Anomaly Detection in High Energy Physics using CWoLa and Autoencoders

Iván Díaz[1] and **Rafael Espinosa**[2]

November 2023

[1] Tecnólogico de Monterrey Campus Santa Fe
[2] Tecnólogico de Monterrey Campus Querétaro

Abstract

We present a novel approach in the search for new physics Beyond The Standard Model (BSM). Traditional model-dependent methods may be limited due to the bias they have towards the specified model, which means they might miss some unexpected anomalies.

We propose a strategy focused on Machine Learning, trying to find anomalies directly from the raw data. We explore the integration of the Classification Without Labels (CWoLa) algorithm with autoencoder neural networks. We analyze the Signal detention performance of various combinations of the CWoLa with various autoencoder architectures, such as Deep and Sparse models. By adjusting some hyperparameters such as the batch size, the activation functions, and the regularizers, we look to further improve the neural network's discrimination capability to differentiate between Signal-like and Background-like events.

Contents

1 Introduction

For several years, the pursuit of new physics Beyond The Standard Model (BSM) has been one of the main topics of discussion in the field of HEP (High Energy Physics). Most of the traditional searches for new particles, predominantly model-dependent ones, consist of simulating the signal S and the Standard Model background B [1]. Choosing theoretical models for S and B generates a Bias towards these specific models which means that they might miss unexpected anomalies that are not included in the assumptions made by the models.

While these methodologies proved to be effective in the past, their own limitations have led physicists to pivot towards new and innovative techniques for anomaly detection for the discovery of new physics, one of them being Machine Learning, which hopefully will be able to answer unanswered questions such as "What is dark matter, and what is it made of?"[2], "Why is the Higgs boson so light, when theoretically it was expected to be extremely massive?"[3], or even yet, "Is the Boson Higgs related to dark matter?".

Machine learning is a great alternative to model-dependent searching because instead of relying heavily on simulations and assumptions, it aims to find anomalies directly from raw data, which capitalizes on the ability of machine learning to identify patterns without predefined models.

Recent advancements in neural network applications, such as *ANOmaly detection with Density Estimation* (ANODE) [4], Autoencoders and Classification Without Labels (CWoLa) have shown promising results in anomaly detection. This paper extends the work of Michelle Calzada and Rafael Espinosa [5] on CWoLa. It also builds upon a key concept introduced in 'Classification without labels: learning from mixed samples in high energy physics' [6] as they precisely articulate, 'It is worth emphasizing that the CWoLa framework can be applied to a huge variety of classifiers without modification to the training procedure, by simply training on mixed event samples instead of on pure samples.'

Thus we aim to explore the integration of the CWoLa algorithm for event labeling and Autoencoder neural networks as event classifiers. Our focus will be on the experimentation of various hyperparameters and Autoencoder architectures such as Deep and Sparse, to enhance the sensitivity of the neural network for the detection of Signal-like events.

2 CWoLa

The CWoLa algorithm as developed by Eric M., Benjamin Nachman, and Jesse Thaler [6] aims to enable the creation of robust classifiers directly from raw data in collider physics.

CWoLa has proven to be effective at classifying events into background (QCD events) and signal BSM events. This weakly supervised algorithm does not rely on any simulation to differentiate between signal S and background B events. The process begins with 2 sample sets, M_1 and M_2 as illustrated in Figure 1, where M_1 has a greater fraction of signal events and M_2 a greater fraction of background events.

A model is then trained to classify events as coming either from M_1 or M_2 (labeled as 0 and 1 respectively), let $\overrightarrow{x}$ be the observables that are of use to distinguishing signal from background events. The optimal classifier used to distinguish examples drawn from $pM_1(\overrightarrow{x})$ and $pM_2(\overrightarrow{x})$ (the probability distributions of $\overrightarrow{x}$ for the set M1 and M2 respectively) is the likelihood ratio $L_{M_1/M_2}(\overrightarrow{x}) = pM_1(\overrightarrow{x})/pM_2(\overrightarrow{x})$. The same optimal classifier to distinguish events drawn from the probability distributions of S and P is the likelihood ratio $L_{S/B}(\overrightarrow{x}) = pS(\overrightarrow{x})/pB(\overrightarrow{x})$

We can relate these two likelihood ratios algebraically:

$$L_{M_1/M_2} = \frac{pM_1}{pM_2} = \frac{f_1 ps + (1-f_1)pB}{f_2 ps + (1-f_2)pB}$$

$$= \frac{f_1 L_{S/B} + (1-f_1)}{f_2 L_{S/B} + (1-f_2)}$$

which is a monotonically increasing rescaling of $L_{S/B}$ as long as $f_1 > f_2$, this means that $L_{S/B}$ and L_{M_1/M_2} define the same classifier, and so by training a classifier to correctly distinguish events either coming from the set M_1 (Mostly signal) and M_2 (Mostly background) we can use the exact same classifier to distinguish single events from either being S or B.

