University of Venda

Creating Future Leaders

Implementing a robust anti-QCD tagger with mass de-correlated jet image data

by

Rapetsoa Kokotla

Student no: 11640816

Research report submitted in partial fulfilment of the requirements for Master of Science Degree in Physics (e-Science)

School of Mathematical and Natural Sciences

University of Venda

Thohoyandou, Limpopo

South Africa

Supervisor/Promoter: Prof. Deepak Kar

Co-Supervisor/Co-Promoter: Dr Eric Maluta

Abstract

This project studies a robust anti-QCD tagger with mass de-correlating jet image data produced using the pre-processing method introduced in arXiv: 1903.02032. A semi-supervised (where data is only trained on background) learning anomaly detection approach using convolutional autoencoder neural networks is explored as an anti-QCD tagger in this study. Jet image data is used to train the algorithm instead of conventional high level multivariate observables. The pre-processing steps first perform momentum re-scaling followed by a Lorentz boost transformation to find a frame of reference where any given jet is characterised by the same mass and energy, and remove the residual rotation by applying the Gram-Schmidt method on the transverse plane to the jet axis. This is expected to increase the sensitivity of the autoencoder to non-hypothesised resonance and particles as it will not experience non-linear correlation of the jet-mass with other jet observables. A negative result shows that contrary to the convolutional autoencoder outperforming autoencoder in every case where image data is used, it failed to do so in this project. The pre-processing method results in jet images data that the convolutional layer cannot extract information or features from.

Declaration

I, **Kokotla Rapetsoa [student number: 11640816]** declare that this research project is my original work and has not been submitted for any degree at any other university or institution. The project does not contain other persons' writing unless specifically acknowledged and referenced accordingly.

Signed (Student):[illegible].. Date: ....03/09/2020.........

Acknowledgement

I would like to express my greatest gratitude to the DST-CSIR National e-Science Postgraduate Teaching and Training Platform for funding and supporting this Masters of Science in e-Science qualification. I would also like to offer my special appreciation and thanks to my supervisor Prof. Deepak Kar for his guidance and constant support during the research project. I would also like to acknowledge the University of the Witwatersrand CSAM department for providing me access to their GPU cluster which was necessary to train the algorithms. I would also like to thank iThemba LABS for the financial support which enabled me to visit CERN and attend workshops.

Contents

List of Figures

List of Abbreviations

AE	Autoencoder
CAE	Convolutional Autoencoder
LHC	Large Hadron Collider
MC	Monte Carlo
PCA	Principal Component analysis
QCD	Quantum Chromodynamics
ROC	Receiver Operating Characteristic

Chapter 1

Introduction

It is crucial to develop methods that would help physicists find new physics. These findings may be useful in solving some of the mysteries the Standard Model does not account for. For example, the standard model does not include gravity, which is a force evident in our daily lives, and why is there more matter than antimatter in the universe. Another puzzling phenomenon is dark energy, which explains the expansion of the universe and movement of galaxies and dark matter. These two are believed to make up 68% and 27% of the known universe respectively. However, their constituent particles have not yet been detected. A number of methods, both physics and machine learning-based, have been explored, studied and extensively tested to try and find hypothetical new physics beyond the standard model, but they have not yet found any new particles.

Particle accelerators like the 27 km circumference Large Hadron Collider (LHC) are used to perform proton-proton collision experiments at center of mass ($\sqrt{s}$) of 13 TeV to search for new physics. As mentioned in ref. [3], protons are extracted from hydrogen atoms and accelerated to near the speed of light in opposite directions. The protons travel in beams made of around 2808 bunches containing 1×10^{11} protons. These beams are directed by super conducting magnets to a collision point at the centre of all of the four detectors surrounding the collision points. Protons collide but interaction occurs among their constituent particles partons (collective term for quark and gluons that make the protons). Depending on the type of collision and energy transferred, new particles are formed following $E = mc^2$. As quarks and gluons cannot be observed in nature, they decay hadronically to form colour neutral hadrons (collective term for colour neutral particles made up of 2 or 3 quarks held together by gluons, e.g neutrons and protons), but also unstable hadrons decay to form stable ones which we can detect.

Colour neutral hadrons are used to approximate the energy flow of coloured partons which they originate from by the construction of jets. A jet is basically a collection of a collimated bunch of handrons, where each bunch is assumed to have originated from a single particle. Jets can be taken as proxies for partons. Jets are not fundamental objects, but are constructed to capture products originating from some particle.

New physics, if it exists is expected to be found in high mass particles resulting from high p_T (which is the transverse momentum of a particle defined as $p_T = \sqrt{p_x^2 + p_y^2}$ where p_x and p_y are momentum measured in the x and y plane of the detector) collisions. Heavy hadronically decaying particles with high p_T result in a boosted system where all the decay products of those particles collimate together and from this a large radius jet is constructed containing all of its decay product information[1]. Large-radius jets do not only originate from resonance decaying particles but can be a result of background[2] particles like light-quarks, top quark and gluons. This makes discrimination based on subjet combinations difficult, but correlation of jet observables can show the difference in radiation patterns for jets produced through resonance and non-resonant decays.

