LIP-STUDENTS-24-11

Expanding the ATLAS Physics Reach with Anomaly Detection at Trigger Level

Ana Rita Ferreira Carvalho[1,a]

[1] *Instituto Superior Técnico, Lisboa, Portugal*

Project supervisor: Inês Ochoa

December 9, 2024

Abstract. Anomaly detetction methods are an increasingly important strategy to detect beyond the Standard Model physics and are characterized by selecting events because of their incompatibility with a learned background model. In this work, we apply anomaly detection at the ATLAS trigger level to assess if it can be used to select anomalous and possibly signal-like events. A machine-learning reconstruction model, an autoencoder, is developed to reconstruct an enhanced bias background dataset, and its performance is evaluated using a rare Standard Model simulated signal.

Keywords: LHC, ATLAS, Machine Learning, Autoencoder, Enhanced Bias Dataset

1 Introduction

1.1 Anomaly Detection and Autoencoders

The Standard Model (SM) provides a framework for understanding fundamental particles and interactions that has been remarkably predictive of experimental results over several decades. However, it remains an incomplete theory, since there are also several experimental results it cannot explain. This has motivated the search for new physics beyond the Standard Model (BSM physics). Since the discovery of the Higgs boson in 2012, one of the main objectives at the Large Hadron Collider (LHC) has been the search for new particles or interactions, but, so far, nothing has been detected yet.

Traditionally, event selection is optimized to target specific signatures of signals, performing a series of hypothesis tests based on the corresponding BSM physics models. One of the main disadvantages of this approach is that it may not be sensitive to deviations different from those expected by the class of hypothetical models that is being tested [1]. So, even if new physics is present in data, it may not be detected. Furthermore, these signals may be present in an enormous scope of cross-sections, decay channels and final states, some of which have not been covered yet, and the limited knowledge of how BSM physics should behave means that we are not able to determine a priori which phase-spaces to cover.

In order to overcome current biases and blindspots in our search coverage, a model independent approach, anomaly detection, can be adopted. These methods are optimized to detect events that differ from the majority (background) instead of selecting those which meet certain kinematic criteria.

Machine learning methods, namely autoencoders (AE), are widely used in anomaly detection. An AE is a neural-network architecture commonly used in unsupervised learning as a reconstruction model. It consists of two modules: an encoder, that compresses the dimension of the inputs to its latent space (or bottleneck) representation, and a decoder, that reconstructs the data as accurately as possible from that representation. The AE is trained to minimize the loss, or reconstruction error, that measures the difference between the input and the output. This reconstruction error can also be used as an anomaly score: anomalous events are expected to have a higher reconstruction error than normal events used during training [2]. One of the main advantages of this method is that, as an unsupervised method, it is data driven, so by training an AE using a sample dominated by SM background events, it can be used to identity events which do not meet these SM properties, since the AE will not be able to reconstruct them.

[a]e-mail: ana.rita.carvalho@tecnico.ulisboa.pt

1.2 ATLAS Trigger System

The ATLAS detector at the LHC consists of an inner tracking detector surrounded by a thin superconducting solenoid, electromagnetic and hadronic calorimeters, and a muon spectrometer [3]. It covers nearly the entire solid angle around the collision point and uses a right-handed coordinate system, where the x-axis points from the collision point to the center of the LHC, the y-axis points upward and the z-axis points in the direction of the beam. In the transverse plane, cylindrical coordinates are used, where ϕ is measured around the z-axis (the x-axis corresponds to $\phi = 0$). The polar angle θ is measured from the beam axis and is usually transformed to a pseudorapidity $\eta = -\ln\tan(\frac{\theta}{2})$.

During data taking, the LHC provides a bunch crossing every 25 ns, which corresponds to a collision rate of 40 MHz. Besides being impossible to read, store and process such an enormous volume of data, only a small subset of these events contain interesting information for physics analysis. Thus, it is crucial to select those events that are relevant for permanent storage and offline analysis.

The ATLAS Trigger and Data Acquisition System is responsible for the online processing and the selection of these events with distinguishing characteristics (energetic leptons, photons, hadronic jets, large missing energy...) [4]. It is based on a two-level event selection (trigger) system: the Level-1 Trigger (L1), which is hardware-based, and consists of custom-built electronics, and the High-Level Trigger (HLT) that uses a dedicated computing farm to run reconstruction algorithms similar to those implemented during offline analysis. The Data Acquisition (DAQ) system transports the triggered data from custom subdetector electronics through to offline processing [3].

