Deployment of unsupervised learning in a search for new physics with ATLAS Open Data

Testing of auto encoder for semi unsupervised learning

Sakarias Frette

Spring 2022

Abstract

University of Oslo

CONTENTS

I. INTRODUCTION

The standard model is arguably the best model ever created by man, showing remarkable accuracy comparing with experiments. Its crown jewel was discovered in 2012 by both ATLAS[1] and CMS[2] at CERN. There has however not been any new particle discoveries since the early 70's, and this is somewhat alarming. The standard model has several issues, some of them includes explanation of dark energy, cosmic inflation, the hierarchy problem and no including of gravity.

This leads scientists to look beyond the standard model, and there are several models proposed, such as dark matter candidates, extra vector bosons and super symmetry. Attempts are made by multiple collaborations to discover evidence for these models, and there are several obstacles to overcome, such as method of analysis and sufficient amount of data. With the impressive progress of machine learning software such as Tensorflow[7], machine learning methods have become more and more popular as possible methods to use for new physics.

In the last decade machine learning has excelled from being tedious and hard to program to become available to almost everyone. It's scale-ability and ability to discover hidden structure in large datasets makes it an intriguing candidate to use on data from the Large Hadron Collider. One possible candidate is semi unsupervised learning, which is done by an auto encoder. The idea is to train a model-independent algorithm on only background, with the hope that what ever new physics that is in the data is detected, without knowing what it is. This machine learning model is the method of choice for this report.

This report is structured in a theory, implementation, results and discussion, and conclusion section. In the theory section the necessary theory is introduced, with respect to both data, method of analysis and choice of evaluation. In the implementation section I discuss data handling , with respect to cuts, event selection, scaling and feature choices, choice of hyper parameters for grid search and the software and api's used for training and tuning. In the results and discussion section I display the results and discuss them as as well as distinct observations. In the conclusion I summarize my findings.

II. THEORY

Some of theory sections about anomaly detection and machine learning algorithms are based on previous work done in other courses, such as [4] and [3].

A. Anomaly detection

Anomaly detection is a tool with a wide range of uses, from time series data, fraud detection or anomalous sensor data. Its main purpose is to detect data which does not conform to some predetermined standard for normal behavior. The predetermined standard varies from situation to situation, and can be set by the context it self, and what is expected as an anomaly. Anomalies are typically classified in three categories[1]:

1. Point anomalies

2. Contextual anomalies

3. Collective anomalies

For the purpose of this report we will mostly consider collective anomalies, as anomalies in the standard model has to be collective to claim anything due to noise etc.

In high energy physics we can, using machine learning, separate anomaly detection into two categories, supervised and unsupervised searches.

[1] Paper from ATLAS collaboration can be found here.
[2] Paper from CMS collaboration can be found here.

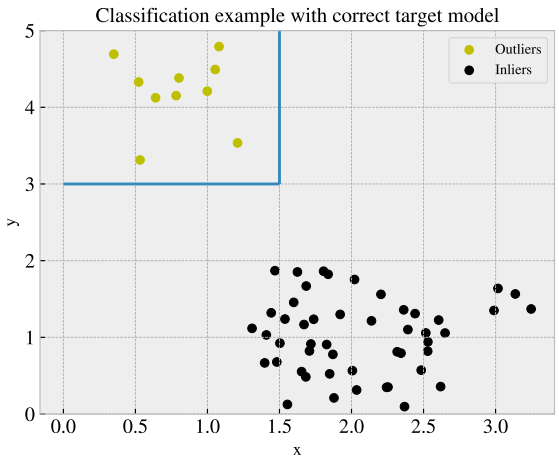

Figure 1: Example of supervised classification of anomaly where the target model is correct. Here the machine learning model manages to correctly identify the anomalies.

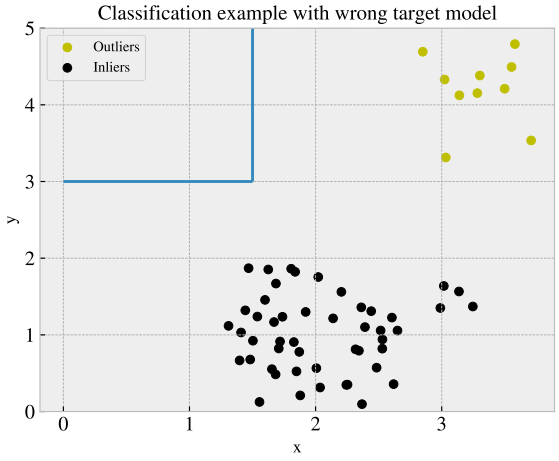

Figure 2: Example of supervised classification of anomaly where the target model is wrong. Here the machine learning model manages to wrongly identify the anomalies.

In figure 1 we see an example of supervised classification. Because the model trained to recognize certain anomalies that are in the data set in manages very well to classify them. In figure 2 we do however see that the same model does not find any anomalies, as they are completely different from the ones it trained on.

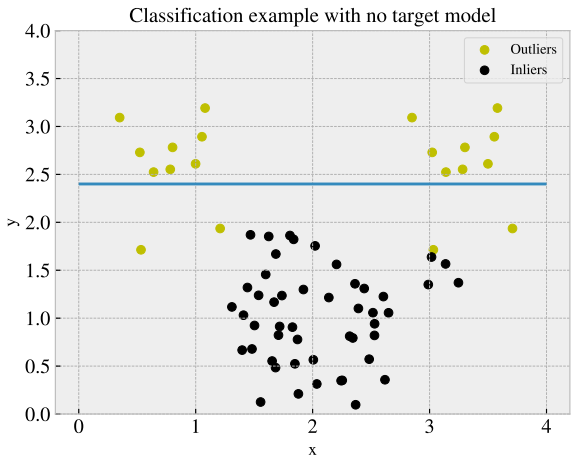

Figure 3: Example of unsupervised classification of anomaly with no target model. Here the machine learning model manages to with good precision identify the anomalies.