Physics methods like the cut based tagger studied in ref. [5, 6] are taggers that discriminate signal and background based on a set of event selections on jet momenta that would give the same signal efficiency across a large p_T range. Another example of a physics based jet tagger is the shower reconstruction in ref. [8] which classifies jets according to the compatibility of radiation pattern of a jet with a predefined set of pattern shower hypotheses. HEPTopTagger in ref. [9] relies on reconstructing jets using a large radius jet of $R = 1.5$ to allow the tagging of fully contained boosted top quarks to be effective at lower values of $(p_T > 200\,\text{GeV})$ and to take advantage of the clustering sequence which attempts to reverse the decay structure of the top-quark decay. Machine learning algorithms like deep neural networks and boosted decision trees have also been used. They are mainly trained on high level multivariate observables discussed extensively in ref. [7]. Ref. [1] goes

[1]For example, a hadronically decaying top quark forms a large-radius jets of $\Delta R = 1$ at $p_T > 350\,\text{GeV}$. This can be calculated using the equation $\Delta R = \frac{2m}{p_T}$ where m is the mass (this is the mass of the particle that decays to form a jet.) The invariant mass as it is called is the mass at the inertial or resting frame of the decay system obtained from energy and momenta of the initial particle decay products and p_T is the transverse momentum of the particle.

[2]Signal are events of interest and background are events with somewhat similar properties to events of interest but are not of interest.

on to assess and optimise performance of these algorithms, to determine which jet momenta combination gives the best signal efficiency and background rejection for both boosted decision trees and deep neural networks.

While these taggers work relatively well, we have not found any new particles. This may be due to a number of shortcomings brought by supervised learning methods. The most notable drawback is the reliance on Monte Carlo simulations as data from the detector are not labelled and supervised machine learning performs well when trained on controlled data. Monte Carlo simulations are not yet perfect. Ref. [1] mentions that the primary limiting factor in the description of the tagging efficiency by the Monte Carlo prediction derives from the theoretical modelling of the Monte Carlo processes studied. Machine learning models learn these flaws in Monte Carlo simulated data, thus inevitably making the tagger less effective on real data.

Bottom-up orientated taggers have recently gained traction as viable anti-QCD(Quantum Chromodynamics) taggers since they provide a set of advantages not found in previously mentioned taggers. Anti-QCD taggers look for variations from standard model in data as anomalies, meaning that one does not need to define a signal. This means that they can be trained and implemented directly on data. This project utilises an autoencoder neural network as an anti-QCD tagger. An autoencoder does dimensional reduction on the input data and reconstructs the input as output with minimum error. This way the algorithm learns to reconstruct the predominant data class, in this case light non-top quarks and gluon jets, very well that any event giving a large reconstruction error can be considers as an anomaly. Anti-QCD tagger like bump hunting were being used before autoencoders gained popularity. Bump hunting searches for deviation in the distribution at high m/p_T phase space where QCD-jets have a constantly falling distribution. A peak will be seen near the respective signal mass. However this does not conclude that any anomaly is new physics, because certain tests like checking for detector effects and many others have to be carried out in a concrete study.

Ref. [10] describes the drawbacks that follow with the reliance of correlation when using machine learning based taggers. They mention that machine learning algorithms learn the non-linear correlation of jet mass with other jet momenta as jet mass is a powerful discriminating feature. This can be seen in Figure 1.1 which shows top quark mass peak with large-R jets obtained from dedicated semi-leptonic top selection. Fixed threshold event selection on jet tagging observables distorts the shape of the distribution of non-resonant particle jet, making it similar to the distribution of resonance particle jet. This effect mentioned before along with others make the machine learning algorithm less useful in identifying hadronically decaying particles contained within a large radius jet where the jet mass is not known prior, but has limited effect on the tagging of hadronically decaying top quark, W and Z boson, of which the mass is known.

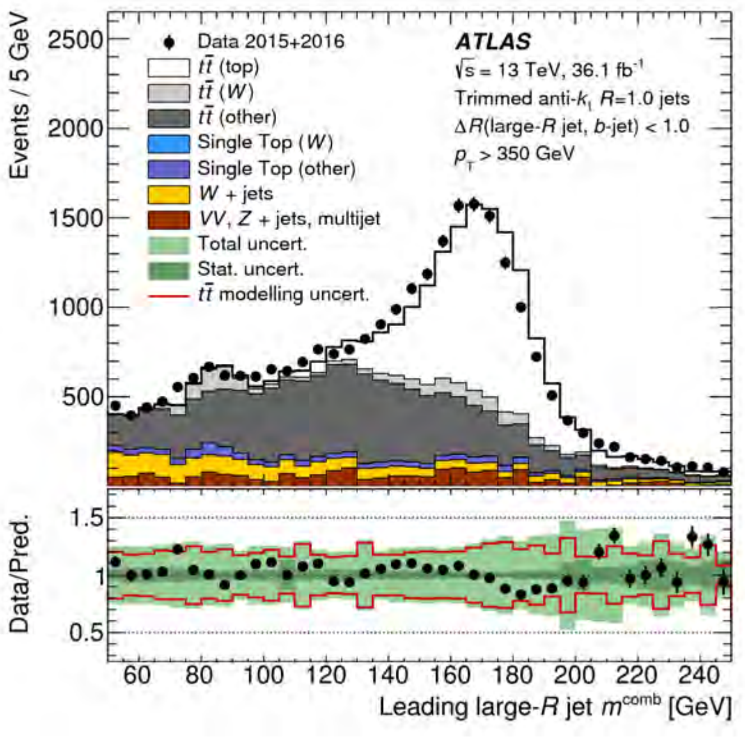

Figure 1.1: Distribution of m^{comb} and semi-leptonic events from ref. [1].

