1.1 Inner Detector

The inner detector includes, in order of increasing radius, the pixel detector, the semiconductor tracker and the transition radiation tracker.

The pixel detector and the semiconductor tracker cover the region $|\eta| < 2.5$. The pixel detector consists of three concentric cylinders in the barrel and four disks in each end-cap. The semiconductor tracker consists of four concentric cylinders of silicon microstrip detectors in the barrel and nine disks in the end-caps.

The transition radiation tracker contains straw tubes with a 4 mm diameter. In the barrel, the straw tubes have a length of 144 cm and are positioned parallel to the beam. In the end-caps, the straw tubes have a length of 37 cm and are arranged in wheels. The transition radiation tracker covers the region $|\eta| < 2$ [1].

The inner detector is surrounded by a thin solenoid magnet (0.66 radiation lengths), which provides a 2 T magnetic field. This magnetic field changes the trajectory of the charged particles, allowing to measure their momentum.

2.1.2 Calorimeter

Calorimeters stop particles and measure their energy. They are constituted by an absorbing high-density material and an active material.

The ATLAS calorimeter includes the electromagnetic calorimeter and the hadronic calorimeter. The electromagnetic calorimeter measures the energy of electrons, positrons and photons. It uses lead as the absorber material and Liquid Argon (LAr) as the active material, where the ionisation charge left by charged particle cascades is measured. It includes a barrel ($|\eta| < 1.475$) and two end-cap ($1.375 < |\eta| < 3.2$) components, with each end-cap component consisting of two wheels with a common axis.

The hadronic calorimeter is responsible for the measurement of the energy of hadrons. It includes the tile calorimeter, the LAr hadronic end-cap calorimeter and the LAr forward calorimeter. The tile calorimeter is placed right after the electromagnetic calorimeter and is constituted by one barrel ($|\eta| < 1.0$) and two extended barrels ($0.8 < |\eta| < 1.7$). It uses steel as the absorber and scintillating tiles as the active material. The LAr hadronic end-cap calorimeter consists of two wheels in each end-cap. Each wheel is formed by 32 modules. Its η range overlaps with the LAr forward calorimeter and the tile calorimeter. The LAr forward calorimeter has three modules in each end-cap, covering the range $3.1 < |\eta| < 4.9$. One of the modules is made of copper, for the measurement of charged particles and the other two modules are made of tungsten, for the measurement of the energy of hadronic interactions. The three modules have liquid argon has the sensitive medium [1].

2.1.3 Muon Spectrometer

The muon spectrometer surrounds the calorimeter. It is designed to detect muons and measure their momentum, since they are not stopped in the calorimeter. The muon spectrometer includes precision-measurement tracking chambers (monitored drift tube chambers and cathode-strip chambers) and trigger chambers (resistive plate chambers and thin gap chambers). These chambers are arranged in three layers parallel to the beam in the barrel region and perpendicular to the beam in the end-caps.

The monitored drift tube chambers cover the region $|\eta| < 2.7$, except in the innermost layer, in which they cover $|\eta| < 2$. The region $2 < |\eta| < 2.7$ is covered by the cathode strip chambers.

The trigger chambers cover the region $|\eta| < 2.4$, with resistive plate chambers in the barrel and thin gap chambers in the end-caps. They provide fast signals that are used in the ATLAS trigger system to select events containing high-momentum muons in real time.

The muon tracks are bent by one barrel toroid and two end-cap toroids. In the region $|\eta| < 1.4$, the magnetic field is provided by the barrel toroid, in $1.4 < |\eta| < 1.6$ (transition region) by the combination of the barrel and end-cap toroids and in $1.6 < |\eta| < 2.7$ by the end-cap toroids [1].

2.2 Trigger

The trigger selects the events to be saved for offline analysis. It has three levels, the Level-1 (L1) trigger, Level-2 (L2) trigger and the event filter. The L2 trigger and the event filter form the High-Level Trigger (HLT). The L1 trigger is implemented in hardware and the HLT in software running on the trigger/DAQ CPU farm located in service cavern USA15 near the detector.

The trigger is configured through a trigger menu, in which trigger chains are defined. A chain consists of a L1 trigger item and HLT reconstruction and kinematic selection algorithms [3, 8]. Each chain is built to select a specific signature, like the existence of leptons, photons, jets, missing transverse momentum, total energy and

B-meson candidates [3].

Figure 2.3 represents the schematic of the trigger and data acquisition system.

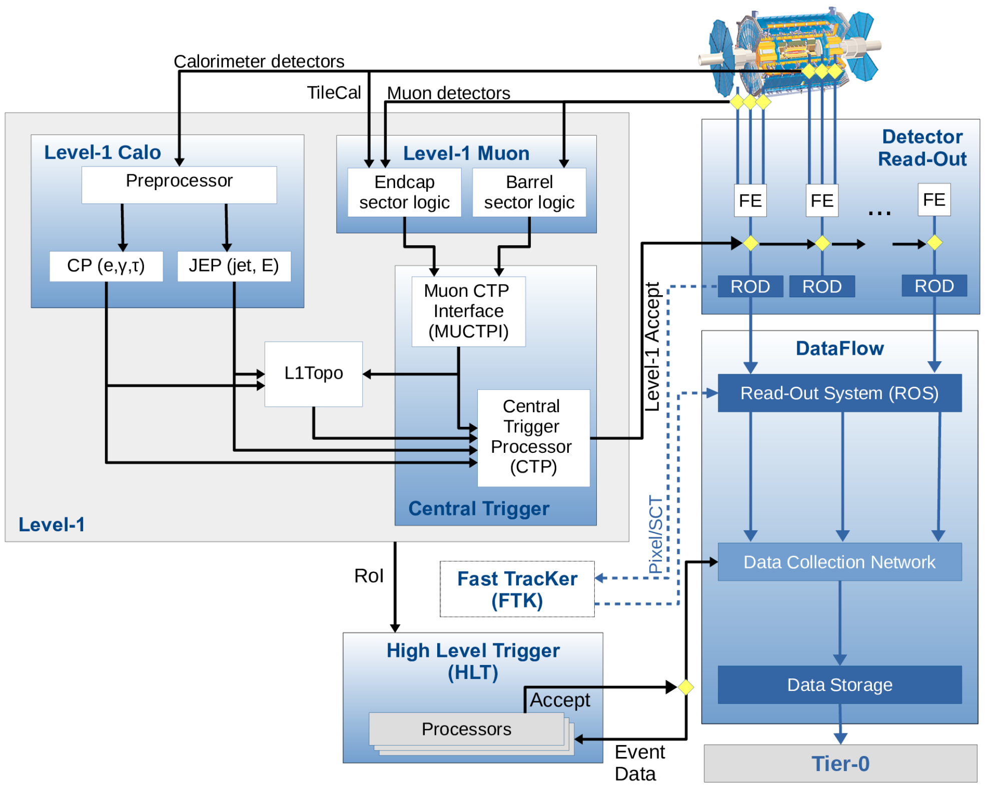

Figure 2.3: Trigger and Data Acquisition (TDAQ) system [3].

The L1 trigger reduces the event rate to a maximum of 100 kHz in 25 μs [3]. It uses information from the trigger chambers to search for muons with high transverse momentum, and information from the calorimeter to search for jets, τ-leptons and events with high missing transverse energy and total transverse energy [1]. Hadronic τ-lepton decays are identified by tracks in the inner detector matching a cluster in the electromagnetic and hadronic calorimeter; electrons by a deposit in the electromagnetic calorimeter combined with a track in the inner detector.

The L1 trigger is composed by the Level-1 Calorimeter trigger (L1Calo), the Level-1 Muon (L1Muon) trigger, the Level-1 Topological (L1Topo) trigger and the Central Trigger Processor (CTP). The L1Calo receives input from the calorimeter. The input signals are digitised and calibrated by the Preprocessor, which then sends them to the Cluster Processor (CP) and Jet/Energy-sum Processor (JEP). The CP identifies possible electrons, photons and τ-leptons that pass a defined transverse momentum threshold and the JEP identifies possible jets and determines sums of total and missing transverse energy [3].

The L1Topo receives objects from the L1Muon or L1Calo and makes selections using combined geometric or kinematic information from the objects received, such as angular distances [9, 3].

The CTP combines information from the L1Calo, L1Muon (passed through the Muon Central Trigger Processor Interface (MUCTPI)) and the L1Topo. The CTP uses this information to make the decision to accept or reject each event [9]. The CTP is also responsible for limiting the minimum time between two events being accepted and number of events accepted in a certain period of time (to stay within the available data transfer rate capability) [3].

