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[6], 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 we introduce the necessary theory, with respect to both data, method of analysis and choice of evaluation. In the implementation section we 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 we display the results and discuss them as as well as distinct observations. In the conclusion we summarize our findings.

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

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. We typically classify anomalies 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.

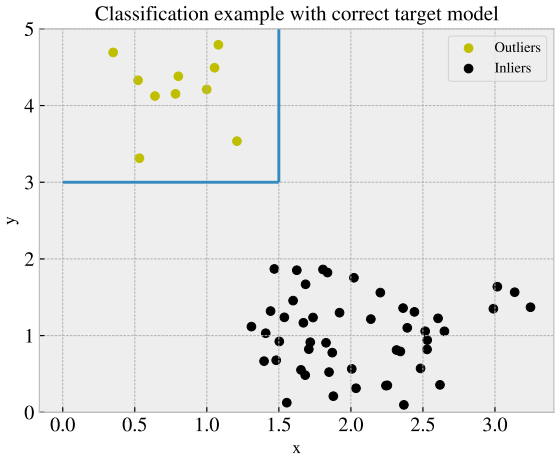

Figure 1: Example of supervised classification of anomaly where the target model is correct. Here the machine learning model manages to correctly identify the anomalies.

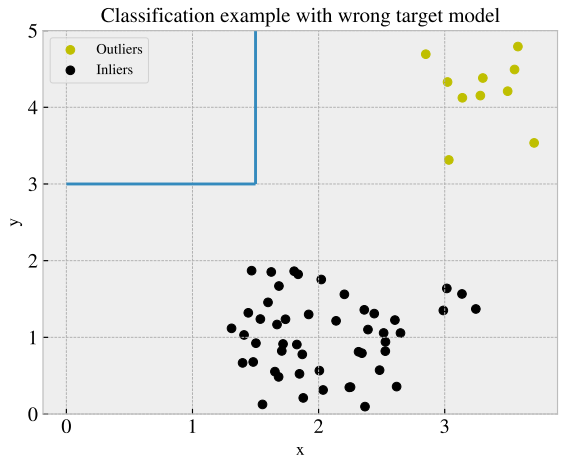

Figure 2: Example of supervised classification of anomaly where the target model is wrong. Here the machine learning model manages to wrongly identify the anomalies.

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.

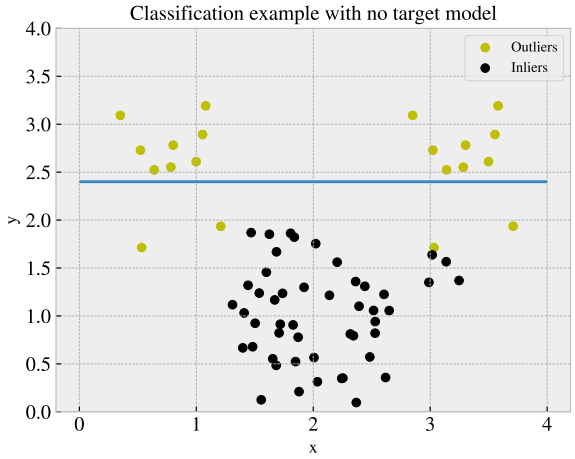

Figure 3: Example of unsupervised classification of anomaly with no target model. Here the machine learning model manages to with good precision identify the anomalies.

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, both supervised and unsupervised methods are tried continuously in the search for 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 very 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.

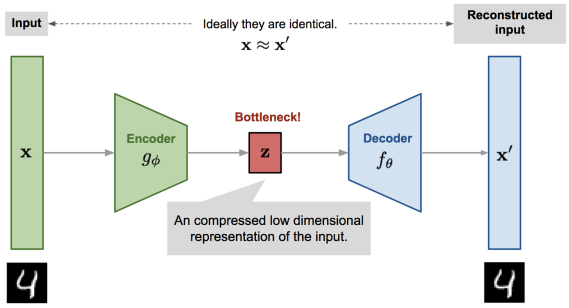

Figure 4: The architecture of an autoencoder with image reconstruction as the example use. Source, accessed 28.04.22

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 **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 seperated into two categories, Monte Carlo simulated (MC) data and actual data. 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 OpenData[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 we did as few cuts as possible. 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 we 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]

We tested two methods of scaling in this report, Min Max Scaling[5] and Standard scaling[6][8].