This algorithm enables greater searches for anomaly events in HEP because it manages to yield the same decision boundaries as a pure strong supervised model, as substantiated in "Classification without labels: learning from mixed samples in high energy physics" [5]. We can train the model without the need to know the data fraction found within each set or information about the individual signal and background labels.

In the quest for the search of BSM events using CWoLa, it is theorized that the signal will be detectable within a specific range of invariant mass. Given a hypothetical resonance mass and its corresponding width, we define a signal region (where the set M_1 is extracted) by choosing events within a window around the hypothesized resonance mass. The upper and lower sideband regions (where the set M_2 is extracted) are created by selecting events in two zones around the signal region.

To differentiate a potential signal event from the background, we use additional features y that are orthogonal and transversal to the resonance mass [7] an illustration can be seen in Figure 1.

With CWoLa we can classify events as being more Signal-like or Background-like, we cannot say with full certainty what type of event we are dealing with as we would on a strongly supervised model, but one of the main benefits CWoLa has over strong supervised models is that we do not need pure perfect labeled examples for it to work, in fact, we only need a large enough data set (LHC2020 Olympic's data set is made up of 1,100,000 event samples) and the correct admixture of examples previously mentioned. We don't even need to know what the fraction of the mixtures that we have is, this makes CWoLa a pow-

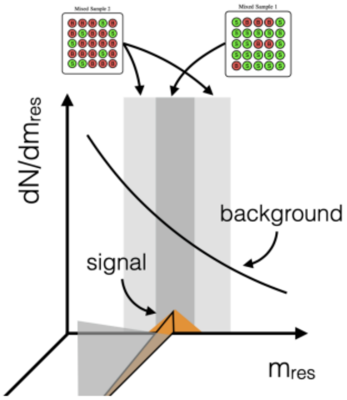

Figure 1: The CWoLa model is trained to distinguish between two sets M_1 which is mostly signal and M_2 which is mostly background. A hypothetical invariant mass range specifies the signal region, the sideband regions are located around this signal region. After training, the classifier is capable of distinguishing signals by additional features y.[8]

erful tool for anomaly detection in HEP.

3 Anomaly Detection with Auto-encoders

Autoencoders (AE's) are a class of unsupervised neural networks that try to learn a function $h_{W,b}(x) \approx x$ that will minimize the reconstruction loss of the output, or in other words it is trying to learn an approximation to the identity function that makes the output $\tilde{x}$ as similar as possible to x.

The architecture of a simple autoencoder can be broken down into three core components:

- The **Encoder**: It compresses the input.
- The **Latent Layer / Bottleneck**: Where the maximum compressed representation resides.
- The **Decoder**: It reconstructs the input up from the compressed representation found in the Bottleneck.

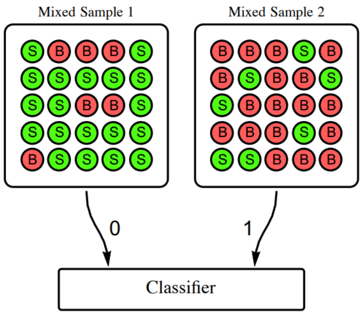

Figure 2: An illustration of the CWoLa algorithm, instead of being trained to classify directly signal and background events, it is trained to distinguish data coming either from M_1 or M_2. [5]

The architecture varies with different types of autoencoders, such as Deep, Generative, Variational, or Sparse autoencoders, each with its unique approach to encoding and decoding of the data. An illustration of a simple autoencoder architecture can be seen in Figure 3.

One might question the utility of an autoencoder if as one could think that the neural network could simply learn to mimic (copy) the output $\tilde{x}$ to the input x, which would be effectively useless. However, by constraining the properties of the hidden layers we will force the latent layer to generalize the data and actually "learn" about the key features that make up the data and weigh them appropriately.