If an event is accepted, the L1 trigger selects Regions of Interest (RoI), which are defined by coordinates in ϕ and η. These are areas where interesting features were found. These RoI are then used by the high-level trigger to refine the event selection.

After being accepted by the L1, the Front-End (FE) detector electronics read out the event data for all detectors and the event data is transferred to the detector-specific Readout Buffers (ROB), where the data is stored in fragments [3, 8]. The ROBs are grouped into Readout Systems (ROS) that are connected to the HLT [8].

The HLT, formed by the L2 trigger and the event filter, reduces the event rate to 1000 Hz [9].

The Fast TracKer (FTK) is a system based in hardware planned to reconstruct the inner detector tracks and make them available to the HLT at the same rate the events are accepted by the L1 trigger [3].

The L2 trigger uses part of the information in the RoI to accept or reject an event. If an event is accepted by the L2, the event builder assembles the data fragments from the ROBs, to be used by the event filter in a more precise event reconstruction and selection [8].

After being accepted by the HLT, the events are stored locally at the experimental site and then transferred to the tier-0 facility at CERN's computing centre for offline reconstruction [9].

In 2018, the average number of collisions per bunch crossing was 36, which gives around 1.4 billion events per second [10]. With an event size of around 1.6 MB, it would not be possible to save all the events, which is why the trigger exists, to select the 1000 events per second that can be saved.

Using anomaly detection methods at the trigger level might be a more efficient way of filtering events than the way the current trigger filters them. The purpose of this thesis is to investigate the use of machine learning to identify anomalous events at the trigger level with potential interest in the search of new physics.

2.2.1 Trigger-Level Analysis

Trigger-Level Analysis (TLA) is a method used to save events not selected by the trigger. Not all events can be saved, but Trigger-Level Analysis is a way of saving a higher number of events by saving only part of the information of each event. It saves HLT objects from events which are not fully reconstructed [11]. As the TLA has access to events before they are filtered by the HLT, it can also do a rudimentary (but fast) analysis of a larger ser of data than is available offline.

2.3 Data Analysis

The simulation of the data obtained by the ATLAS detector is performed and is compared with real data. First, the proton collisions are simulated and the events are generated using an event generator which uses the Monte Carlo method. Then, the resulting particles pass through a simulation of the detector, to simulate their interaction with the detector, using also Monte Carlo. After the detector simulation, a simulation of the trigger is done, to see which events are selected. Next, comes the full reconstruction of the physics objects, which the real events saved by the trigger also pass through. After the reconstruction, the obtained data are analised.

The Monte Carlo method is based on the generation of random numbers. Random numbers are generated following a uniform distribution, and are then transformed to follow a probability density function we are interested in. The resulting values can be treated as simulated measurements [12].

Chapter 3

Machine Learning

Machine Learning is an area of Artificial Intelligence in which the computer learns from "experience", which can be data. The goal is to find relations in the data and then use those to make predictions. Machine learning can be divided by the type of learning: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

Supervised learning uses labelled data. The dataset used is made up of feature vectors and their labels. Each instance of a feature vector contains the values of several features (variables) which characterize a given object or event. Labels may be discrete or continuous and classify events or objects into one of several classes, or quantify some property of these objects or events. The goal of supervised learning is to use the labelled examples to create a model that, given a new feature vector, can predict its label [13].

Supervised learning can be divided into regression and classification. In regression, the goal is to predict the labels which are continuous. One example of regression is the prediction of the price of a car [14]. In this case, the model would be trained with characteristics of multiple cars (feature vectors) and their prices (labels).

In classification, the labels are discrete. The goal is to determine to which class a new event or object belongs to. There is a defined number of classes. One example of classification is the spam filter, in which emails are classified as spam or not spam. By moving or not moving the emails to the spam folder, they are being classified as spam or not spam, allowing the filter to learn how to recognize whether new emails are spam or not [14].

Unsupervised learning uses unlabelled data. The goal is to find relations in the data without labels. For example, clustering, an unsupervised learning algorithm, can be used to find groups of similar people in the visitors of a blog [14].

Semi-supervised learning uses labelled and unlabelled examples. The goal is to build a model to predict the label of new examples. By using unlabelled examples, we are providing more data and it is expected that the resulting model will be better [13].

In reinforcement learning, the machine is in an environment and can perform actions in that environment. By performing actions, the machine gets rewards. The goal is to learn a policy, that indicates which action to perform in order to maximize the rewards [13, 14].

3.1 Neural Networks

A neural network is a machine learning method inspired by the human brain [14]. It is composed of a number of computational objects called artificial neurons, which are organized into an input layer, a set of hidden layers and an output layer, each containing one or several neurons, as represented in figure 3.1.

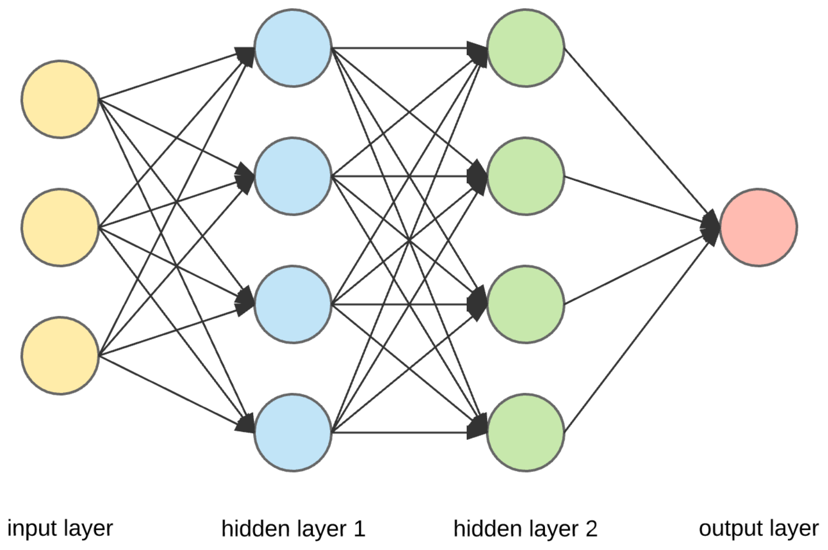

Figure 3.1: Neural network with input layer, two hidden layers and output layer [4].

The neurons of each layer are connected to neurons in the next layer. Each connection between neurons has a weight associated. Each neuron has also a bias term associated. The output of each neuron is the result of the combination of its inputs.

In fully connected layers or dense layers, all the neurons in a layer contribute to the input of each of the neurons in the next layer [15, 13].

The output of a neural network layer is given by

$$\boldsymbol{f_l} = \boldsymbol{g_l}\left(\boldsymbol{W}_l \boldsymbol{z} + \boldsymbol{b_l}\right) \tag{3.1}$$

where l is the number of the layer, $\boldsymbol{g_l}$ is the activation function, $\boldsymbol{W_l}$ is the weight matrix and $\boldsymbol{b_l}$ is the bias vector [13].

The activation function is a non-linear function, whose purpose is to allow the neural network to approximate non-linear functions [13].

The Rectified Linear Unit (ReLU) activation function, defined by

$$ReLU(x) = \begin{cases} 0, \text{ if } x < 0 \\ x, \text{ otherwise} \end{cases} \tag{3.2}$$

is commonly used in the hidden layers of a neural network [13].

The sigmoid function is used as the activation function for the output layer of the neural network in binary classification. It has an output between 0 and 1 and is defined as [13]

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

3.1.1 Training

Training is the process by which the weights of each connection between neurons are defined. Neural networks are trained by minimizing a loss function. The loss is a differentiable measure of how close the predicted labels are to the real labels. It is calculated using the predicted labels and the real labels.

The binary cross-entropy is used as the loss function for binary classification and is defined by

$$L = -\frac{1}{N} \sum_{i=1}^{N} y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i)) \tag{3.4}$$

where y_i is the label and $p(y_i)$ is the probability of the sample belonging to the class 1.

In order to find the minimum loss, stochastic gradient descent is used. In this method, a batch of the dataset is chosen randomly and the data used for training are passed through the network, in order to obtain the predictions. The loss is calculated using these predictions and then the gradient of the loss with respect to the training parameters (weights and bias) is calculated using backpropagation. Backpropagation starts from the loss and works from the last layer to the first to calculate the gradient using the chain rule [15]. The gradient indicates whether the loss is decreasing or increasing in that point. Since the goal is to find a minimum of the loss, we want to go in the opposite direction of the direction in which the loss is increasing. So, the weights are updated by subtracting a quantity proportional to the gradient of the loss [15, 13]:

$$\boldsymbol{W} \longleftarrow \boldsymbol{W} - \eta \nabla_{\boldsymbol{W}} L \tag{3.5}$$

where η is the learning rate which controls the size of the update of the weights.