The L1 Trigger is based on two independent systems, L1Calo and L1Muon, that use reduced-granularity (less detailed) information from the calorimeters and muon system, respectively. The L1Calo trigger identifies signatures from high-p_T (transverse momentum) objects such as electrons, photons, jets, and τ-leptons and selects events with large missing E_T, while the L1Muon trigger selects high-p_T muons. The events that satisfy the trigger menu requirements are accepted at a rate up to 100 kHz at a fixed latency below 2.5 µs [4].

Then, these events are sent to the HLT, whose software reconstructs the events at higher levels of detail than L1: information from the calorimeters, and the muon and tracking systems is used to provide better energy and momentum resolution, as well as particle identification for the reconstructed objects. This analysis can be restricted to certain regions of interest (RoIs), detector regions ($\eta \times \phi$) identified by the L1 trigger [4]. The trigger menu comprises a list of chains, where a chain corresponds to a set of L1 physics objects and a series of HLT online algorithms that reconstruct them and apply kinematic selections to them, reflecting the physics priorities of the experiment. The physics output rate of the HLT is expected to be 3 kHz on average, and the decisions are made within a few milliseconds, with the accepted events being then sent to offline storage.

The goal of this work was to develop an anomaly detection model that could be used at trigger level, namely at the HLT, to select anomalous events. For that, we optimized an AE that was trained using a background dataset of real data, as is described in the following sections.

2 Dataset

Two datasets were considered: a background dataset, with 658537 events, and a simulated SM signal corresponding to di-Higgs production with each Higgs boson decaying to a pair of b-quarks, HH→bbbb, with 99720 events. The AE was trained with the background dataset and could then be used to distinguish the signal from the background. Despite using a specific signal to test the AE, our goal was to create a model that could be applied to select generic signals. Training the AE using real data prevents relying on simulation models, which are not completely accurate.

This work follows from a previous project, where a similar model was developed using the same background dataset and a good performance was achieved. However, this model relied on track-based variables with considerable CPU cost [5]. Thus, adapting the model so that it would not depend on these variables, using low-level variables instead (such as data from the calorimeters), would significantly reduce its CPU usage and present more favourable use-case for adoption in the trigger menu, which was the focus of this project.

Thus, for each event, the data from the first and second leading jets (in transverse momentum, p_T) was used, and the variables considered were the jets p_T and the difference between the pseudorapidity of the jets ($|\eta_1 - \eta_2|$). Only the events where both jets had $p_T > 20$ GeV and $|\eta| < 2.5$ were considered.

Each jet comprised several constituents, which correspond to clusters of cells in the calorimeter where the particles deposit their energy [6]. Therefore, for each jet, the two leading p_T constituents were considered and, for each constituent, the variables used were: p_T, the number of cells (nCells) and time where the particles deposit their energy and the location of the constituent relatively to the axis of the jet, given by three variables $d\eta$, $d\phi$ and dr. In order to identify each of these variables, a double index was assigned, where the first number identifies the jet and the second the constituent. In both cases, the index 1 is assigned to the leading p_T object and 2 to the subleading one. This notation is employed throughout the following sections.

The plot of each of these variables for both the background and signal is presented in Figure 1. As an example, only the three variables of the jets and the variables from the leading constituents of the leading jets are presented.

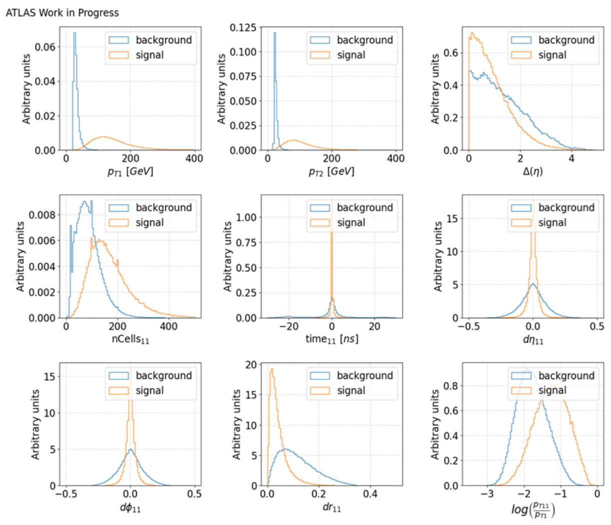

Figure 1: Distributions of input variables for background and signal datasets. All distributions are normalised to unit.