The autoencoder also learns the jet mass correlation in data as in the signal is only expected at a specific mass value. This was shown in ref. [4]. To mitigate this occurrence, we implement a mass de-correlation pre-processing method presented in ref. [11]. Basically the pre-processing method re-scales the jet mass to a pre-determined value, and then performs a Lorentz boost to the jet frame with respect to pre-determined mass and energy value. By removing the dependency on jet mass, we hope that the autoencoder will be more sensitive to other m/p_T phase space where data is scarce.

1.1 Problem statement

Machine learning algorithms require controlled data to be able to give satisfactory results in identifying jets and the particles they originate from. This makes particle physicists reliant on Monte Carlo (MC) simulation as data from the detectors are not labelled. Although MC simulations are precise, they are not yet perfect at resembling the actual particle interaction process. Also, the performance of machine learning algorithms in most cases learn the correlation of jet mass with other observables as it is a powerful discriminant. This limits the sensitivity for detecting new particle of unknown mass. Another issue is that, while new physics is expected to be found at high m/p_T phase space, there is not enough data to train machine learning algorithms in those regions.

1.2 Aim

The aim of this study is to explore an unsupervised learning approach for tagging jet events by implementing a robust anti-QCD tagger trained on jet image data that is de-correlated of jet mass using the pre-processing method introduced in ref. [11] where the anti-QCD tagger is a convolutional autoencoder. For the aim to be achieved and deemed successful, the convolutional autoencoder should have a lower reconstruction error for background QCD jets and a higher reconstruction error for signal top jets with respect to the QCD reconstruction error.

1.3 Objectives

The first objective of this project is to develop an advanced convolutional autoencoder (CAE) that according to previous literature would tag a signal from a background more effectively. This is then followed by testing robustness of the pre-processing method by determining whether indeed QCD events from different m/p_T phase space result in a somewhat similar reconstruction error showing that the reconstruction error is not affected by jet mass.

The third step is to train the model on QCD jet event and test them on top jet events in order to see if it can give a larger reconstruction error for top-jets. This will show us if the anti-QCD is ready to be tested further by comparing its results with those obtained in ref. [11, 4].

Outline

The outline of this report is as follows. Chapter 2 gives literature review on architectures of convolutional autoencoder, their application in anomaly detection in other fields and in high energy physics, as well as some uses of convolutional neural networks in top tagging. Chapter 3 explains how the data is simulated and pre-processed. The description of the CAE's architecture is also extensively discussed in this chapter. Chapter 4 describes the data after being simulated and pre-processed, the architecture of the CAE with features mentioned in chapter 3, the results and discussion of the results. Chapter 5 concludes the project report, mentions the limitations experienced throughout the study and also gives recommendations for future studies.

Chapter 2

Previous work

This chapter reviews previous literature on the best architecture for CAE, application of CAE in anomaly detection and use of autoencoders in searching for new physics. Also, literature of convolutional neural network in top jet tagging is explored.

2.1 Autoencoder

An autoencoder is a neural network trained to copy its input as the output as mentioned in ref. [27]. The aim of the autoencoder is to transform input data into output with the least possible amount of distortion. Autoencoder is a symmetrically structured algorithm that consist of an encoder and decoder part. The encoder learns and describes a code that represent the input data and decoder reconstructs the input from that code. Autoencoders were first introduced by Geoffrey Hinton and the PDP (Parallel Distributed Processing) group in ref. [17] with efforts to address back-propagation (method that allows cost function information to flow backwards through the neural network during training in order to compute the gradient) without explicitly defining an input-output pair, now commonly known as unsupervised learning.

The use of autoencoders gained interest once again when Hinton et. al. (2006) in ref. [34] showed how it can outperform algorithms like Principal Component Analysis (PCA) in dimenstionality reduction. Convolutional neural networks, which contribute a significant part to this project, where introduced in ref. [18] as neural networks that use the linear mathematical operation convolution to force the extraction pf local features in commonly n-dimensional dataset by restricting the receptive fields of hidden units to be local. Ref. [35] is one of the first studies to use convolutional layers in autoencoder for unsupervised learning problem. The model had convolutional and max-pooling layers on the encoder and convolutional and up-sampling layers at the decoder. They learned that the model produces trivial solutions if trained as an identity function, and introduction of sparsity and constraint in the network would better the results.

Masci et. al. in ref. [36] studied CAE with and without max-pooling layer in the encoder part for shallow and deep AE[1]. They realised that the model performs better with the use of max-pooling layer as it introduces sparsity[2] in the neural network, removing the need for regularisers (which are methods used to punish larger or small weights in an algorithm, making it more generalised). They also realised that CAE performs better when the convolutional layers are pre-trained instead of being trained conventionally, where the weights are random values. Zhao et. al in ref. [37] introduced a CAE architecture similar to that used in this project. The CAE architecture consists of several convolutional and max-pooling layers followed by a fully connected neural network on the encoder part. The decoder part consists of fully connected neural network followed by up-pooling de-convolutional layers. They also added a dropout layer to the fully connected neural network section. They realised that the up-pooling layer in the decoder provides better reconstruction image quality than up-sampling.