One iteration of this process over the whole training set is called an epoch.

Gradient descent has the possibility of getting stuck on a local minimum and not finding the global minimum. To solve this, gradient descent with momentum can be used [15]. Momentum takes into account previous gradient values in the update of the weights, which prevents the algorithm from getting stuck in a local minimum.

The Adam optimizer [16] is an optimizer which implements stochastic gradient descent with momentum.

3.1.2 Data Preparation

Neural networks perform better if the features do not have very different ranges. Standardization can be used to rescale features so they have more similar ranges [14]. Standardization rescales features so they have a mean of 0 and a standard deviation of 1, which can be done by performing the following calculation

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}} \tag{3.6}$$

where $x^{(j)}$ is the vector with the values of the feature, $\mu^{(j)}$ is the mean and $\sigma^{(j)}$ the standard deviation of that feature [13].

Before starting training, the dataset used should be divided into three subsets: training, validation and test sets. The training set is used to train the model, the validation set is used to choose the best hyperparameters (defined in the next section), and monitor the performance during training, to check if the model is overfitting to the training data, and the test set is used to test the final model.

3.1.3 Hyperparameter optimization

Hyperparameters are properties of the algorithm that influence how it works, like the number of layers and neurons in a neural network [13]. Hyperparameters are not learned during training, they have to be defined before starting training by the user.

In order to find the best combination of hyperparameters, hyperparameter optimization is used. It consists of repeating the process of training multiple times with different combinations of hyperparameters. The performance of each model is evaluated on the validation set and the combination of hyperparameters with the best performance is chosen.

3.1.4 Regularization

One problem that can occur while training neural networks is overfitting. Overfitting occurs when the model predicts the labels of data it was trained on very well, but performs badly on data it has never seen. Regularization techniques can be used to prevent overfitting.

Early stopping is used to stop training once the loss on the validation set has stopped decreasing for a certain number of epochs (patience). The model is saved after each epoch and the best model is restored when training is stopped. Alternatively, the model can be trained for a determined number of epochs and in the end restored the best model.

Dropout layers are also used as a regularization technique. A dropout layer sets a certain percentage of the neurons to zero. The percentage of neurons set to zero is the dropout rate. Using dropout stops neurons from coadapting with the neurons close to them and makes it so that each neuron does not depend too much on just a few input neurons. This makes the neurons less susceptible to small changes in the inputs, resulting in a network that generalizes better [14].

Batch normalization is the process of standardizing the output of each layer before it is used as input to the next layer, for each batch. Batch normalization makes training faster and has some regularization effect [13].

3.1.5 ROC

In order to evaluate the performance of a neural network, the ROC curve can be used. The Receiver Operator Characteristic (ROC) curve is a plot of the true positive rate as a function of the false positive rate, as represented in figure 3.2.

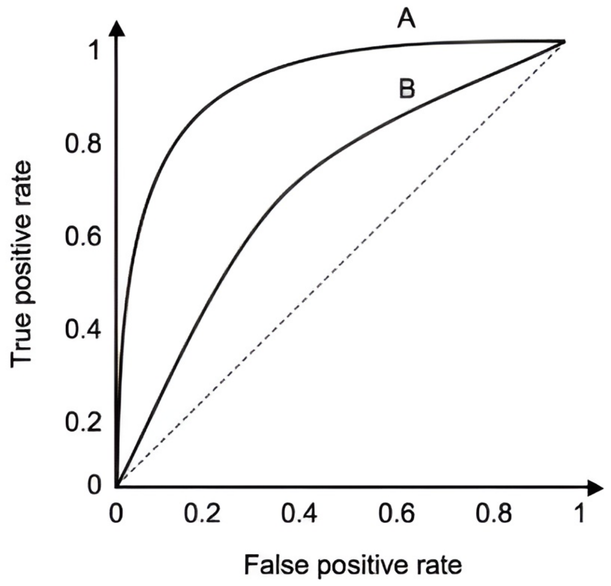

Figure 3.2: Example of two ROC curves. The dashed line corresponds to a random classifier [5].

The True Positive Rate (TPR) and the False Positive Rate (FPR) are defined as:

$$TPR = \frac{TP}{TP + FN} \tag{3.7}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.8}$$

In a binary classification problem, we can consider one of the classes as negative and the other one as positive. Then, True Negative (TN) is the number of points classified correctly as negative and False Negative (FN) is the number of points falsely classified as negative, that are in fact positive. True Positive (TP) is the number of points classified correctly as positive and False Positive (FP) is the number of points falsely classified as positive.

In order to obtain the values of the TPR and FPR necessary to plot the ROC curve, a number of thresholds are defined between the minimum and the maximum of the output values of the classifier. For each threshold, points over the threshold are taken to be positive and points under the threshold are taken to be negative. Comparing with the true labels, the TPR and FPR are computed.

The Area Under the Curve (AUC) of the ROC can be used as a performance metric of a model. The higher the AUC, the better the model is. In figure 3.2, curve A corresponds to a better classifier than curve B, since it has a higher AUC.

3.2 Anomaly Detection

Anomaly detection is an area of machine learning. Anomaly detection methods are used to find anomalies in a dataset constituted mostly by normal events.

Anomaly detection methods can be supervised, semi-supervised and unsupervised. In supervised anomaly detection, normal events and anomalous events are used to train the model. Unsupervised anomaly detection methods find anomalies in an unlabelled dataset [17]. In semi-supervised anomaly detection, methods can be trained with only normal events in order to identify different events not seen during training. Autoencoders are a method that can be used for semi-supervised anomaly detection.

3.2.1 Autoencoder

An autoencoder is a machine learning algorithm that learns to compress data into a space with a smaller dimensionality than the dataset and then decompress it to reconstruct the input data as accurately as possible. An autoencoder is made up of two neural networks, the encoder and the decoder, as represented in figure 3.3. The encoder compresses the input data into a latent space representation and the decoder decompresses it, trying to reconstruct the inputs.

The autoencoder is trained to minimize the loss, which is the difference between its outputs and inputs, called the reconstruction error. The mean squared error can be used as the loss in the training of autoencoders and is calculated by:

$$L = \frac{1}{N}\sum_{i}^{N}(\boldsymbol{y}_i - \boldsymbol{x}_i)^2 \tag{3.9}$$

where $\boldsymbol{x}_i$ is the input vector and $\boldsymbol{y}_i$ is the output of the autoencoder and N is the number of instances [13].

The reconstruction error can be used as an anomaly score. An anomalous event is expected to be more difficult to reconstruct and have a higher reconstruction error than the normal events used to train the autoencoder.

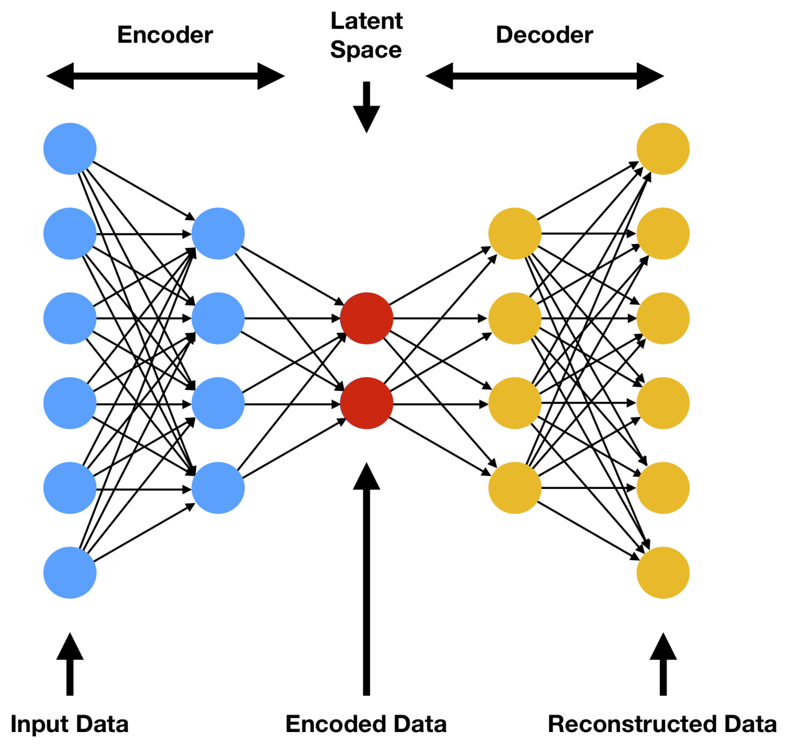

Figure 3.3: Autoencoder [6]

The R^2 can be used as a measure of how well the autoencoder reconstructs its inputs. It is defined as

$$R^2 = 1 - \frac{\sum_i^N (y_i - \hat{y}_i)^2}{\sum_i^N (y_i - \overline{y})^2} \tag{3.10}$$

where y_i is the real value, $\hat{y}_i$ is the predicted value and $\overline{y}$ is the mean of the true values [18]. The closer the R^2 is to 1, the better the reconstruction of the input data by the autoencoder.