We can put up some of these constraints on the hidden layer by decreasing its size (amount of neurons) and/or by constraining which neurons activate on each iteration during training. By constraining the size of the hidden layer (h) and making it smaller than the size of the input layer $n_h < n_{in}$ the neural network learns latent representations of the given data with a smaller set of neurons, an improvement over this simple structure is a Deep Autoencoder, as seen in Figure 4. A deep autoencoder as its name implies, is deeper in the sense that more layers, each with a progressively decreasing amount of neurons are added. This means that each following layer will refine the extracted features from the previous bigger layer which leads to a more robust discrimination of

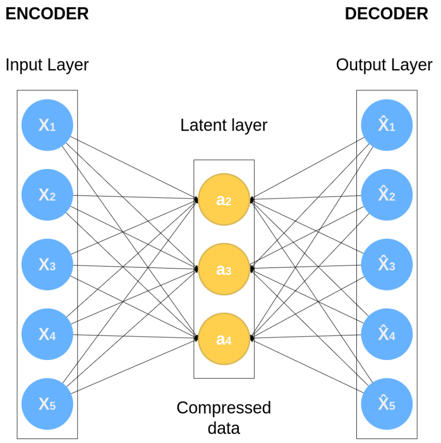

Figure 3: Simple autoencoder neural network architecture made up of an Input layer, being the same vector size as any given data input. The latent layer, which is a lower dimension representation of the input data, and finally the decoder which aims to reconstruct the compressed data back into the input data with the least amount of reconstruction error as possible.

the least important features of the inputed data.

Alternatively, by introducing sparsity through regularization of the neural activation, we encourage the network to learn how to encode and decode the input by only activating a small number of neurons in each hidden layer, an Autoencoder using this type of constraint is called a Sparse Autoencoder as seen in Figure 5.

The key difference between constraining via size limitation and activation regularization is that during training using a Deep architecture, all neurons are generally engaged for every observation. A Sparse autoencoder will be forced to activate or deactivate certain neurons depending on the input data and the selected regularization function, this has the benefit of further reducing the capacity of the autoencoder neural network of memorizing the input data whilst at the same time not limiting its capacity to extract key features from the data.

Autoencoders have proven to be a highly effective tool for anomaly detection. By their very nature autoencoders don't make direct binary decisions; instead, they focus on minimizing the reconstruction error of the given loss function such as *MSE* (Means Squared Error) or *MAE* (Mean Absolute Error). However, by leveraging their robust pattern recognition capabilities, we can train an autoencoder to become proficient at identifying specific kinds of data (such as dog images, patterns in normal-functioning human hearts, or SM events). This training enhances the autoencoder's ability to discriminate against any other kind of data (i.e anomalies).

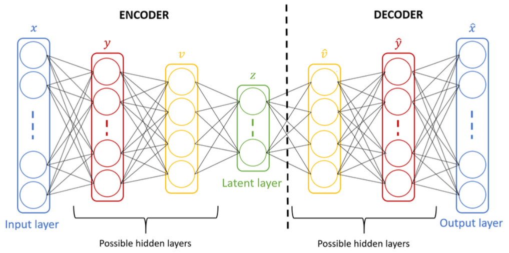

Figure 4: Deep Autoencoder neural network architecture. The encoder progressively compresses the data into a lower dimensional vector (latent representation) capturing the data's essential features. The latent layer, represented by the central layer, holds the most compressed form of the data. The decoder aims to reconstruct the original data out from the latent features as accurately as possible [9].

Therefore, when an autoencoder trained exclusively on SM events (background) receives a BSM event (signal) it is likely to output a higher reconstruction error because the autoencoder will struggle to reconstruct these unfamiliar events as it has been trained to recognize the background events.

In the subsequent Architecture section, we will dive deeper into the specifics of these Autoencoder structures, detailing the layers, hyperparameters, and design choices employed.

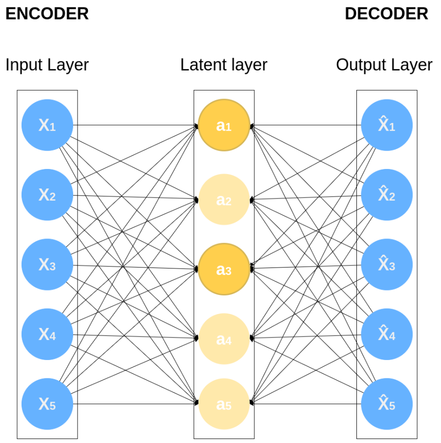

Figure 5: Sparse autoencoder neural network architecture. A specialized type of autoencoder that utilizes regularization in the hidden layers to lower their dimensionality by disabling some of the layer's neurons eventually creating a bottleneck layer.

4 Methods

4.1 Data

The data set used for the experimentation throughout this paper was the one used for the LHC Olympics 2020 community challenge.[10] The data set consists of a million Standard Model (SM) events, in which each event features a pair of jets produced via the strong interaction. In addition, there are one hundred thousand events where a BSM particle (Z') decays into two new BSM particles (X and Y), both decaying into two SM particles, a quark-antiquark pair. The previously described interaction can be appreciated in Figure 7. The masses of the Beyond the Standard Model (BMS) particles are $Z' = 3.5 TeV$, $X = 500 GeV$, and $Y = 100 GeV$. All of the events were generated using $PYTHIA$ 8.219 [11, 12], and $DELPHES$ 3.4.1[13] with default settings and without including any pileup or multiparton interactions.