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 leptons. Thus, most of the features are directly related to the leptons, as they are the main focus of the BSM final states. 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 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 distinguish background from signal, the model needs to train on the background data. However, neural networks are highly susceptible to hyperparameters, which needs tuning. In this project we used Keras-tuner[7] to tune the hyperparameters. The hyperparameters used in the network are the learning rate, the alpha parameter for the LeakyReLU[7] activation function, the activation function

[3] Link to ATLAS OpenData 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.
[7] More on the LeakyReLU api can be found here.

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 auto encoder was written using Tensorflow api[6][2], using a functional structure[8]. In practise, this model could just as well have been written as a Sequential model[9], but at a cost of flexibility and lack of potential non-linear structure in the architecture.

IV. RESULTS AND DISCUSSION

A. Architecture

The first auto encoder model was found with a grid search and is structured as shown in figure 5 below.

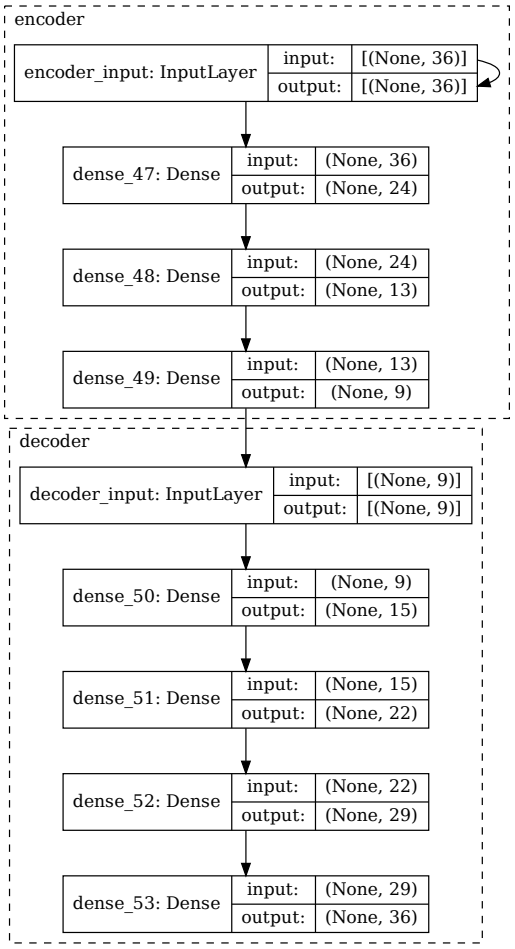

Figure 5: Auto encoder architecture after grid search.

The grid search gave the optimal hyper parameters as follows: the encoder has the following activation functions BLANK, BLANK AND BLANK, and the decoder has the following activation functions BLANK, BLANK, BLANK and BLANK. The optimal learning rate was BLANK, the optimal kernel regularizer was BLANK, and the optimal activity regularizer was BLANK.

This model appeared to struggle with reconstruction of ATLAS data, which lead to tests with a smaller auto encoder, use of dropout, initialization of weights and removal of certain features. The following figures show the architecture for the other models.

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

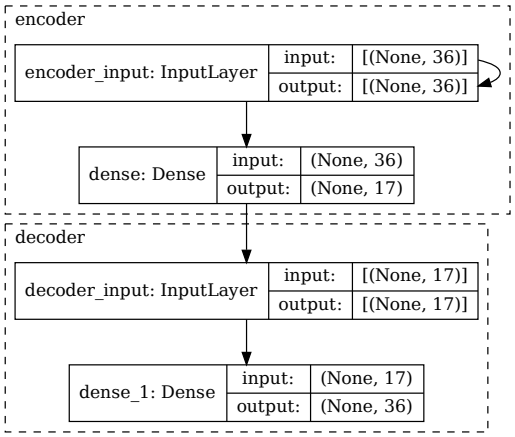

Figure 6: Small auto encoder architecture after grid search.