3.3 Implementation and Software Tools

Several software tools were used during this project. The following gives a short description of their functions

Uproot

Uproot [19] is a Python library for reading and writing ROOT files. Uproot version 4.1.8 was used to read the data in ROOT files and pass it to files in HDF format.

NumPy and Pandas

Pandas [20, 21, 22] is a Python library for data manipulation and analysis. NumPy [23, 24] is Python library to work with arrays and matrices. NumPy version 1.22.0 and Pandas version 1.3.5 were used to manipulate data.

Scikit-learn

Scikit-learn [25, 26] is a Python library for machine learning. Scikit-learn version 1.0.2 was used for the standardization of the data, for the division of the dataset into training, validation and test sets and for the determination of the ROC AUC and the R^2.

Matplotlib,Seaborn

Matplotlib [27, 28] and Seaborn [29, 30] are Python libraries for data visualization. Matplotlib version 3.5.1 and Seaborn version 0.11.2 were used to create plots.

Keras

Keras [31] version 2.5.0 was used for building and training the neural networks and the autoencoders. Keras is a Python library for the implementation of neural networks. The Sequential API which builds models made of layers connected sequentially, was used to build the neural networks.

Optuna

Optuna [32, 33] version 2.10.0 was used for the hyperparameter optimization. Optuna is an hyperparameter optimization framework for machine learning. The Optuna sampler used for the sampling of parameters to test uses the Tree-structured Parzen Estimator [34]. The median pruner was used to stop unpromising trials. The median pruner stops a trial if the best intermediate result is worse than the median of intermediate results of the previous trials at the same step.

Chapter 4

Data

The dataset used is composed by simulated data by Monte Carlo.

The dataset is composed of 10,000 events from Trigger-Level Analysis, 50,000 events from the decay of 2 Higgs bosons into 2 bottom quarks and 2 antiquarks ($HH \longrightarrow b\bar{b}b\bar{b}$), 10,000 events from B^+ meson decaying to J/ψ ($B^+ \longrightarrow J/\psi$), 10,000 from a B^- meson to J/ψ ($B^- \longrightarrow J/\psi$) and 9,000 events from a Z boson and a Higgs boson decaying into one bottom quark and antiquark and one neutrino and antineutrino ($ZH \rightarrow b\bar{b}\nu\bar{\nu}$).

The TLA events were considered to be the background and the other were considered to be the signal.

4.1 Features

From the information of each event, the 4 jets, 2 muons and 2 electrons with higher transverse momentum were selected. The transverse momentum (p_T), pseudorapidity (η) and ϕ angle of each of the jets, muons and electrons, and the number of jets, muons and electrons were used as features. Only jets with $p_T > 20$ GeV and $|\eta| < 4$ were used. When there were not 4 jets which met these conditions, the values of the p_T, η and ϕ of those were set to 0.

The jets used were reconstructed with the anti-k_T algorithm, which is a sequential clustering algorithm. In this algorithm, the distance between two entities (for example reconstructed energy deposits in the calorimeter, but also charged particle tracks or true particle 4-momenta) is calculated by

$$d_{ij} = min\left(\frac{1}{k_{Ti}^2}, \frac{1}{k_{Tj}^2}\right)\frac{R_{ij}^2}{R^2} \tag{4.1}$$

where

$$R_{ij} = \sqrt{(y_i - y_j)^2 - (\phi_i - \phi_j)^2} \tag{4.2}$$

and k_{Ti} and k_{Tj} are the transverse momentum of the entities i and j and y_i and y_j are the rapidity (often replaced with pseudorapidity η in experimental data analysis).

The distance between an entity and the beam is calculated by

$$d_{iB} = \frac{1}{k_{Ti}^2} \tag{4.3}$$

All the distances d_{ij} and d_{iB} are calculated. If the smallest distance is the distance between two entities, the two entities are combined and the process repeats. If the smallest distance is the distance of the entity to the beam, the entity is called a jet [35].

Figures 4.1, 4.2, 4.3 and 4.4 represent the histograms of some of the features used. The histograms were normalized to have unit area and in the case of the pseudorapidity and azimuthal angle, the 0 values are not shown.

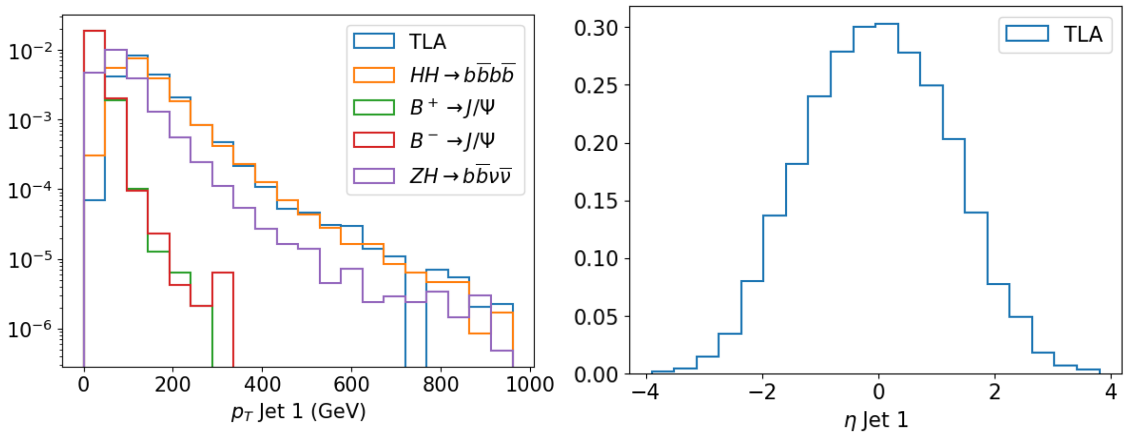

Figure 4.1: Histograms of the transverse momentum for the events of TLA, $HH \longrightarrow b\bar{b}b\bar{b}$, $B^+ \longrightarrow J/\psi$, $B^- \longrightarrow J/\psi$ and $ZH \rightarrow b\bar{b}\nu\bar{\nu}$ and pseudorapidity of the jet with the highest momentum of the TLA events.