AE and CAE have been applied in multiple anomaly detection studies before. Ke et. al. in ref. [2] used CAE for anomaly detection of logo images on mobile phones. They reached 97.4% accuracy for detecting anomalous logos. Park in ref. [38] used CAE to detect hypertext transfer protocol (HTTP) intrusions in network communication. Ribeiro et. al. in ref. [39] used CAE to detect anomalous behaviour in automated video surveillance.

[1]The word 'deep' in neural networks which normally represents the structure of the algorithm in the sense of number of layers, does not have a universally agreed upon value or threshold. In most cases a deep neural network is that with more than two hidden layers as mentioned in ref. [19].

[2]Sparsity, in a literal sense means that most weights in the neural network are 0. This results in a more generalised algorithm that is both space and time efficient.

2.2 Autoencoder in searching for new physics

To this point, there have been three main studies that have explored the use of AE and CAE as anomaly detection tools in high energy physics.

Farina et. al. in ref. [4] studied the use of PCA, AE and CAE to distinguish large radius QCD jets from other types of heavier, boosted resonance decaying hadronically, in their case all hadronic top jets and 400 GeV gluinos decaying to 3 jets. They performed two neural network training, one semi-supervised (where training is on background data only) and the other was unsupervised learning (where training is on background that contains contamination of signal data). For semi-supervised learning, the convolutional autoencoder reached a reconstruction error range of $\approx \times 10^{-7}$ to $\approx \times 10^{-4}$ where the reconstruction error was calculated using the mean squared error. The reconstruction error for signal (top jet and gluino) ranged from $\approx \times 10^{-6}$ to $\approx \times 10^{-4}$. The reconstruction error distribution of QCD is skewed to the right while that of signal data is skewed to left. A cut in the reconstruction loss can be taken as a threshold for anomalies. The convolutional autoencoder gives a higher reconstruction error for top jet but performs badly for gluino while the principal component analysis does substantially better. The high performance of the principal component analysis is according to the authors, a result of the jet mass correlation.

Performance for unsupervised learning was relatively similar to that of weakly supervised learning. There is no major visible difference in reconstruction error for both weak supervised and unsupervised learning. The reconstruction error range and distribution for both QCD background and top jet signal is similar to that of the weak supervised learning. Performance was stable for signal contamination of 1% up to 10%, which shows that the model is not affected by the presence of signal events during training. It was also noted that CAE is least affected by jet-mass correlation compared to the other two algorithms.

On the other hand, Heimel et. al. in ref. [12] had a different approach for dealing with jet mass correlation when using autoencoder for anomaly detection. The architecture of their algorithm was similar where the optimiser used and cost functions but PRelU (Parametric Rectified Linear Unit) was used as an activation function instead of RelU (Rectified Linear Unit). They also tested performance of autoencoders with different bottlenecks ranging from six neurons to 34 to see which gives the lowest reconstruction error for training. They found that bottleneck between 20 and 34 gave the best performance, with an area under curve of 0.89 and loss of $\times 10^{-5}$. The convolutional autoencoder was trained on 100000 images, of which 3% is signal contamination and tested on 10 sets of 40000 (half being background and the other half signal) to have statistical independence.

Unlike the paper discussed before, the latter used an adversarial autoencoder to de-correlate jet events of jet mass. The additional adversary was trained to extract, in their case the jet mass from the autoencoder output. An adversarial loss function used in this case can be written as

$$L_{adv}(M) = \left[\tilde{M}\left(|k^{adv}_{T,i} - k^{auto}_{T,i}| - M\right)\right]^2 \tag{2.1}$$

where $\tilde{M}$ is the trained proxy jet mass, M is the jet mass, k^{adv} is the input and k^{auto} is the output. The combined loss function that replaced the mean squared error was in terms of Lagrangian multiplier as

$$L = L_{auto} - \lambda L_{adv}(M) \tag{2.2}$$

where λ is the Lagrangian value. This method, while giving a good performance, is a complex neural network which would require high computational power, especially when implemented on large scale data. It is also not robust as it can not be used on AE based on Lorentz layers.

This project is mostly based on the study by Roy et. al. in ref. [11]. As mentioned earlier, they developed a physics-based pre-processing method to de-correlate the AE reconstruction error from jet-mass. They tested the robustness of this pre-processing method by training and testing on QCD events from different m/p_T bins to see if they result in the same reconstruction error. They also tested the performance of the autoencoder where anomalous events are top-jets and W boson jets. Then another test was done where the anomalous events are new physics particles decaying into di-W boson jets. The reconstruction error for QCD events at different m/p_T bins were fairly similar, showing the robustness of the method. They were able to obtain up to 60% signal selection efficiency for W and top jets. A signal selection efficiency of di-W boson jets achieved a signal selection efficiency of 70%. One should note that these signal selection efficiencies where achieved when the reconstruction error was combined with jet mass m_J and N-subjettiness variables.