Calorimeter signals are sensitive to out-of-time pile-up, which correspond to signals from particles produced in previous or next bunch-crossings that pile up on top of those that triggered the data recording [7]. The time and energy correlation of each cluster, presented in Figure 2 shows the effect of these contributions, with two secondary peaks appearing at ± 25 ns, which are clearer at higher energies.

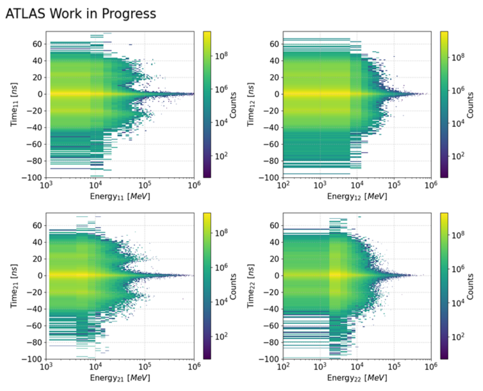

Figure 2: Cell time as a function of the cell energy.

2.1 Enhanced Bias Mechanism

In order to measure the anticipated rate of events passed by a given HLT algorithm, in principle, an unbiased sample of events is needed to obtain the selection rate for that chain. However, by definition, the purpose of most trigger algorithms is to select rare events of particular interest, whose cross sections are much smaller than the total inelastic cross section. This means that an enormous sample of zero bias events would be needed to perform these calculations. To avoid this problem, enhanced-bias (EB) datasets can be used instead [5].

EB datasets correspond to a collection of events selected by the L1 trigger system constructed such that higher energy and object multiplicity bias is removable with event weights. This dataset is collected using a selection of representative L1 triggers of all physics-objects types and p_T ranges capable of producing a compact dataset with enough statistical power to assess the trigger rate on the data typically performed by the HLT. Each chain that is used to take EB datasets introduces a specific prescale to these events, that corresponds to the number of rejected events for every accepted one [8].

The recorded events are only biased by the L1 system, and the EB dataset reflects its configuration and the beam parameters of the LHC at the time it is taken, so that a new dataset is required when any of these changes significantly [5]. Thus, EB datasets can be used to test updates to the software release and trigger menu before they are deployed on the live system and evaluate the trigger performance, calculating rates of individual chains, or the entire menu.

As was mentioned, EB data is taken with an invertible trigger menu, which means that a single weight per event can be calculable to correct for the prescales used to collect the dataset, restoring the zero-bias equivalent [8]. The EB weight of the event e, $w_{EB}(e)$, can be obtained from the equation:

$$\frac{1}{w_{EB}} = 1 - \prod_{j=1}^{EB\,chains} \left(1 - \frac{r_{je}}{p_j}\right) \tag{1}$$

The product runs over the EB chains used to take the dataset, with r_{je} the decision before the application of any prescale (r_{je} assumes a binary value) and total prescale p_j (considering both L1 and HLT prescales and assuming constant prescales during the data taking period) [5].

In this case, the background dataset corresponded to an EB dataset from 2022. Hence, there was an EB weight associated to each event. These weights assume discrete values since each chain introduces a specific prescale. However, since the value of the prescales is highly variable for different trigger components, the weights are distributed over a wide range of values. In Figure 3, the weights are represented as a function of the p_T of the leading jet. As was expected, the higher weights correspond to the events with lower p_T jets, since these are less represented in the EB dataset, so a higher correction is needed.

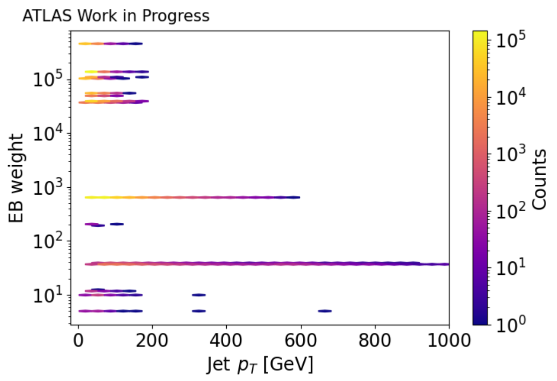

Figure 3: EB weights as a function of the leading jet p_T

Figure 4 represents the normalised leading p_T distribution with and without the weights applied. Including the weights in the p_T distribution shows that the EB weights, besides changing the scale of the distribution, reintroduce its physical meaning, by removing the influence of the trigger selection on the data. Hence, the weighted p_T distribution corresponds to a typical smoothly falling leading jet distribution, where, once again, it is visible that the high p_T jets lose representation when the weights are applied. The distribution is shown for a range of up to 850 GeV in leading jet p_T, where there are enough statistics for a useful visualisation, but the events extend to values of 1000 GeV.