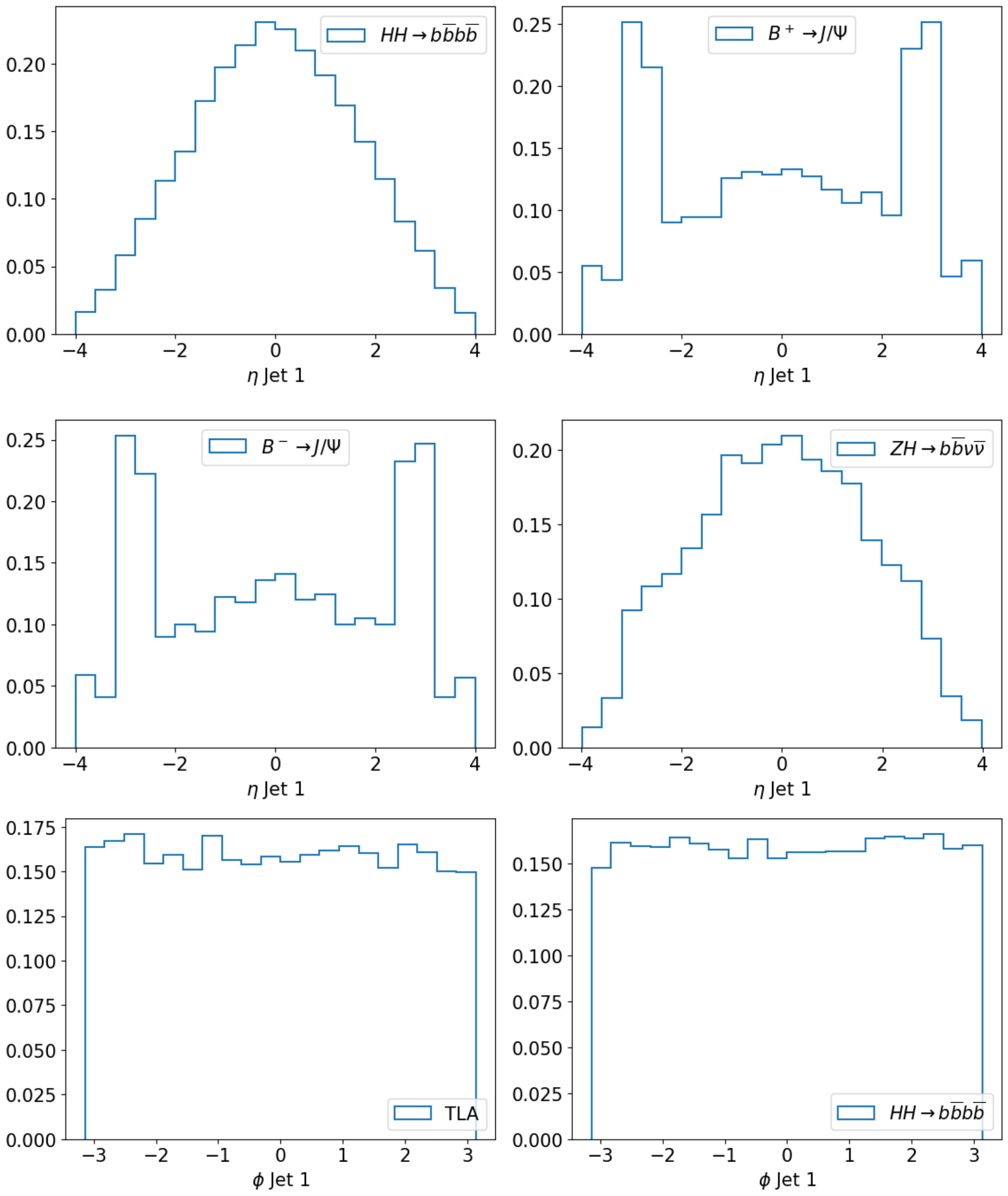

Figure 4.2: Histograms of the pseudorapidity of the jet with the highest transverse momentum of the events from $HH \longrightarrow b\bar{b}b\bar{b}$, $B^+ \longrightarrow J/\psi$, $B^- \longrightarrow J/\psi$ and $ZH \rightarrow b\bar{b}\nu\bar{\nu}$ and azimuthal angle of the same jet of events from TLA and $HH \longrightarrow b\bar{b}b\bar{b}$.

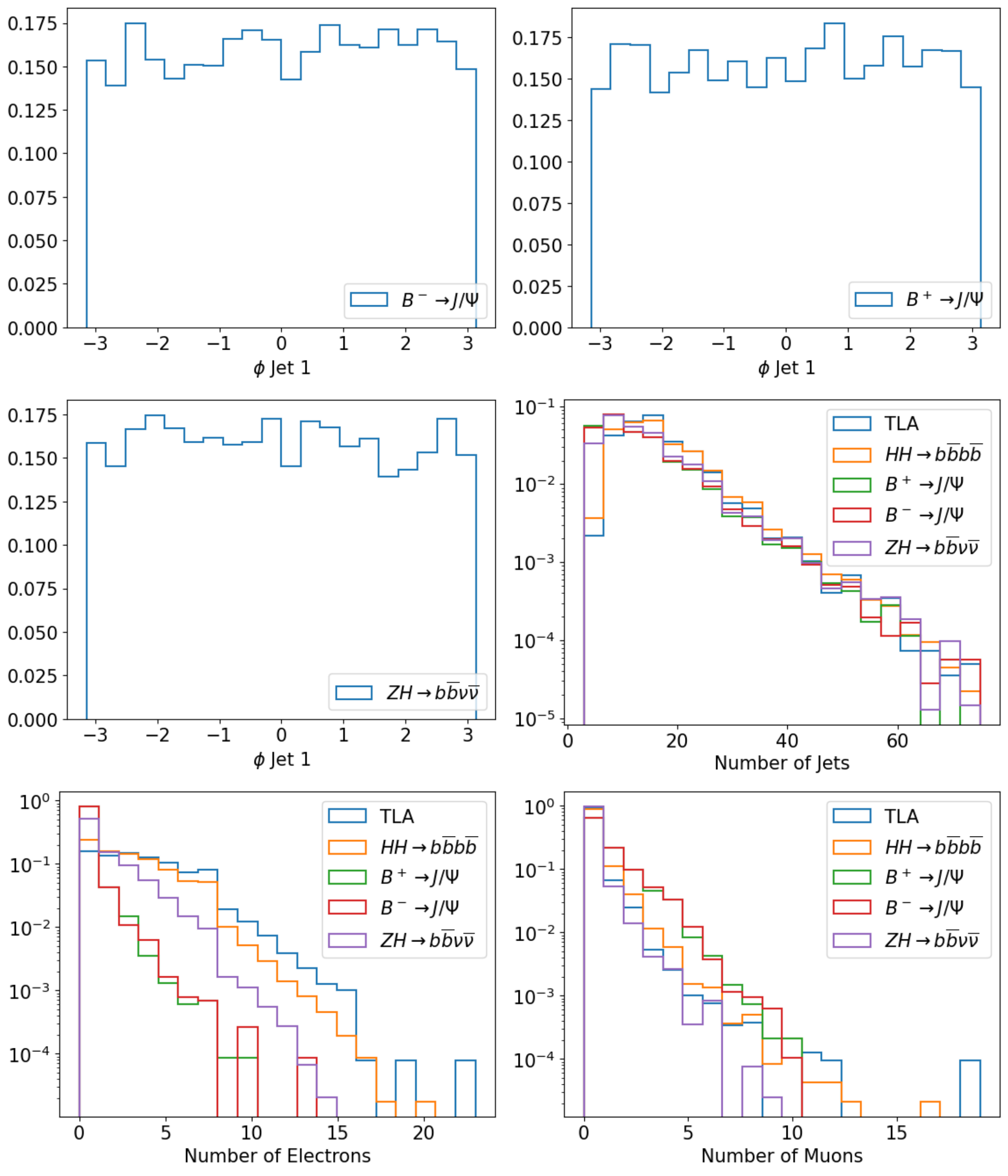

Figure 4.3: Histograms of the azimuthal angle of the jet with higher p_T of the events from $B^- \longrightarrow J/\psi$, $B^+ \longrightarrow J/\psi$, $ZH \to b\bar{b}\nu\bar{\nu}$ and number of jets, electrons and muons.

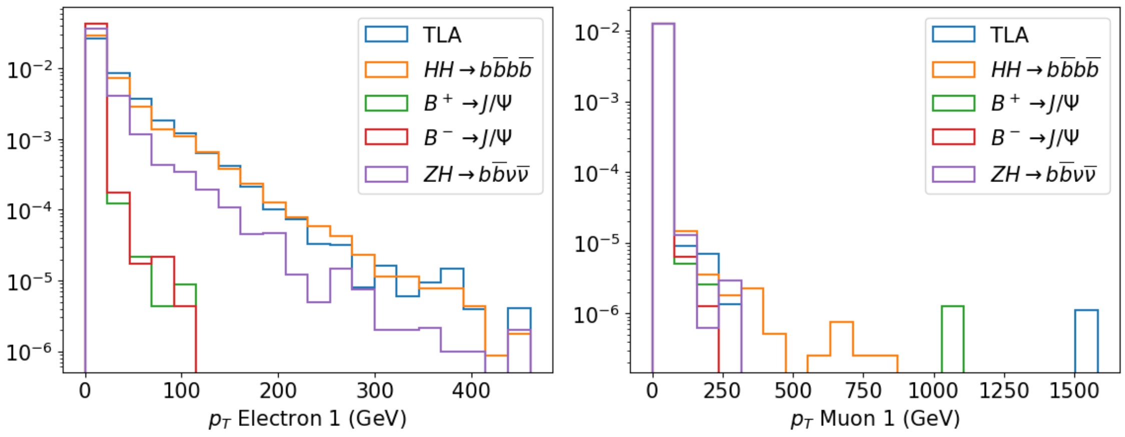

Figure 4.4: Histograms of the transverse momentum of the electron and muon with the highest transverse momentum for the events of TLA, $HH \longrightarrow b\bar{b}b\bar{b}$, $B^+ \longrightarrow J/\psi$, $B^- \longrightarrow J/\psi$ and $ZH \rightarrow b\bar{b}\nu\bar{\nu}$.