In figure 3 we have an example of unsupervised learning. Here we do not tell what is anomalies and what is background. Note that the unsupervised classifier does not classify correctly every anomaly, and the grade of which it does vary from model to model, and from dataset to dataset.

B. Beyond standard model physics

In the context of high energy physics, collective anomalies would be new physics with both do and do not have a model for. Such anomalies could be super symmetric particles[6], new vector bosons or dark matter particles, as well as many other candidates. These types of searches require enormous amounts of data, and both supervised and unsupervised methods are tried continuously to see if there are "hidden" relations in the data that can separate the known standard model from the unknown new physics.

The methods have their strengths and weaknesses, and should be deployed with this in mind. A supervised model will always be better to classify and separate distribution of background and signal events, and if the signal actually exists in the data from the detector, it will give better accuracy than any (semi-) unsupervised methods.

In the event that some signal actually exist in the data from the detector that we do not have a model for, a supervised model will not be able to recognize it purely as a signal, whilst the unsupervised might claim that there is something other than SM background. The certainty of the results will vary with different models, but in the case of an auto encoder, this is due to the fact that it will only recognize background events that is has trained

on, and not any new physics. The drawback with using unsupervised methods in high energy physics, or other fields with multiple anomaly candidates, is that you cannot know what the signal is, only that is it not background.

C. Stacked auto encoder

One method to attack the anomaly detection problem is the so called (stacked) auto encoder. The idea is based on reconstruction, and has been implemented for denoising of images, image compression, and anomaly detection. An auto encoder is a subgroup of feed forward neural networks, and the goal is to compress the information into fewer variables, called the latent space, which then can explain much of the data through decoding that information. This allows the algorithm to learn the most important components of the data. An illustrative image of an auto encoder is shown below.

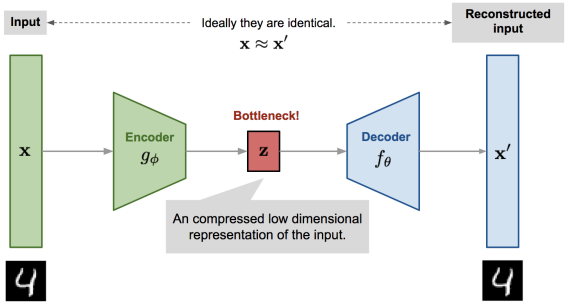

Figure 4: The architecture of an autoencoder with image reconstruction as the example use. Source, accessed 28.04.22

We see here in figure 4 that the input data is deconstructed through an encoder to a lower dimensional space called the latent space, depicted here as $\mathbf{z}$, and then reconstructs the data with the decoder. It is important to note that the number of layers, nodes per layer and activation function per layer for the encoder does not need to match the structure of the decoder. The only requirement is that the input and output layer has the same shape. The end result is the reconstructed data $\mathbf{x'}$. The training of the model is parameterized in the following way. We define the decoder as

$$\mathbf{z} = g_\xi(\mathbf{x}),$$

and the reconstructed information as

$$\mathbf{x'} = f_\theta(g_\xi(\mathbf{x})),$$

where the parameters (ξ, θ) are tuned to reconstruct the data as close to the original data as possible. The model then adjusts following the simple mean squared error of the reconstruction and the actual data, given below

$$L_{AE}(\xi, \phi) = \frac{1}{N}\sum_{i=0}^{N-1}\left(\mathbf{x}^i - f_\theta(g_\xi(\mathbf{x}^i))\right)^2$$

III. IMPLEMENTATION

A. Handling of data

In appendix **??** the features used in the data set are listed, with their respective type and description. The data can be separated into two categories, Monte Carlo simulated (MC) data and actual data from a detector. The MC data contain background samples which are standard model processes, and signal samples which are proposed new physics. The actual data is data from the ATLAS detector. The totality of the data used in this report is given as ATLAS Open Data[3]. This is a dataset that has been analyzed for new physics, and can thus be released to the public.

1. *Cuts, pre-selection and scaling*

To preserve as much information and add as little bias as possible I did very few cuts. The entire data gathering process is biased from the moment the data is collected at the collider, but even then it is good to try to minimize the bias.

The first cut I impose is to require "good leptons". This is done by requiring lep_etcone20/lep_pt < 0.15 and lep_ptcone30/lep_pt < 0.15. We then require only two leptons per event. The event selection was done using RDataframe, to speed up the selection[4]

I tested two methods of scaling in this report, Min Max Scaling[5] and Standard scaling[6][9].

2. *Choice of features*

The choice of features is a mixture of specification and broadness. In this context specification is defined as information that specify for a given part of the physical system, in this case the leptons. Thus, most of the features are directly related to the leptons. Broadness is in this context defined as information about the rest part of the final state, such as jets, photons or tau leptons. Ideally one would pick as much information as possible for

[3] Link to ATLAS Open Data here.
[4] This code was written by Eirik Gramstad, and can be found here.
[5] More on Min max scaling here.
[6] More on Standard scaling here.

each event, but there is a trade-off between amount of information and size of data, and as consequence, execution time.

The features were picked such that there are no missing values, only 0 or larger than 0. For regular machine learning problems, missing values usually gets replaced by the mean value of the feature. This is however a problem when analysing physics, as such a choice could violate the laws of physics. To avoid this, all features are designed to either have 0 if missing, or sum up the contributions of both missing and present values, given that they represent the same type of information.

An example of this is if we have two jets in one event and three in another, using the transverse momentum of the jets as features. We cannot in the first event input the mean values of all third jet-transverse momentum in the entire dataset, but we could put 0, or sum all the transverse momentum for the jets in the given event into one feature. Another possibility is to only give the count of jets in the event, and there are even more ways to solve this issue.