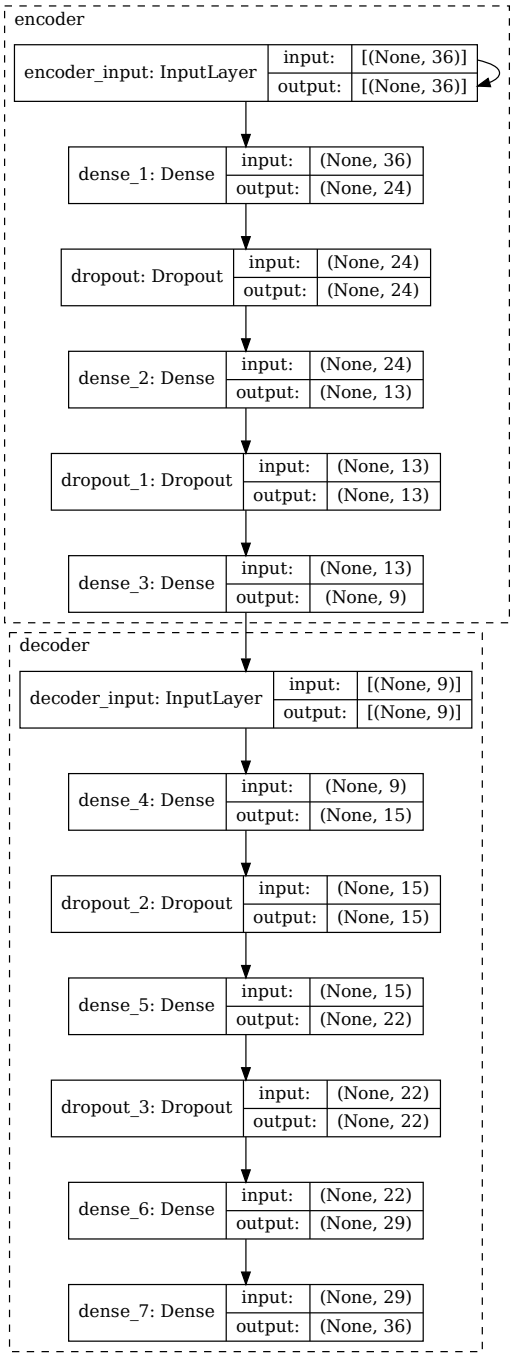

Figure 7: Big auto encoder architecture after grid search, using random uniform distribution as initialization of weights on all layers and dropout between certain layers

B. Semi unsupervised

Using the hyperparameters from the gridsearch, a prediction on background MC, signal MC and ATLAS

data was made after training on 80% of the background MC. REMEMBER THAT BECAUSE OF LARGE DATA AND ONLY CPU THE ALGORITHM ONLY TRAINED 1 EPOCH PER THE FIGURES BELOW!!!
TRAIN THE NETWORKS AGAIN HAVING REMOVED THE CHANNELNUMBER, ITS AFFECTS THE MODELS SIGNIFICANTLY!!!

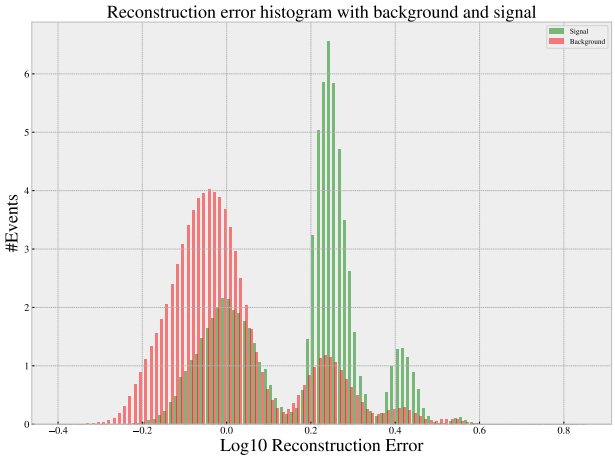

Figure 8: Log Log plot of reconstruction error of background and signal MC for big auto encoder

In figure 8 we have the reconstruction error of background and signal MC in a log log plot. The error shows the same trend for both datasets, and thus has not been able to clearly separate them as something different. This is indicative of a poor classifier.