Each jet is characterized by its 3-momenta invariant masses, which are represented by p_{xj}, p_{yj}, and p_{zj} along with the N-subjetiness variables τ_1, τ_2, τ_3. τ_N or N-subjetines indicates if a Jet has any sub-jets and if it does, it tells us how many branches come from the main Jet. After the reconstruction of a candidate J jet using an algorithm like the anti-kt algorithm [14] and the reconstruction of a N number of possible subjects are identified, we can calculate the N-subjetiness (τ_N) as:

$$\tau_N = \frac{1}{d_0} \sum_k p_{T,k} min \Delta R_{1,k}, \Delta R_{2,k}, \ldots \Delta R_{N,k}$$

where Pt, k is the transverse momenta and $\Delta R_{J,k} = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$ is the distance in the rapidity-azimuth plane between the sub-jet J and the constituent particle k. The normalization factor d_0 is taken as [14]:

$$d_0 = \sum_k p_{T,k} R_0$$

where R_0 is the jet radius used in the jet clustering algorithm. Thanks to the results retrieved from the paper "A Deep Analysis of the CWoLa Algorithm for Anomaly Detection in High Energy Physics" [5] we will only be using the subjettines ratio $\tau_{21} = \frac{\tau_2}{\tau_1}$ because they found that the most signal sensitive variable was indeed τ_{21} as shown in the ROC curves in Figure 6

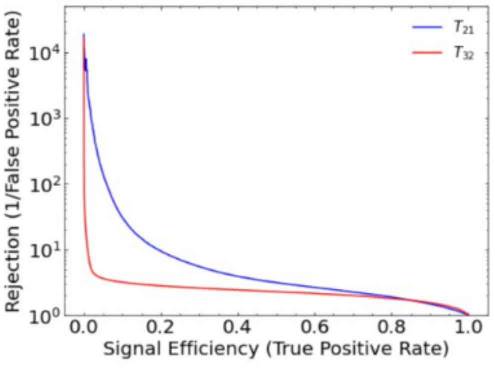

Figure 6: A receiver operating Characteristic (ROC) curve for τ_{21} and τ_{32} where the sensibility to signal of $\tau 21$ is superior to that one of $\tau 32$ [5].

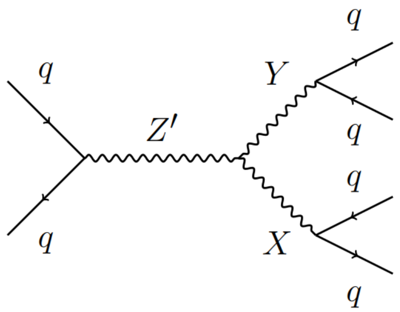

Figure 7: Feynmann diagram of the particle decay into two BSM particles, which in turn decay into two hadrons which are detected as jets.

4.2 Event Classification with CWoLa and Autoencoders

For the development, training, and validation of the proposed models, TensorFlow 2.0 was employed as the primary framework.

This section will explore the chosen architectures of the Deep autoencoder and the Sparse autoencoder. We will discuss the parameters that were adjusted during the experimentation phase and analyze the outcomes these modifications produced.

4.2.1 Shared Characterisitcs

All of the neural networks were trained during 20 epochs whilst using a batch size of 200 events. The models were all optimized using the Adam algorithm and the ReduceLROnPlateau callback with a patience of 5 and a factor of 0.2.

The signal region was defined by $m_{jj} \in [3.3, 3.7] TeV$ and the side band regions were defined as $m_{jj} \in [3.1, 3.3] \cup [3.7, 3.9] TeV$.

4.2.2 Sparse Autoencoder

This segment of the experimentation primarily focused on evaluating three different-sized sparse autoencoders.

- A **1-hidden layer** sparse autoencoder, comprising of an input layer, a latent layer, and an output layer.
- A **3-hidden layer** sparse autoencoder.
- A **7-hidden layer** sparse autoencoder.

Each layer of all 3 modules contains 4 neurons. We use ReLU as the activation function in every layer except for the output layer, where we chose the sigmoid activation function. The choice of ReLU as the main activation function was made because it has been proven to successfully avoid the vanishing gradient problem as discussed in "A Deep Analysis of the CWoLa Algorithm for Anomaly Detection in High Energy Physics"[5]. We choose the Sigmoid activation as the output function to aid in the normalization of the data and keep all the outputs as a probability between 0 and 1 before rebuilding the representation of the features to the original 4-dimensional vector.