Looking at the distribution of the transverse momentum of the jet with the higher transverse momentum, the TLA and $HH \longrightarrow b\bar{b}b\bar{b}$ events follow a similar distribution. $B^+ \longrightarrow J/\psi$ and $B^- \longrightarrow J/\psi$ events also have a similar distribution. TLA and $HH \longrightarrow b\bar{b}b\bar{b}$ events have higher p_T, followed by the $ZH \rightarrow b\bar{b}\nu\bar{\nu}$ events, and then by $B^+ \longrightarrow J/\psi$ and $B^- \longrightarrow J/\psi$ events.

Regarding the pseudorapidity of the jet with higher p_T. TLA and $HH \longrightarrow b\bar{b}b\bar{b}$ events have a distribution of the pseudorapidity with the same shape, both with a maximum around $\eta = 0$. The pseudorapidity distribution for the $ZH \rightarrow b\bar{b}\nu\bar{\nu}$ seems to have a shape close to the TLA and $HH \longrightarrow b\bar{b}b\bar{b}$, but with more variation. The pseudorapidity distribution of the $B^+ \longrightarrow J/\psi$ events is similar to the distributions of the $B^- \longrightarrow J/\psi$ events, with both distributions having two big maxima around $\eta = -3$ and $\eta = 3$, and two smaller ones to the left and right of these, respectively.

The distribution of the ϕ angle of the jet with the higher p_T, looks similar for all samples, with some having more variation than others, but it is mostly evenly distributed for all values of ϕ.

Looking at the distribution of the number of electrons, TLA events have a higher number of electrons, followed by the $HH \longrightarrow b\bar{b}b\bar{b}$ events, then by the $ZH \rightarrow b\bar{b}\nu\bar{\nu}$ events and then by both $B^+ \longrightarrow J/\psi$ and $B^- \longrightarrow J/\psi$ events.

Relatively to the distribution of the number of muons, $B^+ \longrightarrow J/\psi$ and $B^- \longrightarrow J/\psi$ events have a similar distribution. The distribution of the $ZH \rightarrow b\bar{b}\nu\bar{\nu}$ events is close too the distribution of the TLA events.

Looking at the distribution of the p_T of the electron with higher p_T, we can see that $B^+ \longrightarrow J/\psi$, and $B^- \longrightarrow J/\psi$ events have lower values of p_T of electrons, with the TLA and $HH \longrightarrow b\bar{b}b\bar{b}$ events having higher values of p_T.

Chapter 5

Benchmarks for Anomaly Detection

In order to compare the performance of the autoencoder at discriminating between background and signal events, it was determined the AUC of the ROC of each feature and trained a supervised classifier for each signal.

The events with negative weights were removed and the dataset was randomly divided into training, validation and test sets, each with one third of the events.

5.1 AUC Features

For each feature, the area under the curve (AUC) of the ROC was determined on the test set. The ROC is obtained by applying thresholds on the values of each feature and determining the true positive rate and the false positive rate. The AUC of the ROC is a measure of how separable the distributions of signal and background are.

The determined values of the AUC of the ROC for each feature are presented in table 5.1.

Table 5.1: Values of the AUC of the ROC of the features for each of the signals. The highest values of the AUC for each signal are marked in bold.

Feature	$HH \to b\bar{b}b\bar{b}$	$B^+ \to J/\Psi$	$B^- \to J/\Psi$	$ZH \to b\bar{b}\nu\bar{\nu}$
p_T Jet 1	0.5485	0.9947	0.9944	0.8216
p_T Jet 2	0.5643	**0.9953**	**0.9951**	0.9133
p_T Jet 3	0.5684	0.9939	0.9937	0.9428
p_T Jet 4	0.5824	0.9844	0.9842	**0.9456**
ϕ Jet 1	0.5128	0.5184	0.5151	0.5127
ϕ Jet 2	0.5066	0.5079	0.5085	0.5058
ϕ Jet 3	0.5040	0.5066	0.5109	0.5089
ϕ Jet 4	0.5007	0.5063	0.5051	0.5079
η Jet 1	0.5015	0.5132	0.5048	0.5099
η Jet 2	0.5013	0.5032	0.5059	0.5108
η Jet 3	0.5013	0.5033	0.5035	0.5039
η Jet 4	0.5028	0.5048	0.5099	0.5018
Number of Jets	0.5200	0.6956	0.6886	0.6498
p_T Electron 1	0.5327	0.6430	0.6481	0.5156
p_T Electron 2	0.5097	0.5725	0.5780	0.5095
ϕ Electron 1	0.5012	0.5000	0.5031	0.5048
ϕ Electron 2	0.5007	0.5038	0.5050	0.5005
η Electron 1	0.5037	0.5083	0.5011	0.5028
η Electron 2	0.5002	0.5042	0.5033	0.5012
Number of Electrons	0.5314	0.6498	0.6556	0.5162
p_T Muon 1	0.5480	0.9176	0.9179	0.7297
p_T Muon 2	0.5587	0.8891	0.8887	0.7479
ϕ Muon 1	0.5322	0.6914	0.6921	0.6120
ϕ Muon 2	0.5268	0.5961	0.5954	0.5737
η Muon 1	0.5018	0.5297	0.5286	0.5254
η Muon 2	0.5019	0.5005	0.5004	0.5026
Number of Muons	**0.5922**	0.9299	0.9293	0.7938

For the $HH \to b\bar{b}b\bar{b}$, the feature with the higher AUC is the number of muons, while for the other signals it is the p_T of jets. For the $B^+ \to J/\Psi$ and $B^- \to J/\Psi$, it is the p_T of the second jet with the highest p_T and for the $ZH \to b\bar{b}\nu\bar{\nu}$, it is the fourth.

5.2 Neural Network

For each of the signals, a neural network was trained to classify events into background or signal.

The data was standardized and the event weights were normalized so that the sum of the weights of background events was equal to the sum of the weights of signal events and the sum of all the weights was equal to the number of events.

The models were built using the Sequential API from Keras. Each model had an input layer with 27 neurons and the hidden layers were dense layers with the activation function ReLU. After each hidden layer, a dropout layer was placed and for the output a dense layer with 1 neuron and the sigmoid activation function was used.

The model was compiled with the Adam optimizer and the binary cross-entropy loss and the number of epochs was defined as 400. Early stopping with 20 epochs of patience was used to stop training once the loss on the validation set started increasing and to restore the best model.

The hyperparameters were optimized and their possible values are presented in table 5.2.

Table 5.2: Possible values of the hyperparameters of the neural network optimised.

Hyperparameter	Possible Values
Number of hidden layers	between 2 and 10
Number of neurons	between 30 and 256
Dropout rate	0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

100 combinations of these hyperparameters were used and the best combination was determined by maximizing the AUC of the ROC on the validation set.

5.2.1 Results

The best combination of hyperparameters which resulted from the optimization and the AUC of the ROC determined on the test set are presented in table 5.3. Figure 5.1 contains the ROC curves of the neural networks of each of the signals.

Table 5.3: Best hyperparameters and AUC of the ROC of the neural network for each signal.