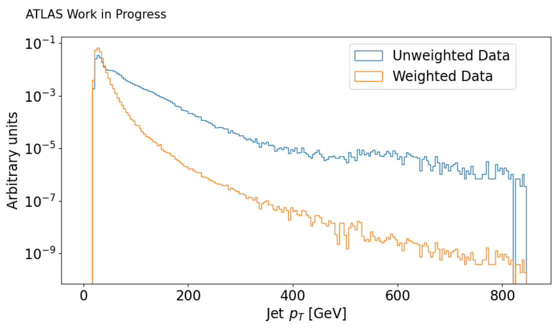

Figure 4: Distributions of the leading jet p_T with and without considering EB weights normalised to unit.

These weights can also be considered during the AE training process, as is shown in the next section.

3 Autoencoder

The AE was implemented using TensorFlow/Keras. Several aspects influence the AE performance, namely the choice of input features, the standardization of input data and the AE architecture (number of hidden layers, latent space dimension and training hyperparameters), which were optimized to reconstruct the background as closely as possible. The main conclusions about the AE implementation, and the influence of each of these factors, are presented in the following sections. Before implementing the model, the background dataset was split into three sets: training and validation (which are used during training) and the test set (used after training to assess the AE performance).

3.1 Standardization

Standardization of the input data is an important step before the training process. The training and validation sets contain different features, with different units and ranges of values, which can affect the model's performance, as features with larger ranges may dominate the training. Therefore, standardizing the input features balances the inputs, keeping the weights and activations within a reasonable range and reduces the impact of any outliers in the data, reducing the network's sensitivity to changes in inputs or weights [9].

In this case, all the sets were standardized using scikit-learn's StandardScaler, which transforms each of the sets using the equation $x \longrightarrow \frac{x-\overline{x}_{train}}{\sigma_{train}}$.

3.2 Variable Transformations

The first iterations of the AE training showed that the AE struggled to reconstruct some variables with sharp features, such as the p_T distribution for both jets and constituents, as commonly seen in literature [10]. Figure 5 presents an example of these results, for the subleading jet and the subleading constituent of the subleading jet. In order to minimize this problem, the following variables were used instead: $\log\left(\frac{p_{T1}}{p_{T2}}\right)$ (for the jets) and $\log\left(\frac{p_{Tij}}{p_{Ti}}\right)$ (for the constituents), (where, as mentioned previously, i identifies the jet and j the constituent, with both being either 1 or 2).

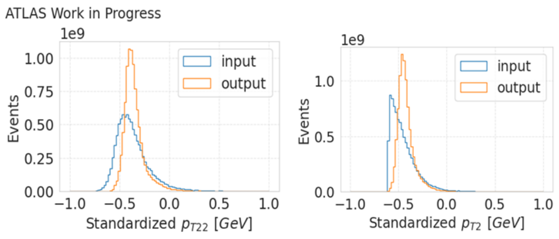

Figure 5: AE poor reconstruction of p_{T2} and p_{T22}.

This transformation significantly improved the reconstruction of the variables associated to the constituents; however, that was not the case for the jets, as is shown in Figure 6. Hence, the p_T of the jets was once again considered in separate.

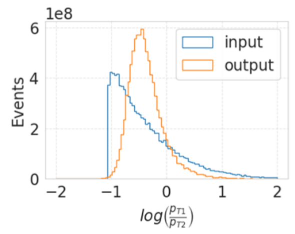

Figure 6. AE poor reconstruction of $\log\left(\frac{p_{T1}}{p_{T2}}\right)$.

3.3 AE Architecture and Performance

Several different AE architectures were tested, considering different combinations of the parameters presented in section 3. The goal was to find the version of the AE that better reconstructed the input features (lower validation loss), and whose performance was stable when the training was repeated with different network initializations and using different training and validation sets.