B. Tuning and training

For the auto encoder to accurately reconstruct the standard model, the model needs to train on the background MC. However, neural networks are highly susceptible to hyperparameters, which needs tuning. In this project I used Keras-tuner[8] to tune the hyperparameters. The choice of tuner for this report was Hyperband[6]. It is not well understood how the hyperparameters interact with each other to affect the result, thus there are really no guidelines for approaching this problem. A result of this might be that the choice of hyperparameters seem completely random or strange. The hyperparameters used in the network are the learning rate, the alpha parameter for the LeakyReLU[7] activation function, the activation function for each layer, the amount of nodes per layer and regularization for layer's kernel and output AND . The activation functions used in the hyperparameter grid search where

- ReLU
- LeakyReLU
- tanh
- linear

whilst the different learning rates that was tested was $[9 \cdot 10^{-2}, 9.5 \cdot 10^{-2}, 10^{-3}, 1.5 \cdot 10^{-3}]$, and the kernel and output regularization constant values were $[0.5, 0.1, 0.05, 0.01]$. The optimizer used for all the models is the ADAM[8] optimizer.

The auto encoder was written using Tensorflow api[7][2], using a functional structure[9]. In practise, this model could just as well have been written as a Sequential model[10], but at a cost of flexibility and lack of potential non-linear structure in the architecture.

Certain features where removed, as they did not have a good enough overlap between MC and ATLAS data. All features where checked to ensure as good correspondence as possible, and the figures for all the features can be found in the Github repo for this project, under the "figures/implementation" folder. The features that were chosen to exclude where lep1_tracksigd0pvunbiased and lep2_tracksigd0pvunbiased, which are shown below:

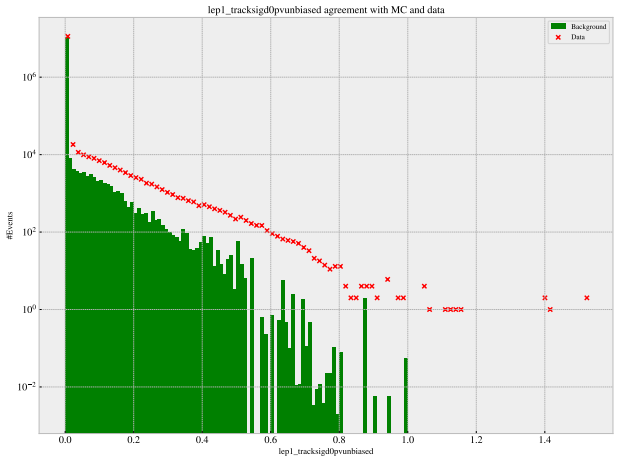

Figure 5: Lep1 tracksigd0unbiased histogram for background MC and ATLAS data.

[7] More on the LeakyReLU api can be found here.

[8] More on ADAM optimizer here.

[9] More on the functional api can be found here.

[10] More on sequential models can be found here.

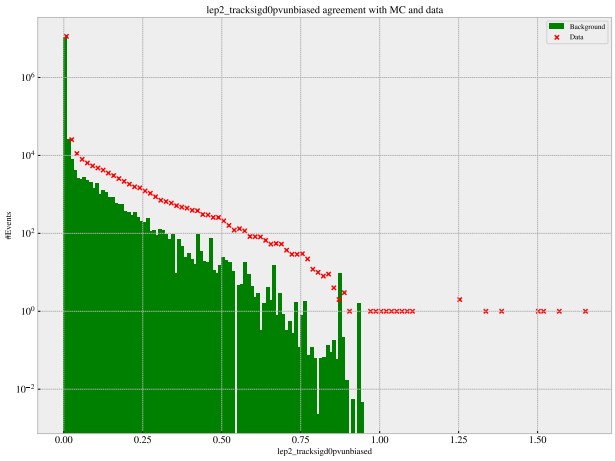

Figure 6: Lep2 tracksigd0unbiased histogram for background MC and ATLAS data.

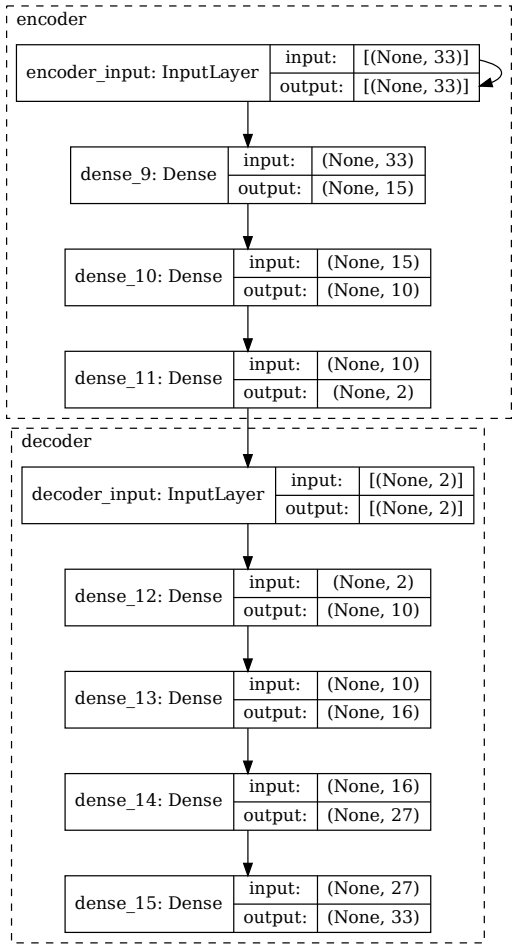

Figure 7: Auto encoder architecture after grid search.

IV. RESULTS AND DISCUSSION

A. The first model

The first auto encoder model was found with a grid search and is structured as shown in figure 7 below.

The grid search gave the optimal hyper parameters as follows: the encoder has the following activation functions: tanh, linear, linear, leakyReLU, and the decoder has the following activation functions: relu, leakyReLU, tanh, and linear. The optimal learning rate was 0.0015, the optimal kernel regularizer was 0.05, the optimal activity regularizer was 0.5 and the optimal alpha for the leakyReLU was 1. These hyper parameters was found after around 30 minutes of searching, and then stopped, as it appeared that the validation MSE did not improve much. Thus, there are most likely a better set of hyper parameters that could get an even lower validation MSE.