The hidden layers have the Keras l1 regularization function as their main activation regularizer. It constrains the neural network by adding the sum of absolute values of the weights to the loss function of the layer, expressed as:

$$L = (x, \tilde{x}) + \lambda \sum_i |a_i^{(h)}|$$

It encourages sparsity by selectively activating regions of the network.

4.2.2.1 1-hidden layer Sparse Autoencoder

The performance of this model was very poor. Across various CWoLa signal injections, this model demonstrated its lack of capacity to correctly generalize key features of background events. This hindered its ability to distinguish background from signal events. We arrived at this conclusion because the reconstruction loss from signal events is almost as low as that from background events, these results can be observed in Figure 8.

4.2.2.2 3-hidden layer Sparse Autoencoder

Even though we increased the amount of hidden layers from 1 to 3, whilst at the same time increasing

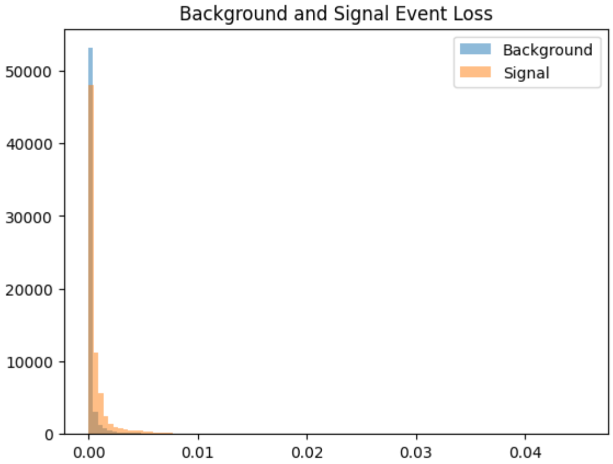

Figure 8: Reconstruction loss of 1 layer sparse autoencoder.

the amount of layer regularization in the hidden layers. We got a very similar result. It is not surprising as very little of the 1-hidden layer architecture was modified.

4.2.2.3 7-hidden layer Sparse Autoencoder It yielded by far the best results. We used this model architecture to plot the Sparse AE in the ROC curves shown in Figure 11. We went from using a value of $1e-6$ for the lasso regularization in the first hidden layer, all the way up to $1e-3$ for the latent layer.

4.2.3 Deep Autoencoder

This segment focuses on the evaluation of the implications of making use of the Mean Squared Error (MSE) and Mean Absolute Error (MAE) as loss functions in the context of anomaly detection.

4.2.3.1 MAE vs MSE Before showcasing the experimental results, it's essential to understand which loss function would hypothetically yield better results on anomaly detection tasks.

MAE computes the average of the absolute differences between the predicted and observed values, represented as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i|$$

where y is an observed value, and $\tilde{y}$ is a predicted value. The absolute difference makes it so that negative errors are also taken into account.

On the other hand, **MSE** simply squares the distance of the difference of y and $\tilde{y}$ such that all of the results will turn out to be positive.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

This squaring of the function makes it so that higher errors (distances between the observed and the predicted values) weigh more in the metric than lower ones, this could prove advantageous as the MSE loss function should be more sensitive to outliers in data sets. In the context of autoencoders, this could prove to be beneficial as it could lead to a heavier penalization of large reconstruction errors.

4.3 Results

In this section, we will dive deep into a comparative analysis of the different Autoencoder architectures and loss functions that were previously proposed to determine which yield the best results for anomaly detection when integrating them with CWoLa.

When setting up the Sparse Autoencoder against the Deep Autoencoder (see Figure 12), the ROC curve indicates a small superiority from the Deep AE over the Sparse AE at detecting true signal events without throwing out a lot of false positives.

Several factors could contribute to these observed results. One possibility could be the lack of optimization spent on the Sparse Autoencoder's hyperparameters. Another factor could potentially be the over-constraint imposed on each layer of the Sparse Autoencoder, as we chose to gradually increase the activation regularization for every following hidden layer, which may have caused the loss of important key features due to the deactivation of certain neurons during the training phase.

Upon establishing the slight but superior efficacy of the Deep Autoencoder, let's shift our attention to the impact that different loss functions could have on the model. It was hypothesized that using MSE as the chosen loss function would most likely yield better results than MAE due to MSE's greater sensitivity to larger reconstruction errors This hypothesis was confirmed, as evidence shown in Figure 11. We can appreciate a significant performance difference between the two, probably because the MSE gives more weight in the metric to larger errors.