As was mentioned in 1.1, the encoder contains several hidden layers and compresses the input (with dimension 27, in this case) to its representation on the latent space, which corresponds to a latent layer, whose dimension needs to be optimized. The decoder mirrors the structure of the encoder, with the same number and dimension of layers, but in the reverse order, and with an output layer that recovers the input's dimension.

In this case, we started by testing an AE with three hidden layers and the latent space with dimension 14, which did not give a good reconstruction of the data. Thus, we increased the dimension of the latent space by successively adding two and testing the results of the reconstruction after each increase. For each latent space dimension, an encoder structure with two and three hidden layers was tested. As expected, the results were consistently better when three layers were added, but adding a fourth layer did not improve the results. The dimension of the latent space was increased until 20, after which the performance of the model stabilized. Therefore the final version of the model contains three hidden layers, with dimensions 25, 24 and 22, and the latent layer with dimension 20.

For all training iterations, the ReLU activation function is applied to all layers, except the last, the model is compiled using the accuracy as the weighted metrics and the Adam optimizer. The batch size is 1024 (a batch size of 512 was also tested, but it was not found any clear influence on the results). The reconstruction loss function corresponds to the mean squared error between the input and the reconstructed values of the dataset. The training process is monitored using the reconstruction loss of the validation set for 300 epochs, with an early stopping when this value does not decrease within 15 epochs (in this case, the best weights are restored).

The EB weights of each event were also added to the AE. During the training process, the loss of each sample of the batch is rescaled by the corresponding element in the sample weight vector, so that samples with higher weights will have a greater influence on the total loss and the model will prioritize minimizing the loss of these samples.

The results for the final version of the AE are presented in Figure 7, where the data of the jets and the subleading constituents of the subleading jets from the test dataset is represented before and after reconstruction. While the optimization of the model led to a very good reproduction of some of the variables, others, namely the jets p_T and nCells, were still not perfectly reconstructed.

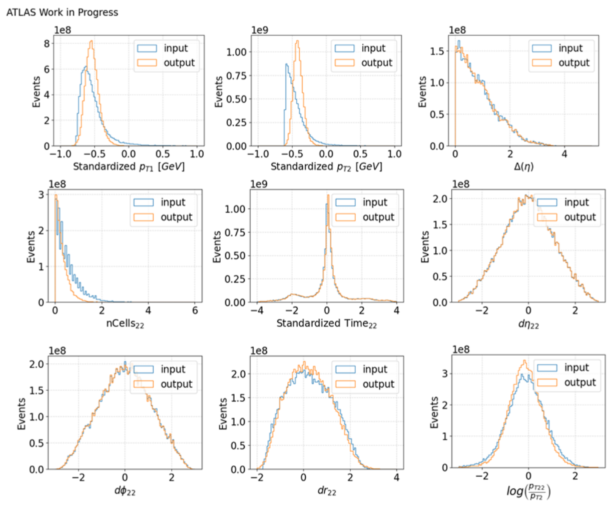

Figure 7: AE reconstruction of the jet variables and the subleading constituent of the subleading jet from the test dataset.

One possible explanation for this would be that the duration of the training was insufficient for the model to learn all the features of the variables. The evolution of the training and validation loss function shows that it is common for both of these to stabilize for a certain number of epochs, before start decreasing again. An example of this behaviour is presented in Figure 8. Imposing an early stopping in the training could prevent the model from reaching its maximum performance. So, to test if increasing the duration of training would improve the results, both the number of epochs and the patience of the model (that controls the early stopping) were increased. Different values of both parameters were tested, up to 2000 epochs with 100 epochs of patience, but the model still struggled to reconstruct the same variables as previously (the reconstruction of the jets p_T is presented in Figure 9, which confirmed that the training hyperparameters that were previously used did not significantly compromise the AE performance).

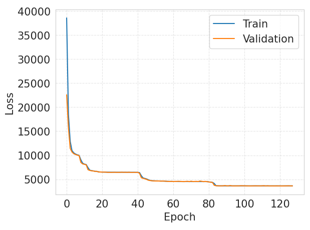

Figure 8. Example of training and validation loss.