The reconstruction error of ATLAS data is shown in figure 8.

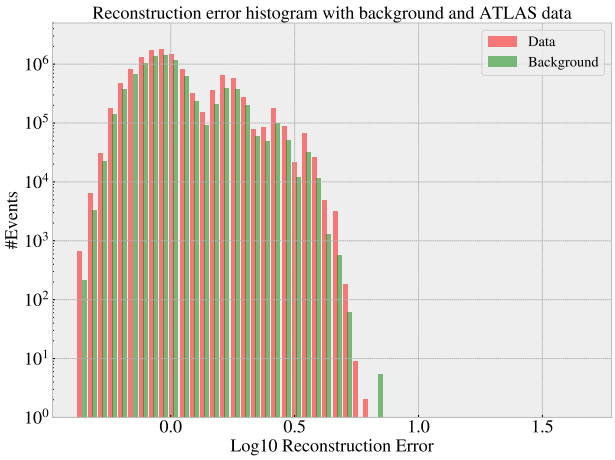

Figure 8: Histogram showing reconstruction error for ATLAS data and background MC for big auto encoder.

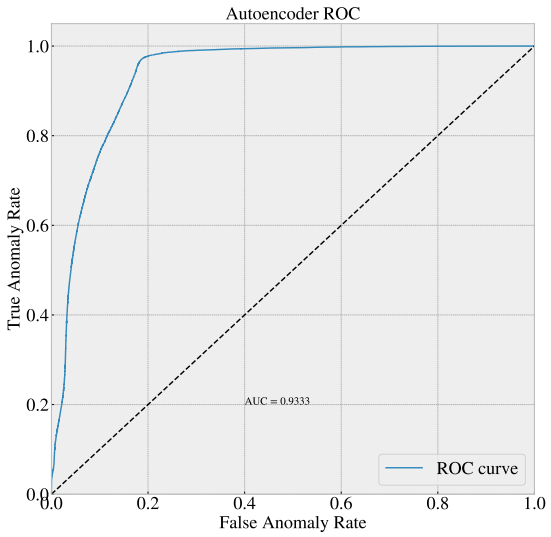

Figure 10: ROC curve signal MC and background MC for big auto encoder with all signal samples.

In figure 8 we see that the auto encoder gets a good overlap of reconstruction error between ATLAS data and background MC. This is in accordance with the fact that the ATLAS collaboration have with very high certainty only found SM data in the ATLAS Open Data. Using all the signal MC samples, I can test if the classifier can distinguish new physics signal MC from SM processes by mere reconstruction. The reconstruction error is shown in figure 9.

From these results in figure 9 and 10 it appears that the auto encoder very well manages to separate signal from background, having only been shown background MC. However, there is some problems with these results. The Randall-Sundrum Graviton to dimuons samples in the signal MC were given with very high weights compared to the Graviton to dielectrons samples, which highly indicates that the data samples are corrupt somehow. Thus we have to check the reconstruction error on signal MC without the Randall-Sundrum Graviton samples. This is shown in figure 12.

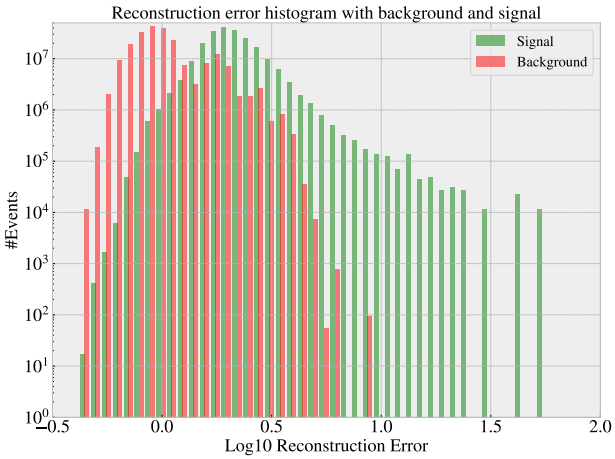

Figure 9: Reconstruction error of background MC and signal MC for big auto encoder with all signal samples.

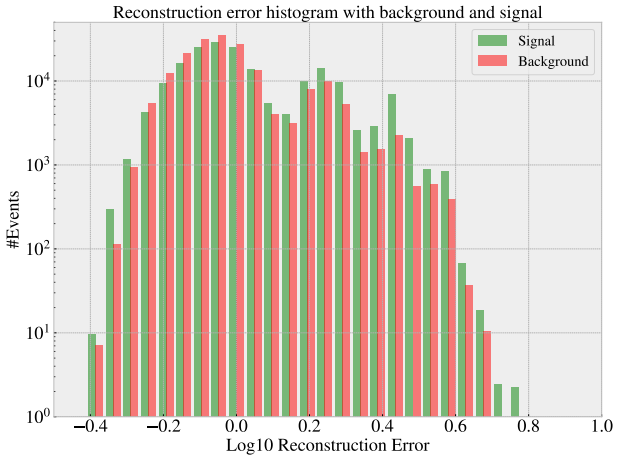

Figure 11: Reconstruction error of background MC and signal MC for big auto encoder without Graviton to dimuons.

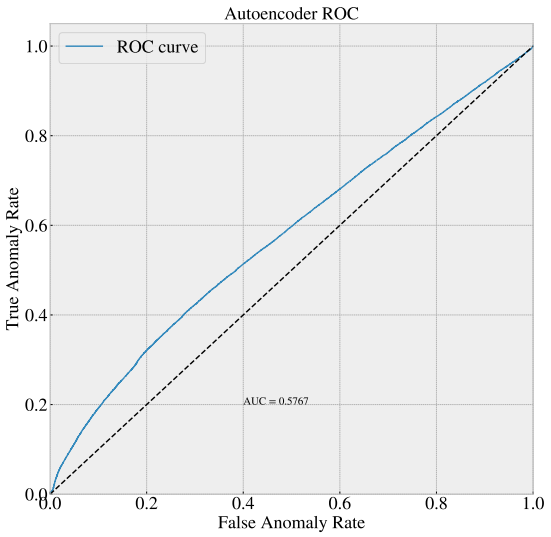

Figure 12: ROC curve signal MC and background MC for big auto encoder without Graviton to dimuons.