Finally, we can appreciate the reconstruction error loss for the validation set on the Figure 9. Where the combined CWoLa-Autoencoder technique managed to achieve a distinguishable separation between the reconstructed Background events when comparing them with the Signal events (the worse reconstruction error for S events lets us know that the B trained model was able to recognize key features in B events that may not be present in S events). The encoder and decoder architectures used for the showcased model can be found in the images 14 and 15 respectively.

Incorporating the CWoLa algorithm with the Deep Autoencoder represents a significant advancement, although it has not yet achieved efficiency levels comparable to that of the fully supervised models this integration seems to point us in the right direction, as it outperforms both the standalone autoencoder and standalone CWoLa as depicted in Figure 13 and Figure 10.

This integration marks a right step in the pursuit of improving anomaly detection efficiency in HEP.

5 Conclusions

We were able to propose an improved classification model, instead of using a fully connected 4-layer NN, we developed a Deep Autoencoder which demonstrated superior performance by reducing the AUC (Figure 10). Even so we consider feasible to further increase the reconstruction loss for anomalous events and increase the AUC of the ROC plotting by further improving the autoencoder's architecture and/or hyperparameters. It's crucial to note that we made no modifications to the data inside the latent layer. And so we open the door towards taking a more complex approach. Future iterations of this work could explore the manipulation of the information within the latent space, architectures such as a Variational Auto Encoder may yield great results. Another promising approach would be to further improve the discrimination power of the autoencoder by opting to implement an Adversarial Neural Network where the classifier goes through iterative training, where the classifier outputs $logP_T$ which is the log of the invariant transverse momentum of every jet, which along the other jet mass variables are passed to a so called Adversary, which aims to predict the invariant mass of the jet, when the mass of the jet is predicted incorrectly by the adversary we know that the classifier has decorrelated itself from the mass of the jet. [15]

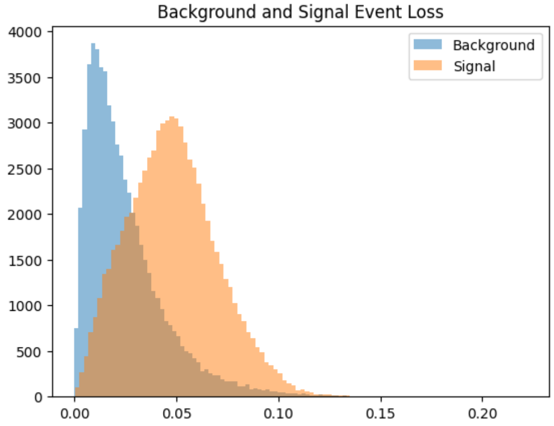

Figure 9: Background and Signal Events Reconstruction Loss using CWoLa + Deep Autoencoder with MSE reconstruction loss.

6 Code

Github:

https://github.com/IvanDLar/CWoLaAutoencoder

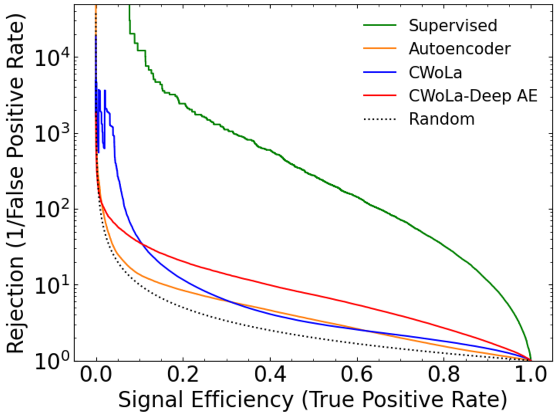

Figure 10: Comparison using a ROC curve between standalone CWoLa and Autoencoder against CWoLa + Autoencoder.

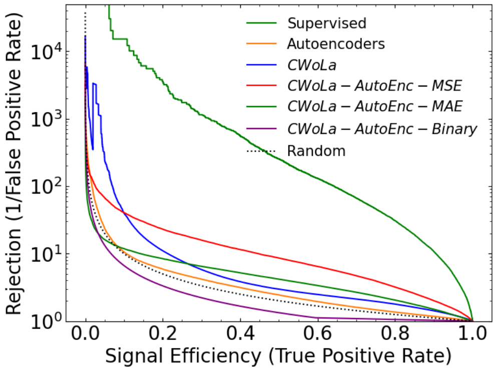

Figure 11: ROC Curve comparing the performance of the CWoLa Deep autoencoder with MSE and MAE loss function. Autoencoders, CWoLa and supervised models used as performance references.