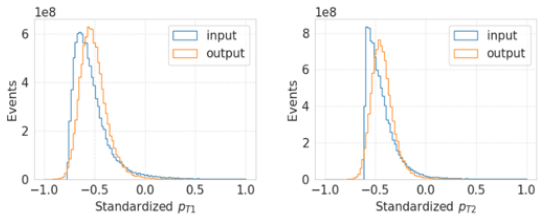

Figure 9: Reconstruction of p_{T1} and p_{T2} after increasing the duration of training.

The model was then used to try to reconstruct the signal (which was also standardized), and the results are presented in Figure 10. As was expected, in general, the AE was not able to reconstruct the signal, although for some variables, such as time, the distribution of the reconstruction was closer to the original distribution than expected.

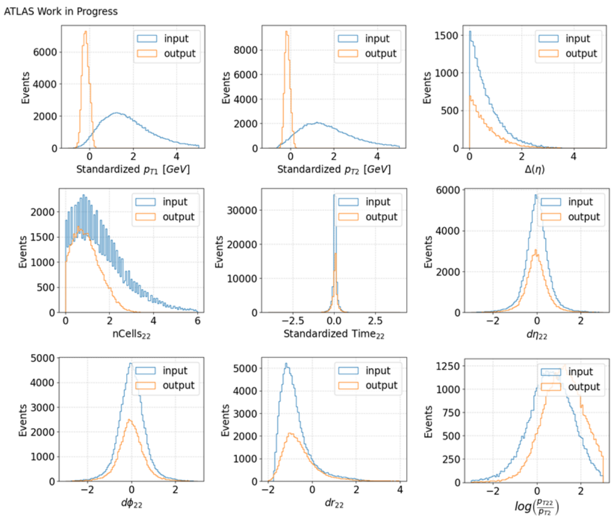

Figure 10: AE reconstruction of the signal dataset.

3.4 ROC Curve and Background Rate

The results from the background and signal reconstruction for the AE can be used to obtain the ROC curve of the model. A ROC (Receiver Operating Characteristic) curve is a common method to assess the performance of classification models, although it can also be applied to classifiers that return some confidence score or a probability of prediction, such as decision trees or neural networks [11]. In this case, as this was a case of unsupervised learning, the ROC curve was obtained using the logarithm of the reconstruction loss (mean squared error between the input and the output), $\log_{10}(mse)$, as the anomaly score for each event in both datasets. The plot of the anomaly score for both datasets is presented in Figure 11. Although the highest values for the anomaly score correspond to the signal, as intended, the fact that some of the background variables were not perfectly reconstructed increases the anomaly score of the corresponding events. This also increases the overlap between the anomaly scores distributions, compromising the model's ability to distinguish signal from background, which is reflected in the ROC curve.

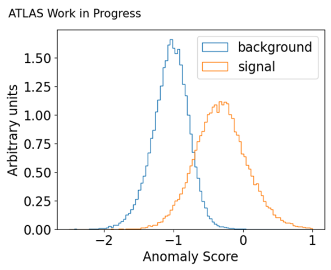

Figure 11. Anomaly score for the signal and background reconstruction. Distributions are normalised to unit area.

The ROC curve can be obtained by plotting the background rejection, defined as the inverse of the false positive rate (proportion of background examples predicted incorrectly) against the signal efficiency, or true positive rate (proportion of signal examples predicted correctly), at various thresholds settings. A better classifier maximizes the two parameters simultaneously, which corresponds to a curve closer to the upper right corner of the ROC space. In this case, the signal efficiency is obtained simply by calculating the ratio between the number of events that are above the threshold and the total number of events. However, for the background dataset, the sum is done on the weights of the events instead.

The ROC curve is presented in Figure 12 and confirms what the distribution of the anomaly scores already suggested: the model presents some difficulties distinguishing the two datasets. Since the distributions of the anomaly scores are so close, for lower thresholds, there are still a a significant number of signal events considered as background, so the signal efficiency doesn't stabilize in 1.

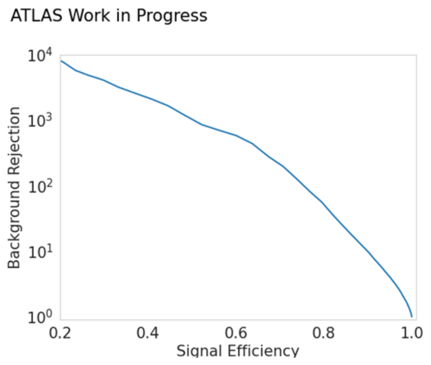

Figure 12: ROC curve of the AE.