Signal	Number of Neurons	Number of Layers	Dropout rate	AUC
$HH \to b\bar{b}b\bar{b}$	132	2	0.6	0.7858
$B^+ \to J/\Psi$	251	2	0.4	0.9983
$B^- \to J/\Psi$	201	6	0.6	0.9982
$ZH \to b\bar{b}\nu\bar{\nu}$	192	2	0.6	0.9611

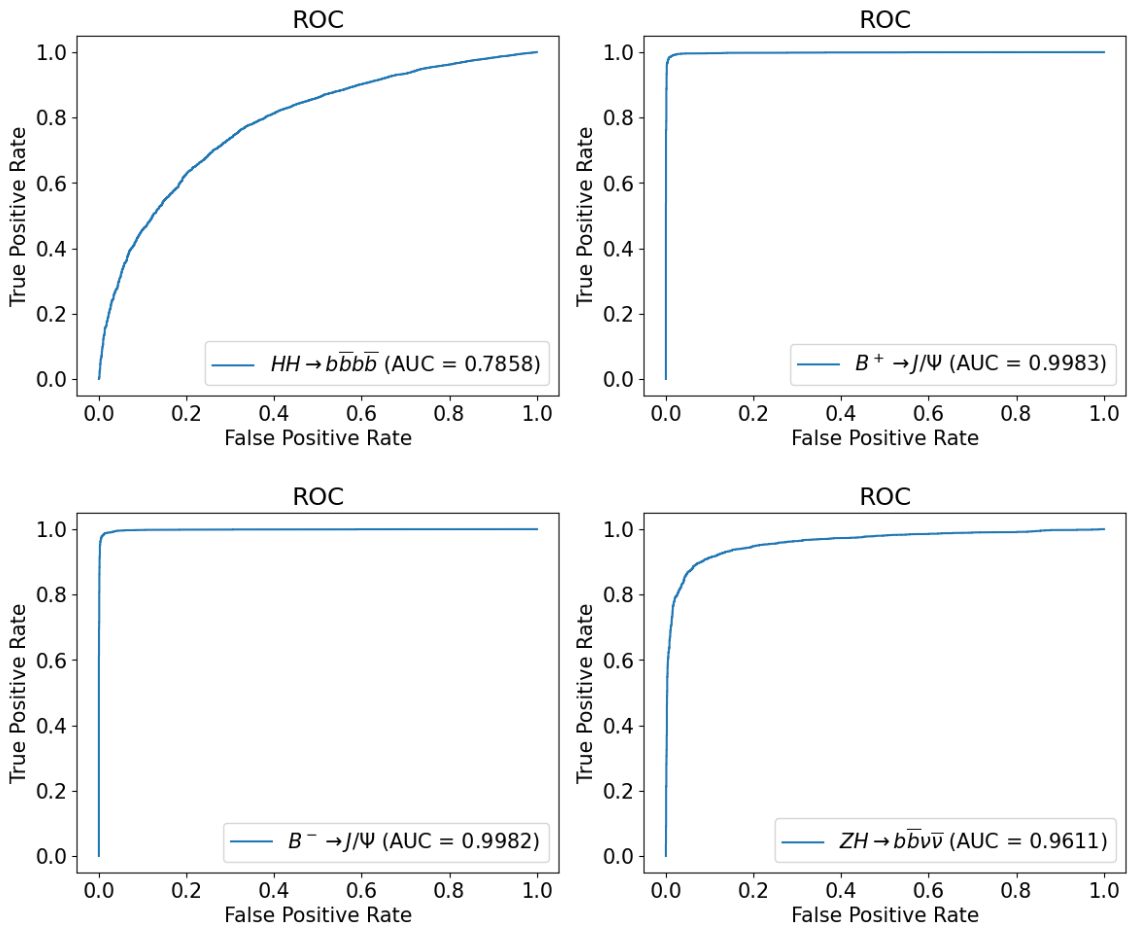

Figure 5.1: ROC curves of the neural networks trained for each of the signals.

The $HH \to b\bar{b}b\bar{b}$ signal is the hardest to separate from background out of the four signals considered. The $B^+ \to J/\Psi$ and $B^- \to J/\Psi$ are the signals easier to discriminate from background.

Comparing the AUC of the neural network with the AUC of the best feature, for all 4 signals the neural network provides better discrimination than the best feature. The difference between the AUC of the best feature and the AUC of the neural network is bigger for the $HH \to b\bar{b}b\bar{b}$ signal. The $B^+ \to J/\Psi$ and the $B^- \to J/\Psi$ signals are the ones with the least variation between the AUC of the neural network and the feature, and are also the ones with the highest AUC in both cases.

Chapter 6

Semi-supervised Anomaly Detection

Autoencoders trained only with normal events can be used for anomaly detection. This can be used for finding new physics. The autoencoder finds relations between the features in the training set, composed only by background events. Interesting events different than the ones the autoencoder was trained on will have a higher reconstruction error.

In this work, autoencoders were trained to reconstruct the background. The latent space dimension of the autoencoder was varied from 1 to the dimension of the dataset, 27.

6.1 Autoencoder building and training

The data was standardized so that the features have a mean of 0 and a standard deviation of 1, as explained in section 3.1.2.

Then, the model for each autoencoder was built. Each autoencoder was built with an input layer and output layer each with 27 neurons. For the hidden layers, dense layers with the ReLU activation function were used. The output layer had no activation function. The model was trained with the Adam optimizer and the mean squared error loss for 600 epochs. In the end of training the best weights were restored.

100 combinations of the hyperparameters presented in table 6.1 were tested.

The hyperparameters were optimized by maximizing the R^2 determined on the validation set and the ”median pruner” from Optuna was used with 10 startup trials and 10 warm up steps to stop unpromising trials.

Table 6.1: Hyperparameters optimized for each of the autoencoders and their possibles values. Number of layers is the number of layers in each of the encoder and the decoder. When batch normalization was used, after each hidden layer, a batch normalization layer was added to the model.

Hyperparameter	Possible Values
Number of layers	between 1 and 5
Number of neurons	between 30 and 256
Batch normalization	true or false

Each trial corresponds to one combination of hyperparameters. The number of warm up steps is the number of epochs after which the pruner starts to see if it should prune the trial and the number of startup trials is the initial number of trials that are carried out to the end, without being pruned. The pruner checks the loss on the validation set at each epoch and prunes the trial if the loss is worse than the median of the loss values of the previous trials at the same epoch.

6.2 Results

The R^2 was calculated using the background and its reconstruction by the autoencoder on the test set for each value of the latent space dimension and it is represented in figure 6.1.

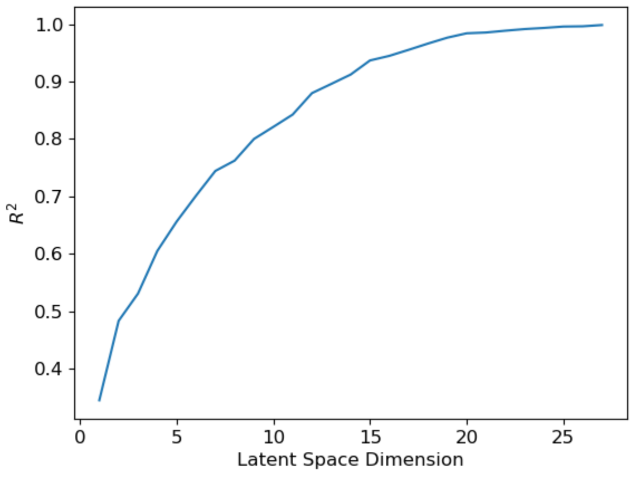

Figure 6.1: R^2 calculated with the background test set and its reconstruction of the autoencoder for each of the latent space dimensions of the autoencoder.

In order to evaluate the performance of the autoencoder in discriminating between background and signal, the AUC of the ROC was determined using the reconstruction error of the test set, constituted by background and signal events. The variation of the AUC of the ROC with the dimension of the latent space of the autoencoder is represented in figure 6.2.

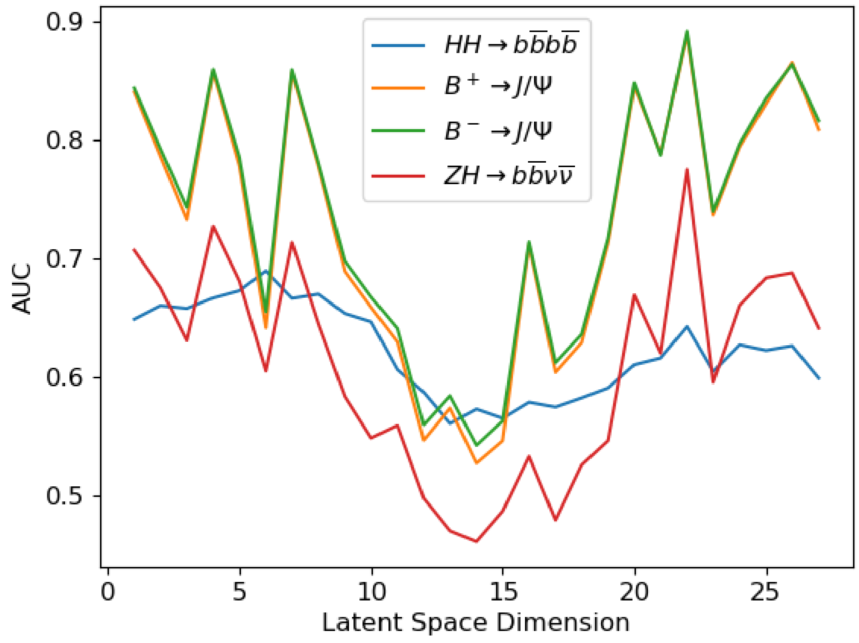

Figure 6.2: AUC of the ROC for each of the signals as a function of the latent space dimension of the autoencoder.