In figure 11 and 12 clearly see that the auto encoder struggles to separate background and signal MC, with a AUC score of around 0.58. This shows that for signals that are fairly similar to background MC, this model will struggle to tell them apart. There are possibilities to try to mediate this.

B. Attempts at improvement

One such method is altering the architecture of the auto encoder. To compare this architecture I created a smaller auto encoder, shown in figure 13.

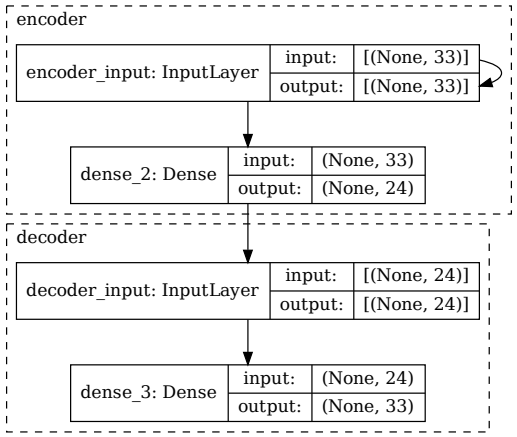

Figure 13: Smaller auto encoder architecture after grid search.

The hyper parameters used for this model was a learning rate of 0.9, 24 nodes in the latent space, the activation function for the latent space was tanh, the activation function for the output was linear, the kernel regularization was 0.05 and the activation regularization was 0.01. The model was trained for 6 epochs.

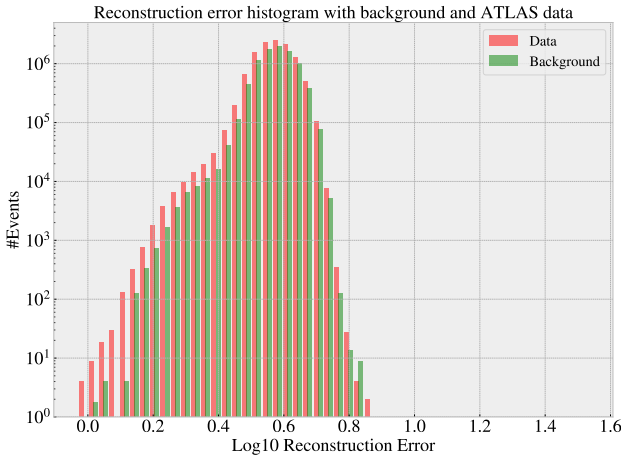

Figure 14: Histogram showing reconstruction error for ATLAS data and background MC for small auto encoder.

Using the architecture in figure 13 I get a good overlap of ATLAS data and background MC, with a center of about 4 MSE reconstruction error.

Figure 15: Reconstruction error of background MC and signal MC for small auto encoder.

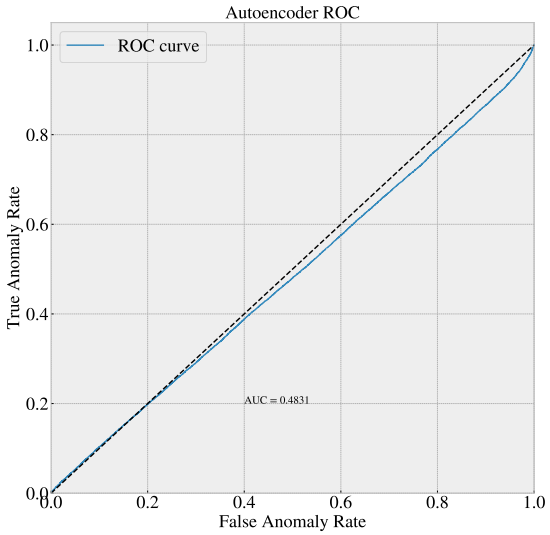

Figure 16: ROC curve signal MC and background MC for small auto encoder

As shown in figure 15 and 16 we observe that even though this model gets a good overlap, it struggles even further than the big auto encoder, having an AUC score of only 0.48 which is only slightly better than guessing. This could mean that such a simple model needs more complexity in terms of layers and node reduction per layer to be able to learn how to separate.

Another idea is removing low level features that are fairly similar for background and signal MC. From testing it was found that even with as low as 13 features, the output from the algorithm only improved slightly.

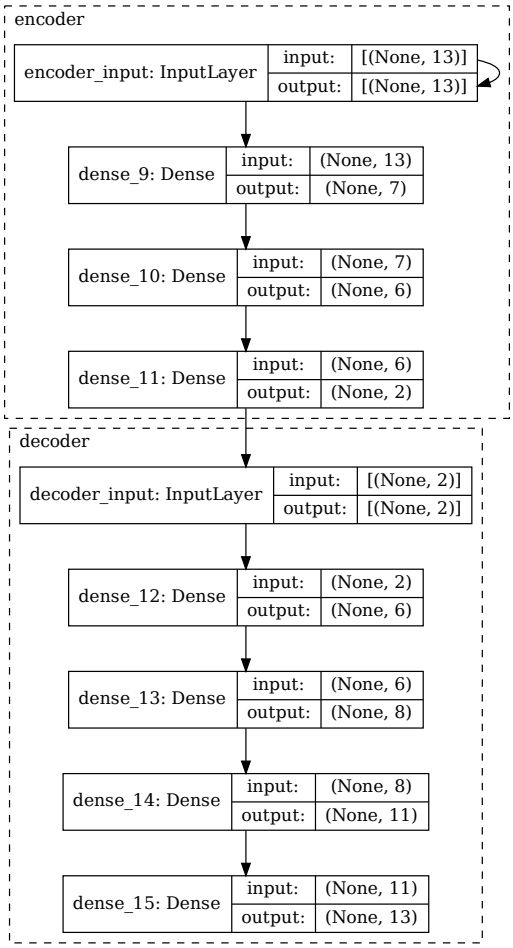

Figure 17: Auto encoder architecture found by hyper parameter search with 13 features in the input.