Even though Roy et. al. have not used convolutional layers in their autoencoder, there has been some application of convolutional neural networks (CNN) for top tagging which gave promising results. Kasieczka et. al. in ref. [16] developed a convolutional neural network based top tagger called DeepTop trained on jet image data. They realised that it gives comparable results to a QCD-based top tagger made of BDT (boosted decision tree) trained on high-level multivariate inputs. The results gave confidence that CNN will give better performance if better optimised. Macaluso et. al. in ref. [47] explored a number of improvements to DeepTop tagger mentioned above. Some of the improvements explored were increasing the number of filters and using deeper layers, using coloured[3] jet images instead of conventional greyscale jet image and exploring other optimisers among others. These features improved performance of DeepTop by a factor of 3-10 factors for the CMS sample and 1.5-2.5 factors on the DeepTop sample. They also obtained better performance compared to high level multivariate-based taggers currently in use.

[3]Coloured means additional information such as calorimeter p_T, track p_T, etc. to the jet image making it 3 dimensional.

Chapter 3

Methodology

This chapter gives a brief description of how the data were simulated, how it were pre-processed and the convolutional autoencoder. All features of the CAE used in this project are explained in this chapter.

3.1 Simulating data

This section gives a short description of the simulated data, how it was simulated and a description of the pre-processing method used to produce jet-mass de-correlated simulation data. Even though the aim is to reduce the decency simulated data, it is used for this project. The reason being that ATLAS (A Toroidal LHC Apparatus) experimental data requires one to have ATLAS affiliation to use it, also this project is a proof of concept project to test if this method does actually work. The background events, which are gluons and non-top light flavour quarks are simulated by a dijet production. They are generated using Pythia8, ref. [13], which uses LO[1] calculations. The signal events for this study are high-p_T top quark, which can be obtained from the hadronical decay of W' or Z' boson, which are physics phenomenons beyond the standard model. Top-quarks used are of semi-leptonic decay. Top quarks are also simulated using Pythia8. The simulation is pre-detector simulation, meaning only the particle is looked at.

From the simulated results, this jets, which are a construction of captured hadrons originating from a single parton,[2] are formed. The clustering algorithm anti-k_t will be used to reconstruct jets. This algorithm clusters jets by starting with the highest p_T input object, then takes all the softer input within a radius $R = 1$ (in the case

[1]LO(Leading order) corresponds to the tree level of the Feynman diagram.

[2]Parton refers to a class of fundamental particles which mainly consist of quarks and gluons.

of this study) to form a roughly circular shaped jet.

This project only focuses on a parton level study. The authors in ref. [11] of the pre-processing method to be explained below are confident that traditional trimming methods will be able to handle this contamination at a detector level study without deteriorating the performance of the tagger. The next sub-section explains the decorrelation method.

3.1.1 Pre-processing method

Four basic momentum variables and energy of jets are pre-processed to form jet image data to be used to train the anti-QCD tagger. A summary of the pre-processing method is presented below. The process is divided into three stages. For a detailed explanation and understanding, refer to ref. [11].

- The initial step is to re-scale the constituent momentum vector of the jet so that mass of the jet is m_0.
- The second step is performing a Lorentz boost to a frame where the energy of the jet is E_0, the direction of the boost is either parallel to the jet momentum if $E'_J > E_0$ or anti-parallel to the jet momentum if $E'_J < E_0$ where E'_J is the rescaled jet energy. The boost factor is shown in Figure 3.1.
- The third step forms a jet image using dimensionless ratios between the momentum components and energies of jet constituents. Jet images are two dimensional histograms of energy (or p_T) of its constituents with respect to the transverse plane to jet axis. The Gram-Schmidt method is used to determine the optimal set of basis vectors in the transverse plane while also removing the residual rotation symmetry still present for the constituents.

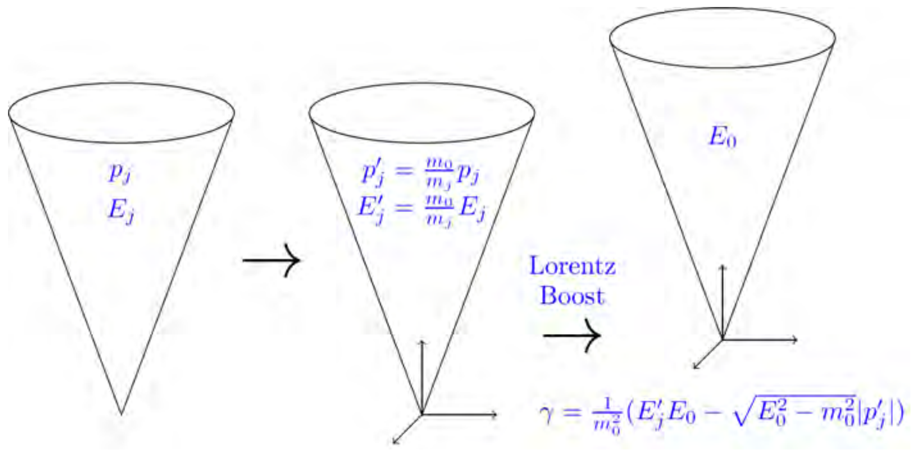
p_j
E_j
$p'_j = \frac{m_0}{m_j} p_j$
$E'_j = \frac{m_0}{m_j} E_j$
Lorentz Boost
E_0
$\gamma = \frac{1}{m_0^2}(E'_j E_0 - \sqrt{E_0^2 - m_0^2}|p'_j|)$

Figure 3.1: Visual representaion of the pre-processing method.