For each signal, the AUC is lower for the intermediate dimensions. The $HH \to b\bar{b}b\bar{b}$ signal has the least variation in the values of the AUC. All the signals follow a similar variation of the AUC of the ROC with the dimension of the latent space.

The correlation between the R^2 calculated with the background and the background reconstruction and the AUC of the ROC of each of the signals was determined and is represented in figure 6.3.

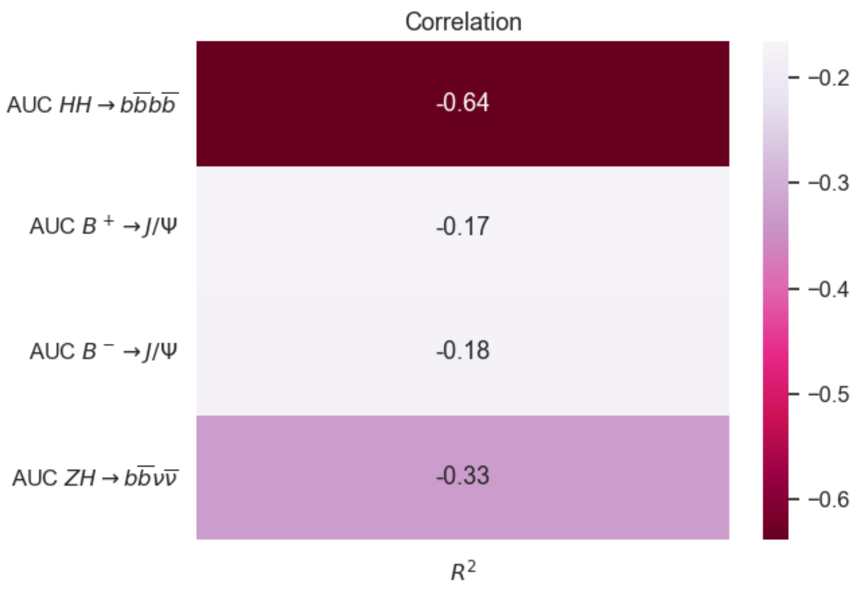

Figure 6.3: Correlation between the R^2 of the background and the AUC of the ROC of each of the signals.

From figure 6.3, we see that all signals have a negative correlation between their AUC and the R^2 and the highest correlation occurs for the $HH \to b\bar{b}b\bar{b}$, with the $ZH \to b\bar{b}\nu\bar{\nu}$ also having some correlation.

6.3 Comparison with supervised method and features

In order to compare the performance of the autoencoder, table 6.2 presents the values of the AUC of the best feature and of the neural network trained for each signal from chapter 5 and the AUC of the best autoencoder.

Table 6.2: Feature with the higher AUC and the value of its AUC, AUC of the Neural Network (NN), dimension of the latent space (zdim) of the autoencoder (AE) with the highest AUC and its AUC for each of the signals. The highest values of the AUC for each signal are marked in bold.

Signal	Best Feature	AUC Feature	AUC NN	AUC AE	Best zdim
$HH \to b\bar{b}b\bar{b}$	Number of Muons	0.5922	**0.7858**	0.6891	6
$B^+ \to J/\Psi$	p_T Jet 2	0.9953	**0.9983**	0.8885	22
$B^- \to J/\Psi$	p_T Jet 2	0.9951	**0.9982**	0.8914	22
$ZH \to b\bar{b}\nu\bar{\nu}$	p_T Jet 4	0.9456	**0.9611**	0.7747	22

For all the considered signals, the supervised neural network performs better at separating background and signal than the best autoencoder. The bigger difference between the AUC of the neural network and the autoencoder occurs for the $ZH \to b\bar{b}\nu\bar{\nu}$ signal.

Comparing the AUC of the best feature with the AUC of the best autoencoder, only for the $HH \to b\bar{b}b\bar{b}$ signal the autoencoder provides better discrimination than the best feature. For the other signals, using just one feature to separate signal and background is better.

Chapter 7

Conclusions

In this work, autoencoders trained only with background events were used to discriminate between background and signal events. Simulated Trigger-Level Analysis events were used as background and simulated events from $HH \to b\bar{b}b\bar{b}$, $ZH \to b\bar{b}\nu\bar{\nu}$, $B^+ \to J/\Psi$, $B^- \to J/\Psi$ were used as signal.

Autoencoders with different latent space dimensions were trained and their performance at discriminating between signal and background events was compared with the the best feature and a supervised neural network trained for each signal.

It was observed that the autoencoder had a better performance for lower and higher dimensions of the latent space. It was also observed that for all signals, the autoencoder performed worse at separating background from signal than the supervised neural network trained for each signal. Out of the considered signals, only for the $HH \to b\bar{b}b\bar{b}$, the best autoencoder had a better performance than just using the best feature for separating background and signal events. A negative correlation was found between the quality of the background reconstruction by the autoencoder and the autoencoder's ability to discriminate between background and signal events for the $HH \to b\bar{b}b\bar{b}$ and $ZH \to b\bar{b}\nu\bar{\nu}$ signals.

Possible future work includes trying to understand why the autoencoder has a better performance for lower and higher dimensions of the latent space and using more data statistics. Also using real Trigger-Level Analysis events and considering different signals and other features.

References

[1] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider," *JINST*, vol. 3, no. 08, 2008.

[2] I. Neutelings, "Pseudorapidity." `https://tikz.net/axis2d_pseudorapidity/`. Accessed: 2022-11-29.

[3] ATLAS collaboration, "Operation of the ATLAS trigger system in Run 2," *JINST*, vol. 15, no. 10, p. P10004, 2020.

[4] Ksenia Sorokina, "Image classification with convolutional neural networks." `https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8`. Accessed: 2022-08-12.

[5] M. E. Cross and E. V. E. Plunkett, *Physics, Pharmacology and Physiology for Anaesthetists: Key Concepts for the FRCA*. Cambridge University Press, 2nd ed., 2014.

[6] Steven Flores, "Variational autoencoders are beautiful." `https://www.compthree.com/blog/autoencoder/`. Accessed: 2022-07-27.

[7] "Experiments." `https://home.cern/science/experiments`. Accessed: 2023-01-11.

[8] ATLAS Collaboration, "Performance of the ATLAS trigger system in 2010," *Eur. Phys. J. C*, vol. 72, jan 2012.

[9] ATLAS Collaboration, "Performance of the ATLAS trigger system in 2015," *Eur. Phys. J. C*, 2017.

[10] ATLAS Collaboration, "Public atlas luminosity results for run-2 of the lhc." `https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResultsRun2#Publications_and_Conference_Resu`. Accessed: 2023-01-21.

[11] A. Boveia, “Trigger level analysis technique in atlas for run 2 and beyond.” https://indico.cern.ch/event/773049/contributions/3474303/attachments/1938074/3212450/20191105-tla.pdf, 2019. Accessed: 2022-12-21.

[12] G. Cowan, *Statistical Data Analysis.* Clarendon Press, 1998.

[13] A. Burkov, *The Hundred-Page Machine Learning Book.* 2019.

[14] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Inc., 2nd ed., 2019.

[15] F. Chollet, *Deep Learning with Python.* Manning, 2nd ed., 2021.

[16] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

[17] P. Zyla *et al.*, “Review of Particle Physics,” *PTEP*, vol. 2020, no. 8, p. 083C01, 2020. and 2021 update.

[18] M. J. W. Davide Chicco and G. Jurman, “The coefficient of determination r-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,” *PeerJ. Computer science*, 2021.

[19] “Uproot.” https://uproot.readthedocs.io/en/latest/.

[20] The pandas development team, “pandas-dev/pandas: Pandas 1.3.5,” Dec 2021.

[21] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[22] “pandas.” https://pandas.pydata.org/.

[23] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[24] “Numpy.” https://numpy.org/.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn:

Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] "scikit-learn." `https://scikit-learn.org/stable/`.

[27] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[28] "matplotlib." `https://matplotlib.org/stable/index.html`.

[29] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[30] "seaborn." `https://seaborn.pydata.org/`.

[31] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[32] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[33] "Optuna." `https://optuna.org/`.

[34] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[35] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-k_t jet clustering algorithm," *JHEP*, vol. 04, p. 063, 2008.