In figure 17 we have the tuned auto encoder architecture based on a hyper parameter search for 13 feature dataset. The hyper parameters were as follows. The encoder had leakyrelu, linear, leakyrelu as activation functions. The latent space had 2 nodes and linear as activation function. The decoder had leakyrelu, tanh, relu and linear as activation functions. The kernel regularization was 0.5, the activation regularization was 0.5, the leakyrelu alpha parameter was 0.1 and the learning rate was 0.0015. The model was trained with 5 epochs.

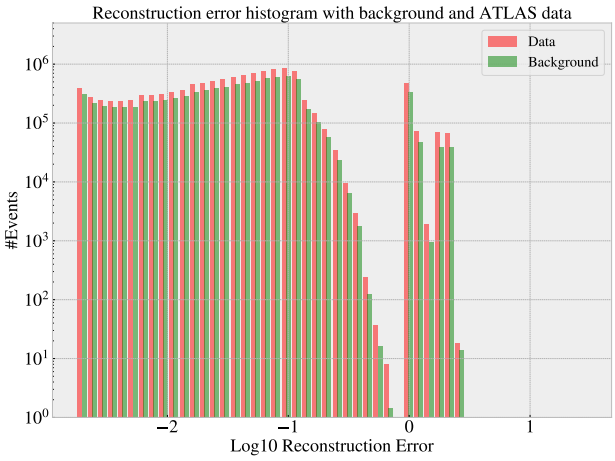

Figure 18: Histogram showing reconstruction error for ATLAS data and background MC for big auto encoder using only 13 features.

In figure 18 we observe that the reconstruction error of ATLAS data and background MC have fairly good overlap. As expected the reconstruction distribution is centered at a much lower value than for the larger data sets, as there are a lot fewer features to reconstruct.

Figure 19: Reconstruction error of background MC and signal MC for big auto encoder using only 13 features.

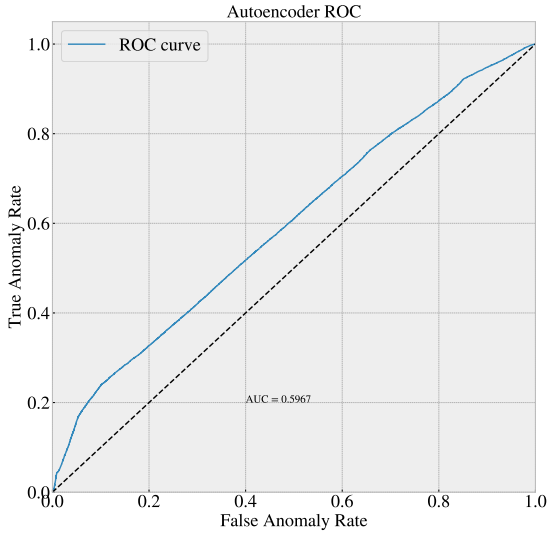

Figure 20: ROC curve signal MC and background MC for big auto encoder using only 13 features.

From figure 19 we observe that even with only 13 features, [njet60, nbjet85, isOS,isSF, mt2, met_et, , lep1_pt,lep1_E, lep1_ptcone30, lep1_etcone20, lep2_pt, lep2_E, costhstar], the auto encoder struggles to separate background and signal MC. There are about 10 events around -0.3 MSE reconstruction error that it cannot match with background MC, but the rest of the data overlaps fairly well. The ROC score for the auto encoder was around 0.59, which only slightly better than for the dataset with all 33 features.

A third option is to change the scaling and look at its effect on the handling of data. The previous models have all used MinMax scaling to scale the data, where as this model used standard scaling instead. Using standard scaling I found the following results.

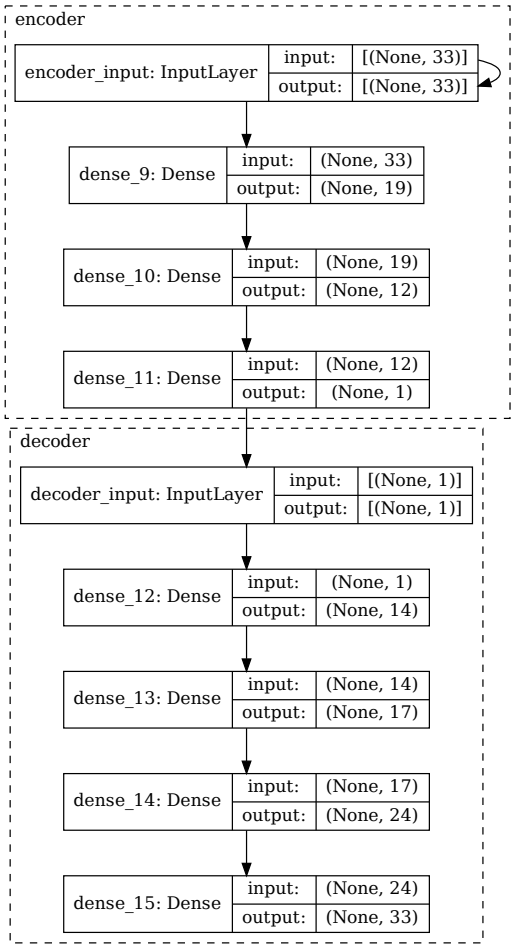

Figure 21: Auto encoder architecture after tuning of hyper parameters using standard scaling on the datasets.