Figure 3.1 shows a visual of how the pre-processing method would be carried out. The jet-images formed are a η vs ϕ plot of jets.

The next section will give a description of the convolutional autoencoder and why it is the preferred algorithm to tackle the task at hand.

3.2 Anomaly detection with convolutional autoencoder

As mentioned before, an autoencoder is used as a mechanism for anomaly detection for physics beyond the standard model in ATLAS. The need for anomaly detection arises when we have one class that is well characterised by instances in the training data, but the other class or classes have very few to no instances or do not form a statistical representation of the class. In the case of this research project, there is a high quantity of jets originating from low m/p_T particles which is usually QCD jets and small high m/p_T particles which is where new physics is expected to be found.

An autoencoder is an unsupervised learning neural network that applies backpropagation to the network, setting the target value (output) to be equal to the input. Backpropagation[3], which is short for "backward propagation of errors" is a method for calculating the gradient of the error function with respect to the neural network's weights. By minimising the cost function, the algorithm is able to reproduce the background data which is plentiful in the training data but fails to reconstruct the

[3]The phrase 'back' is used to reference how this method operates where the last neural network layer's neurons are the first used to calculate the gradient and the first layer is last.

signal or anomaly. A threshold in the reconstruction error can be used to determine whether a particle is categorised as an anomaly or not.

At first glance, the autoencoder may seem like an identity function and a bit trivial to learn but by placing constraints on the hidden layer we are able[4] to learn interesting structures and non-linear correlations about the data. An autoencoder structure consists of two section, an encoder and a decoder. For an input $x \in \mathbb{R}^d$, encoder can be defined as

$$h = f_{W,b}(x) = \sigma\big(\sum_{j=1}^{m} W_{i,j}^{(l)} x_j + b_i^{(l)}\big), \tag{3.1}$$

where $W_{i,j}$ is neuron i from input j, b_i is the bias value in hidden layer l, x_j is the input value j and σ is the activation function. The decoder can then be defined as

$$\hat{x} = g_{W,b}(h) = \sigma\big(\sum_{j=1}^{d} W_{i,j}^{(l)} h_j + b_i^{(l)}\big), \tag{3.2}$$

where $\hat{x}$ is the output of the decoder, $x = \hat{x}$ and $m < d$. This equation is for a fully connected autoencoder, the convolutional part will be explained in the next section. This architecture learns low-dimensions of the data in a manner similar to the Principal Component Analysis (PCA). The following sections will give a detailed explanation of a convolutional autoencoder and why it has the structure it has. A detailed explanation of a convolutional autoencoder will also follow as it is a suitable algorithm for anomaly detection on image data.

3.2.1 Convolutional autoencoder

The architecture proposed for anomaly detection on image data consists of convolutional layers and max pooling layers for the encoder, fully connected neural network in the middle and then de-convolutional layers along with un-pooling or up-sampling for the decoder depart. Convolutional layers have demonstrated superiority in image recognition compared to conventional fully-connected neural networks which makes convolutional layers an obvious tool to achieve the research goal.

Fully-connected neural network structures tend to ignore the topology of input data. A convolution operation of two matrices is a method that produce a third matrix

[4]By doing this, the input data $x_i \in \mathbb{R}^d$ is compressed to $m(m < d)$ number of neurons in the hidden layer as mentioned in ref. [20].

which expresses how the one matrix is affected by the other. Due to the high spatial or temporal correlation of pixels in images, convolutional neural networks are best suited for this as they can extract and combine abstract features to recognise spatial and temporal objects. This way, spatial locality is preserved because convolutional neural network weights are shared among all locations in the input data.

A convolutional layer is composed of two parts, a feature map which is the input image and a kernel of filters. A kernel or filter is a two dimensional matrix, whose weights are multiplied with the feature map to extract or enhance valuable descriptive characteristics from the feature map. A kernel of size $n \times n$ convolve over the feature map of size $m \times m$, and the dot product of the kernel and a certain position on the feature map is what will be used to form a new feature map, which is smaller than the previous feature map in size. The convolving steps over a feature map by a kernel matrix is taken in steps called strides. One stride means a kernel matrix moves one pixel to the right of a feature map and so on. The convolution operation can be represented mathematically, as shown in refs. [37, 40], by

$$h = f_{W,b}(x) = \sigma\big(\sum_{j=1}^{J} W_i^{(l)} \circledast x_j + b_i^{(l)}\big), \tag{3.3}$$

where W_i is a $n \times n$ kernel or filter matrix, b_i is the bias value for the i filter, x_j is the $j - th$ $m \times m$ input feature map in a group of J feature maps from the previous layer, $\circledast$ is a 2D convolution operation and σ is the activation function.