7 Acknowledgments

We would like to thank Michelle Aylin Calzada for providing us with the fully functional code done during her research, this helped us speed up the development process of this paper's proposal. Ivan Diaz would like to thank Rafael Espinosa for his guidance and helpful eye during the entire development of this research paper.

References

[1] P. Calafiura, D. Rousseau, and K. Terao. *Artificial intelligence for high energy physics*. World Scientific Publishing Co., 2022, pp. 86–91.

[2] H. Robert, P. Fisher, B. Heckel, S. Kachru, D. Kaplan, R. Kirshner, D. Kleppner, L. Hay, D. Pine, W. Reinhardt, and N. Roe. *Physics for the 21st Century*. 2010, pp. 86–91.

[3] Katherine Garrett and Gintaras Duda. "Dark Matter: A Primer". In: *Advances in Astronomy* 2011 (Jan. 2011), p. 7. DOI: 10.1155/2011/968283.

[4] L. Dinh, J. Sohl-Dickstein, and S. Bengio. *Density estimation using Real NVP*. 2017. arXiv: 1605.08803 [cs.LG].

[5] Michelle Aylin Calzada and Rafael Espinosa. *A Deep Analysis of the CWoLa Algorithm for Anomaly Detection in High Energy Physics*. 2022.

[6] Eric M. Metodiev, Benjamin Nachman, and Jesse Thaler. "Classification without labels: learning from mixed samples in high energy physics". In: *Journal of High Energy Physics* 2017.10 (Oct. 2017). ISSN: 1029-8479. DOI: 10.1007/jhep10(2017)174. URL: http://dx.doi.org/10.1007/JHEP10(2017)174.

[7] Jack Collins, Kiel Howe, and Benjamin Nachman. "Anomaly Detection for Resonant New Physics with Machine Learning". In: *Phys. Rev. Lett.* 121 (24 Dec. 2018), p. 241803. DOI: 10.1103/PhysRevLett.121.241803. URL: https://link.aps.org/doi/10.1103/PhysRevLett.121.241803.

[8] Jack H. Collins, Kiel Howe, and Benjamin Nachman. "Extending the search for new resonances with machine learning". In: *Physical Review D* 99.1 (Jan. 2019). ISSN: 2470-0029. DOI:

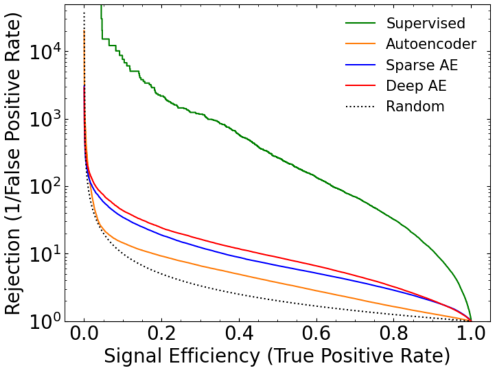

Figure 12: ROC curves for performance comparison between the different autoencoders.

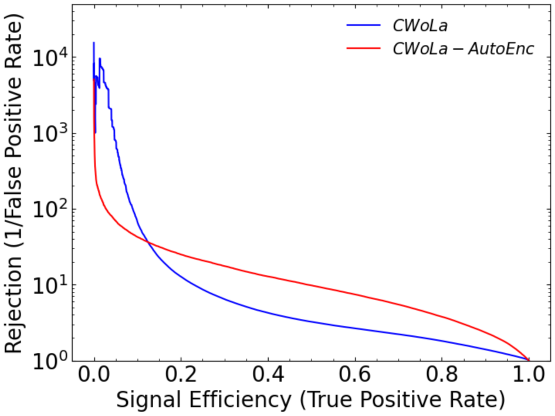

Figure 13: ROC curve comparison of CWoLa Fully Connected vs Cwola Deep AE.

10.1103/physrevd.99.014038. URL: http://dx.doi.org/10.1103/PhysRevD.99.014038.

[9] Fanny Roche, Thomas Hueber, Samuel Limier, and Laurent Girin. "Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models". In: May 2019.

[10] *R&D Dataset for LHC Olympics 2020 Anomaly Detection Challenge*. https://doi.org/10.5281/zenodo.4536377. Apr. 2019.

[11] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. "PYTHIA 6.4 physics and manual". In: *Journal of High Energy Physics* 2006.05 (May 2006), pp. 026–026. ISSN: 1029-8479. DOI: 10.1088/1126-6708/2006/05/026. URL: http://dx.doi.org/10.1088/1126-6708/2006/05/026.