In figure 21 we have the architecture for the auto encoder after hyper parameter tuning where the datasets are scaled using standard scaling. The hyperparameters were as follows. The activation functions for the encoder were tanh on all 4 layers. The activation functions for the decoder were leakyrelu for the first two layers and tanh for the last two layers. The kernel regularization, activation regularization and leaky relu alpha parameter were all 0.01, and the learning rate as 0.0015. The model was trained with 15 epochs.

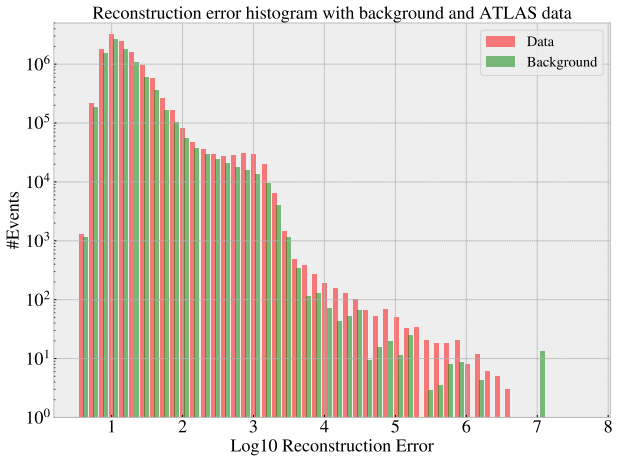

Figure 22: Histogram showing reconstruction error for ATLAS data and background MC for big auto encoder, using standard scaling.

In figure 22 we see that even though the reconstruction error for ATLAS data and background MC has a wide spread, it overlaps well, which indicates that the autoencoder learned to reproduce the standard model, even though some events where much harder to reproduce than others.

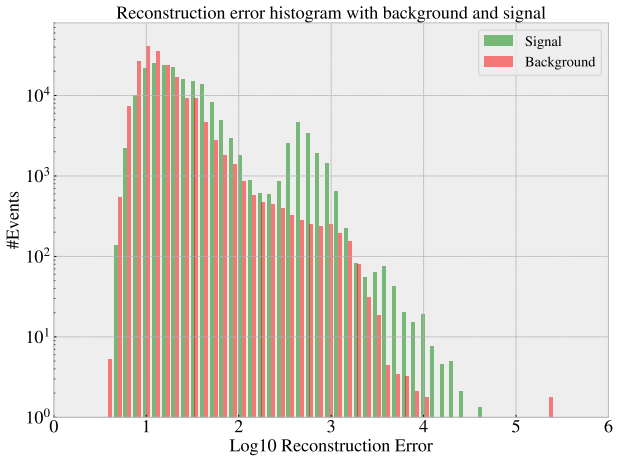

Figure 23: Reconstruction error of background MC and signal MC for big auto encoder, using standard scaling.

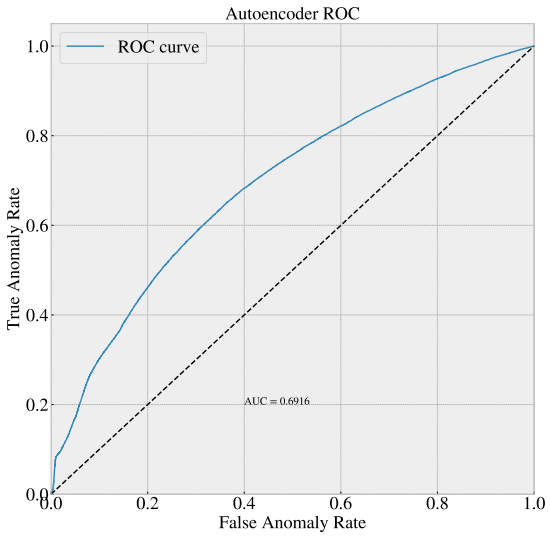

Figure 24: ROC curve signal MC and background MC for big auto encoder, using standard scaling.

In figure 23 we observe that the model somewhat overlaps the background and signal MC, which the exception of a few thousand samples in the second peak. It appears here that the model manages to separate some signal events from the background MC. This model preformed better than the models using Min Max scaling, with a AUC score of about 0.69.

C. Discussion and possible ways forward

There were several issues with training and tuning the model, much of which had to do with hardware. The background and signal MC data set alone was around 17 Gb of data. To do training, prediction and grid searches especially, GPU usage was needed. With Tensorflow's XLA[11] (Accelerated Linear Algebra), the algorithm got a speedup from 1 hour and 30 minutes to around 1 minute per epoch when training on 80% of the background MC. Still, because of the enormous amount of data, the grid search had to be done using a smaller sample size. This was done by extracting 10 million events from the training set, randomly sampled, for training and 2 millions samples from the validation set (20% of the hole background MC) for grid search validation. Thus each epoch only took 10-30 seconds, which allowed for many iterations of hyperparameter tuning. There is however the issue that the random sampling does accurately represent the SM background distribution, and thus the hyperparameters could very well have been even better tuned.

[11] More on XLA here.

It is also worth noting that the ROC curves and reconstruction error figure using the signal MC only in part shows how the auto encoder performs. The samples does not represent all possible new physics, and thus we can only conclude from our results that the auto encoder struggles to separate those signal MC samples. There might very well be possible signals that are much more different than the signal MC samples. This is however only true if the auto encoder actually learned some hidden relations between the features that are not easy to detect.

Another crucial point to make is that in the realm of machine learning, there are few guidelines for how the architecture should look. In this case with the auto encoder, the only requirement is that in the encoder the previous layers must have at least one more node than the one after, and in the decoder the layer after must have at least one more node than the one before it. With respect to the amount of layers, regularization of layers, bias or output, implementation of dropout layers or symmetry with regards to encoder and decoder architecture, an enormous hyper parameter search is the only way to find the ideal model. This bias that the designer of the given auto encoder brings is very hard to quantify, but is indeed crucial, and generates a gigantic pool of possibilities which, given enough time and resources could yield valuable insight.

Scaling also had an interesting impact on the output, as it appears that standard scaling allows for better detection of anomalies. This choice of scaling improved the AUC score from around 0.58 to around 0.69, which is a significant improvement. One reason for why this scaling is better might be that outliers stick out more with this scaling method.