A new feature map has a size of $(m-n+1) \times (m-n+1)$, after a convolution operation which follows the architecture of an autoencoder. In the case of this project, the convolution step does not reduce the dimensionality of a feature map, max-pooling which will be explained in the next section is responsible for that operation. A convolutional neural network learns the best filter matrix values to minimise the cost function. A convolutional layer needs a non-linear activation function before another convolutional step is done.

A de-convolution neural network is basically an inverse convolution step used in the decoder part of the autoencoder. According to ref. [41], after a feature map is enlarged by using un-pooling or up-sampling, they are usually sparse. The de-convolution operation the density of the sparse feature map. This action also suppresses the noise in feature maps while learning the best filters to get a good reconstructed image. This can be represented mathematically as,

$$\hat{x} = g_{W,b}(h) = \sigma\big(\sum_{k=1}^{d} \tilde{W}_i^{(l)} \circledast h_j + b_i^{(l)}\big) \tag{3.4}$$

where $\tilde{W}$ is an inverse filter.

Figure 3.2 gives an example of such an architecture. It shows how the convolutional and pooling layers are followed by and connected to a fully connected neural network in the encoder part. Symmetrically, a fully-connected neural network followed by de-convolutional and pooling layers are used to reconstruct the image to its original dimension.

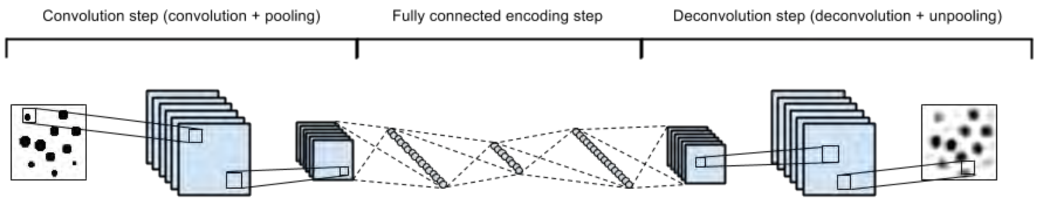
Convolution step (convolution + pooling)
Fully connected encoding step
Deconvolution step (deconvolution + unpooling)

Figure 3.2: A visual example of a convolutional autoencoder. Image source ref. [2]

3.2.2 Non-linear activation functions

The convolutional neural network, like other neural networks, learns the relationship between input and output and stores the learned parameters as filter values. Convolutional layers require a non-linear activation function for better performance. This activation function breaks up the linearity imposed when an image goes through a convolutional neural network. One can further read on why convolutional layers use non-linear activation function in Ref. [21]. The three most popular non-linear activation functions are the sigmoid function, the rectified linear unit (ReLU), and the parameterized ReLu (PRelu), but this project only explores the latter two.

ReLu

Rectified non-linearity layers in convolutional recognition systems were explored in ref. [22] and it was realised that they offer best performance compared to other activation functions. The author also learned that polarity is not an important factor in image recognition. This fact is explained in detail and mathematically in ref. [21]. The ReLU, introduced in ref. [23] is given by

$$\sigma(x) = \begin{cases} x & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x \leq 0, \end{cases} \tag{3.5}$$

where σ is the activation function and x is the input to the activation function. ReLU has some incredible benefits[5] that made it a default activation function for convolutional neural networks. For example, ReLU does not require a lot of computation power, which means that it takes less time to train. This also means that the neural network algorithm converges faster without saturation as the input variables increase. Arguably the most important trait of ReLU is that it introduces sparsity to the neural network. If different and less neurons are activated for a specific input during training, then the algorithm would give better predictive power and a lower case of overfitting.

As great as the ReLU sounds, it does have some shortcomings, the most common being dying neurons. This happens when a neuron keeps outputting a negative value which the ReLU would always takes to 0 i.e would not activate the neuron. This makes it difficult to update the neuron as a gradient descent algorithm does not update a weight that is not initially activated. This issue can sometimes be solved by a smaller learning rate. Alternatively, the activation function used in the next sub-subsection can be used.

PReLU

PReLU was introduced in ref. [24] as a solution to the shortcomings of the conventional ReLU. This activation function helps with dying neuron problem by allowing the neurons to be activated on negative values but with a very small margin. PReLU is defined by,

$$\sigma(x_i) = \begin{cases} x_i & \text{if} \quad x_i > 0 \\ a_i x_i & \text{if} \quad x_i \leq 0, \end{cases} \tag{3.6}$$

where a_i is a learn-able parameter and the i shows that each $i - th$ channel will learn a different a with respect to the cost function error value $\mathcal{E}$.
The authors mentioned that the value a_i is updated with the momentum method given as,

$$\Delta a_i = \mu \Delta a_i + \alpha \frac{\partial \mathcal{E}}{\partial a_i}, \tag{3.7}$$

[5]https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7

where μ is the momentum and α is the learning rate. The decay of weights in the neural network normally pushes a_i close to zero.

3.2.3 Max-Pooling and Un-pooling/up-sampling

Pooling layer in a convolutional autoencoder reduces the dimension of each feature map while retaining the important information about the feature map. The effects of pooling layers in convolutional neural networks were studied in ref. [26]. This project will use the max-pooling method. It is defined by,

$$x = \max_{N \times N}(x^{n\times n} u(n \times n)), \tag{3.8}$$

where $x^{n\times n}$ are the values in a pooling region $u(n \times n)$ and x are the maximum values in the pooling region. The feature map created by values from max-pooling has the same size as the pooling region to prevent overlapping. Ref. [26] mentions that max-pooling helps to improve the sensitivity of filters. It also introduces sparsity by removing non-maximum values in the overlapping sub-regions, which makes the algorithm more broadly applicable. It is also mentioned that max-pooling defeats the obvious need of weights regularisation, which is often done using L_1 and L_2 regularisers[6].