[12] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. "An introduction to PYTHIA 8.2". In: *Computer Physics Communications* 191 (June 2015), pp. 159–177. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2015.01.024. URL: http://dx.doi.org/10.1016/j.cpc.2015.01.024.

[13] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. "DELPHES 3: a modular framework for fast simulation of a generic collider experiment". In: *Journal of High Energy Physics* 2014.2 (Feb. 2014). ISSN: 1029-8479. DOI: 10.1007/jhep02(2014)057. URL: http://dx.doi.org/10.1007/JHEP02(2014)057.

[14] Jesse Thaler and Ken Van Tilburg. "Identifying boosted objects with N-subjettiness". In: *Journal of High Energy Physics* 2011.3 (Mar. 2011). ISSN: 1029-8479. DOI: 10.1007/jhep03(2011)015. URL: http://dx.doi.org/10.1007/JHEP03(2011)015.

[15] Manuel A. Toledo Lugo and Rafael Espinosa. *Adversarial Neural Network on subjetiness substructure observables for mass-decorrelated signal tagging*. 2022.

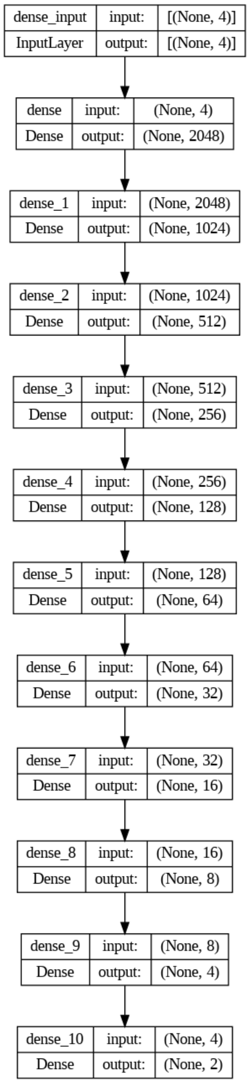
dense_input | input: | [(None, 4)]
InputLayer | output: | [(None, 4)]
dense | input: | (None, 4)
Dense | output: | (None, 2048)
dense_1 | input: | (None, 2048)
Dense | output: | (None, 1024)
dense_2 | input: | (None, 1024)
Dense | output: | (None, 512)
dense_3 | input: | (None, 512)
Dense | output: | (None, 256)
dense_4 | input: | (None, 256)
Dense | output: | (None, 128)
dense_5 | input: | (None, 128)
Dense | output: | (None, 64)
dense_6 | input: | (None, 64)
Dense | output: | (None, 32)
dense_7 | input: | (None, 32)
Dense | output: | (None, 16)
dense_8 | input: | (None, 16)
Dense | output: | (None, 8)
dense_9 | input: | (None, 8)
Dense | output: | (None, 4)
dense_10 | input: | (None, 4)
Dense | output: | (None, 2)

Figure 14: Deep Autoencoder **Encoder** Architecture.

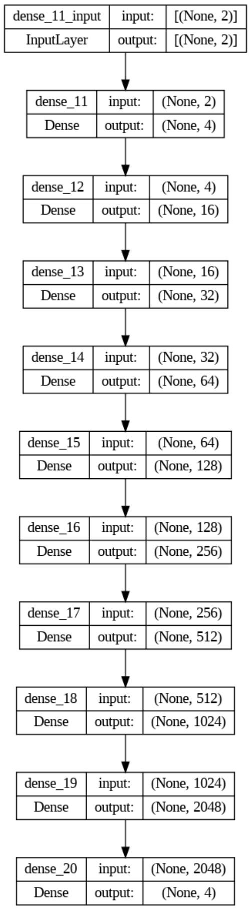
dense_11_input | input: | [(None, 2)]
InputLayer | output: | [(None, 2)]
dense_11 | input: | (None, 2)
Dense | output: | (None, 4)
dense_12 | input: | (None, 4)
Dense | output: | (None, 16)
dense_13 | input: | (None, 16)
Dense | output: | (None, 32)
dense_14 | input: | (None, 32)
Dense | output: | (None, 64)
dense_15 | input: | (None, 64)
Dense | output: | (None, 128)
dense_16 | input: | (None, 128)
Dense | output: | (None, 256)
dense_17 | input: | (None, 256)
Dense | output: | (None, 512)
dense_18 | input: | (None, 512)
Dense | output: | (None, 1024)
dense_19 | input: | (None, 1024)
Dense | output: | (None, 2048)
dense_20 | input: | (None, 2048)
Dense | output: | (None, 4)

Figure 15: Deep Autoencoder **Decoder** Architecture.