The choice of features also contributes to the effect of the auto encoder. The removal of the d_0 significance for both leptons was done by looking at the distribution of data and background MC by checking the difference between them, and was done by eye sight. This could certainly have been done better, for instance by requiring a threshold for how large the ratio between data and background MC can be.

From the results above it is clear that the auto encoder cannot be ruled out as a possibly good candidate for new physics search, as there are still many aspects that are not well understood. A good start would be to further study how the model learns to merge and reconstruct each feature, and if it quickly converges towards only focusing on one or a few main features. If this is better understood, then maybe the construction of certain features in the preparation of the background MC and ATLAS data could be made to handle this better. It could very well be that certain features or combinations of certain features are two similar for the algorithm to tell them apart. Another point of interest is the relation between number of features and complexity of the architecture. From the models tested above, it would appear that a stacked and more complex auto encoder is needed

for the algorithm to learn to reproduce the dataset. It is also not clear if a symmetric architecture is more ideal than a non symmetric architecture, and a hyper parameter search with this included could yield interesting results.

- Difference between broad and narrow search
- Bias
- bias with features to choose
- difference of how well the model detects different signal models in mc data signals
- any interesting in actual data?
- difference in scaling
- tuning effect
- optimizing tensorflow code with use of autoclustering and possibly mixing floating point precision.
- the use of roc curves, and the validity of what they say
- test if removing mll and mt2 makes a difference

V. CONCLUSION

This report attempted to create an auto encoder that manages to high accuracy reconstruct standard model processes based on training done on background MC and with poor reconstruction on collective anomalies given as signal MC samples. Several models where tuned using Keras-Tuner to get the optimal hyperparameters, and whilst all of the models successfully managed to have good overlap between background MC and ATLAS Open data, only the autoencoder trained on standard scaled datasets manages to separate out signals. We found that the ROC score increased both when using a multi layer auto encoder and by using this standard scaling. Based on the findings in this report, the auto encoder cannot be completely disregarded as a useful method for anomaly detection in high energy physics, as there are still a lot about the algorithm that is not well understood.

REFERENCES

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41 (July 2009). DOI: 10.1145/1541880.1541882.

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

[3] Sakarias Frette and William Hirst. "Seaching for Higgs through supervised and unsupervised machine learning algorithms". In: (2022). URL: https://github.com/WilliamHirst/Advanced-Machine-Learning/blob/main/article/Project_1.pdf.

[4] Sakarias Frette, William Hirst, and Mikkel Metzch Jensen. "A computational analysis of a dense feed forward neural network for regression and classification type problems in comparison to regression methods". In: (2022), p. 5. URL: https://github.com/Gadangadang/Fys-Stk4155/blob/main/Project%202/article/Project_2_current.pdf.

[5] Christopher G. Lester. "The stransverse mass, M T2, in special cases". In: *Journal of High Energy Physics* 2011.5 (May 2011). DOI: 10.1007/jhep05(2011)076. URL: https://doi.org/10.1007%2Fjhep05%282011%29076.

[6] Lisha Li et al. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *Journal of Machine Learning Research* 18.185 (2018), pp. 1–52. URL: http://jmlr.org/papers/v18/16-558.html.

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

[8] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

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

[10] *Review of the 13 TeV ATLAS Open Data release*. Tech. rep. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-OREACH-PUB-2020-001. Geneva: CERN, Jan. 2020. URL: https://cds.cern.ch/record/2707171.

Appendix A: Features

Table I: Features and their description [10]

Feature	Type	Description
njet20	int	Number of jets with $p_T > 20$ GeV
njet60	int	Number of jets with $p_T > 60$ GeV
nbjet60	int	Number of b-jets with $p_T > 60$ GeV
nbjet70	int	Number of b-jets with
nbjet77	int	Number of b-jets with
nbjet85	int	Number of b-jets with
isOS	int	1 if leptons have opposite charge, 0 if leptons have same charge
isSF	int	1 if leptons are of same flavor, 0 is leptons are of different flavor, flavor code 11 is electron, flavor code 13 is muon
mll	float	Invariant mass of the two leptons
mt2	float	The maximal lower bound on the mass of each member of a pair of identical parent particles which, if pairproduced at a hadron collider, could have each undergone a two-body decay into (i) a visible particle (or collection of particles) and (ii) an invisible object of hypothesised mass χ[5].
met_et	float	Transverse energy of the missing momentum vector
met_phi	float	Azimuthal angle of the missing momentum vector
lep_flav	vector<int>	Flavor of the lepton, 11 for electron and 13 for muon
lep_pt	vector<float>	Vector containing transverse momentum for the leptons
lep_eta	vector<float>	Vector containing pseudo-rapidity , η, for the leptons
lep_phi	vector<float>	Vector containing azimuthal angle, ϕ, for the leptons
lep_E	vector<float>	Vector containing the energy for the leptons
lep_ptcone30	vector<float>	Vector containing scalar sum of track p_T in a cone of $R = 0.3$ around lepton, used for tracking isolation
lep_etcone20	vector<float>	Vector containing scalar sum of track E_T in a cone of $R = 0.2$ around lepton, used for calorimeter isolation
lep_trackd0pvunbiased	vector<float>	d_0 of track associated to lepton at point of closest approach (p.c.a.)
lep_tracksigd0pvunbiased	vector<float>	d_0 significance of the track associated to lepton at the p.c.a.
lep_isTightID	vector<bool>	Vector containing boolean indicating whether leptons satisfies tight ID reconstruction criteria
lep_z0	vector<float>	Vector containing z-coordinate of the track associated for the leptons wrt. primary vertex
channelNumber	int	Data sample ID
costhstar	float	Cosine of dilepton decay angle
weight	float	MC sample weight
category	string	SM or BSM category
physdescr	string	MC process name
isSignal	int	1 if category is a BSM signal, 0 if the category is SM background