This data can also be used to predict the background rate of a trigger chain based on this model, for a certain threshold, which can be obtained using:

$$R = \frac{\sum_{e=1}^{N} w_{EB}}{\Delta t}, \tag{2}$$

where R is the rate (in Hz), Δt is the time period over which the EB dataset was collected (21600s were considered), the sum runs over all the events whose anomaly score is higher than the threshold and w_{EB} is the enhanced bias weight of those events, which gives the effective number of events passed by the chain [5]. The background rate distribution, along with dynamical restrictions of the operations and trigger menu, sets the threshold for the anomaly score.

The distribution of the rate as a function of the signal efficiency is presented in Figure 13. For lower thresholds, both signal efficiency and background rate are high, which is expected. However, the background rate is still considerable for lower thresholds (or lower signal efficiencies), which confirms that the model does not reject all the background.

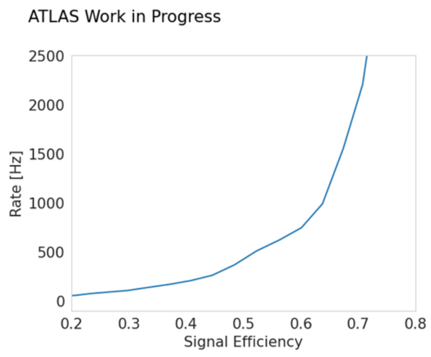

Figure 13: Background rate as a function of signal efficiency.

4 Conclusion and Next Steps

In conclusion, it was possible to develop an unsupervised learning model that used low-level input variables that are readily available at HLT. However, that raised new challenges in its optimization and affected its performance, compared to a model that used track-based variables. Namely, despite testing different architectures, input variables and training hyperparameters, the model struggled to reject enough background, which kept the trigger background rate high.

Thus, it would be important to test the AE on another EB dataset (taken in a different time period, with different configurations) to verify if the problems in the reconstruction of some variables encountered with this dataset were also present and compare the background rate obtained with the new dataset to the one obtained in this project.

Moreover, this work proved that changing the input variables of the AE affected its performance, which may suggest that this ML model might not be the most suitable for this data. An AE, like other typical ML algorithms, was designed for fixed dimensional data instances. Since the number of constituents in each jet is variable, and it would be desirable to use the data from all constituents, using an architecture that doesn't require fixed-size inputs could improve the results. One possible alternative would be using Deep Sets, where inputs or outputs of the model are permutation invariant sets instead of fixed dimensional vectors [12]. Hence, since jets can be seen as variably sized sets invariant under reordering of their constituents, using an architecture within this framework might be a more appropriate solution. Future work can focus on the development of these other architectures in order to test this hypothesis.

Acknowledgments

I would like to thank my supervisor, Inês Ochoa, for always being available to help and for all the guidance and to the ATLAS group at LIP for welcoming me during this project. I also thank LIP for the organization of the 2024 Summer Internship Program.

References

[1] R.T. D'Agnolo, A. Wulzer, Phys. Rev. D **99**, 015014 (2019)

[2] M. Farina, Y. Nakai, D. Shih, Phys. Rev. D **101**, 075021 (2020)

[3] ATLAS Collaboration, JINST **19**, P05063 (2024), 2305.16623

[4] ATLAS Collaboration, *The ATLAS Trigger System for LHC Run 3 and Trigger performance in 2022* (2024)

[5] ATLAS Collaboration, *Trigger monitoring and rate predictions using Enhanced Bias data from the ATLAS Detector at the LHC* (2016)

[6] ATLAS Collaboration, The European Physical Journal C **77** (2017)

[7] ATLAS Collaboration, *Improving topological cluster reconstruction using calorimeter cell timing in ATLAS* (2023)

[8] R. Keyes, Journal of Physics: Conference Series **898**, 032008 (2017)

[9] C. Ranjan, *Understanding Deep Learning Application in Rare Event Prediction* (2020), ISBN 979-8586189561

[10] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari, T. Tomei, Phys. Rev. D **107**, 076017 (2023), 2211.10295

[11] A. Burkov, *The Hundred-Page Machine Learning Book* (2019)

[12] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, A. Smola, *Deep sets* (2018), 1703.06114