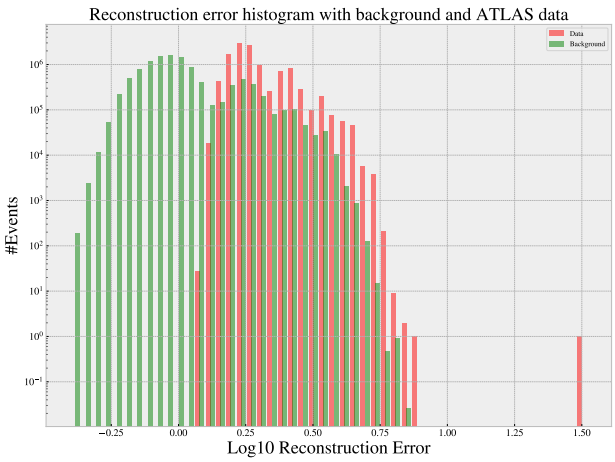

Figure 9: Log Log plot of reconstruction error of background MC and ATLAS Data for big auto encoder

In figure 9 we have the reconstruction error for background MC and ATLAS data in a log log plot. The error displays two separate but close distributions. We also use normalized datapoints in the plot, thus the actual distributions have alot of background and some ATLAS data in the left distributions while the opposite in the right distributions. This is not ideal, as the classifier, with figure 8 in mind, has learned not learned anything useful other than to somewhat copy the input.

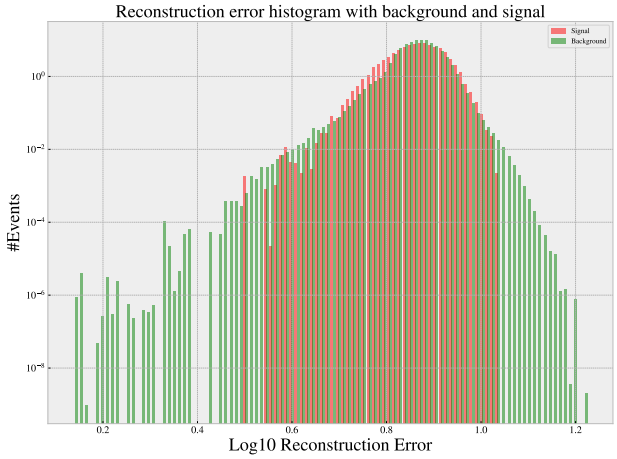

Figure 10: Log Log plot of reconstruction error of background and signal MC for small auto encoder

In figure 10 we have the reconstruction error for background and signal MC in a log log plot. Here we observe the same same trends as as for the larger auto encoder, indicative of poor performance of the auto encoder.

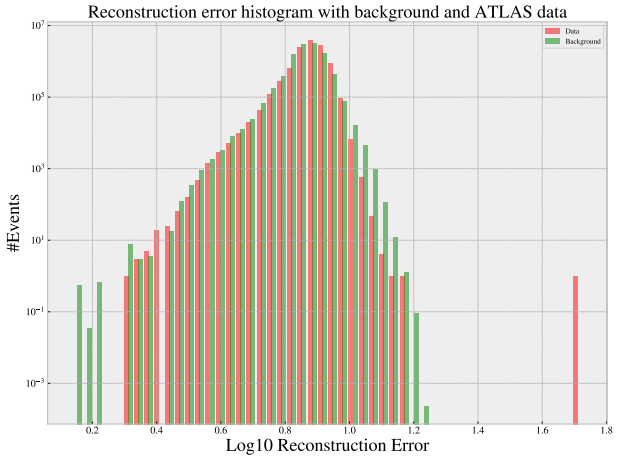

Figure 11: Log Log plot of reconstruction error of background MC and ATLAS Data for small auto encoder

In figure 11 we see that the classifier reconstructs both the ATLAS data and the background MC with the same trend, which means one of two things. Either the classifier just learning the identity function and copies what is put in and gives it out after, or it has manages to learn to reconstruct the standard model.

These results shows that the models themselves

- 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

V. CONCLUSION

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

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

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

[9] *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 [9]

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