Interpolation is a process for increasing the dimension of small image by increasing the number of pixels to a bigger image using features from the smaller image. Unpooling and ups-ampling are commonly used methods in CAE. Ref. [42] states that in the convolutional neural networks, the max pooling operation is non-invertible, obtain an approximate inverse by recording the locations of the maxima within each pooling region in a set of switch variables. In the deconvnet, the un-pooling operation uses these switches to place the reconstructions from the layer above into appropriate locations, preserving the structure of the stimulus.

Up-sampling mentioned in ref. [43] is a method used to increase the dimension of an image by sampling from a smaller image. There are three types of up-sampling methods, namely nearest neighbour, bi-linear and bi-cubic. Only the first method will be used in this project. Nearest neighbour determines the values in the enlarged image by the nearest cell from the input (smaller) image. This means that the influence of any given pixel is limited to the nearest pixel.

[6]Defined in section 3.2.8

3.2.4 Softmax activation function

For multi-class output at the final layer (non-binary classification), we use the softmax activation function mentioned in ref. [27, 28]. It is used when the output of the neural network can be understood as a probability distribution. It is defined by,

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{K} e_k^x}, \tag{3.9}$$

where x_i is the input variable. It is a continuous and differentiable function which allows for gradient optimisation with respect to the loss function.

3.2.5 Cost function

The cost function (also called loss function when working with a single data point) is a method used to quantify the performance of an algorithm in terms of its ability to estimate the relationship between the input image and the reconstructed image. This is often expressed as the distance or difference between the actual value and predicted value. The cost function can be reduced with the help of gradient optimisers by updating the weights or filter values in a way that would give the lowest cost function, in turn increasing the performance of the algorithm. The cost function for convolutional autoencoder is the mean squared error mathematically defined as

$$\mathcal{E} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2 \tag{3.10}$$

Where x is the input image and $\hat{x}$ is the output image. It calculates the difference, the square root eliminates any negative values and gives more weight to large difference. The means is an average of a set of errors.

3.2.6 Stochastic gradient optimiser (Adam)

Gradient descent is an optimising method used in machine learning to find values of weights for a neural network which give the minimum cost function. Gradient descents conduct an iterative search to find weights that give the local minimum. This method is not ideal for machine learning as it utilises the whole dataset to compute an iteration which is computationally expensive and time consuming for training on big data. Stochastic gradient descent was adapted as a solution for a faster, less time consuming optimiser.

Instead of using the whole dataset, stochastic gradient optimiser uses a batch[7] of the dataset to optimise per iteration (normally referred to as epoch in machine learning context). This gives a noisy looking path to optimisation (which is common in most stochastic processes) since a different sample of the training data is used to find the local minimum at each iteration. Stochastic optimisers also required a larger number of iterations to converge to a local minimum because of all the noises introduced by the small batch size.

The optimiser used in this project is another more advanced form of stochastic gradient optimisers which is called the adaptive moment optimiser (or Adam for short). It was introduced in ref. [29]. Adam computes individual learning rates for different parameters from the estimate of the first moment (mean) and second moment (uncentered variance) of the gradient. It uses the exponential moving average of gradients to scale the learning rate. The Adam[8] stochastic optimiser can be expressed mathematically, as shown in ref. [44], by

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{V}_t} + \epsilon}\hat{S}_t, \tag{3.11}$$

where

$$\hat{S}_t = \frac{S_t}{1-\beta_1^t}, \tag{3.12}$$

$$\hat{V}_t = \frac{V_t}{1-\beta_2^t} \tag{3.13}$$

are the bias corrected estimate for first and second moment, and

$$S_t = \beta_1 S_{t-1} + (1-\beta_1)\frac{\partial \mathcal{E}}{\partial w_t}, \tag{3.14}$$

$$V_t = \beta_2 V_{t-1} + (1-\beta_2)\left[\frac{\partial \mathcal{E}}{\partial w_t}\right]^2, \tag{3.15}$$

where $S_0 = 0$ and $V_0 = 0$. The recommended default values for unvarying parameters as mentioned in ref. [29] are

- 0.001 for α,
- 0.9 for β_1,

[7]This is a number of samples from the dataset that is used for calculating gradient for each iteration. The dataset is well shuffled before the sample is extracted, and sampling is done randomly.

[8]https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

- 0.999 for β_2,
- $\times 10^{-8}$ for ϵ,

where β_1 and β_2 are used for decaying the running average and the running over of the square of gradient (they help correct the bias). α is the learning rate and ϵ is used to prevent division of zero. $\frac{\partial}{\partial w_t}\mathcal{E}(w, b; x, \hat{x})$ is the partial differentiation of the loss function with respect to a certain weight parameter. This reason, among others favourable, make the use of Adam a standard optimiser because of high computational efficiency, low memory requirement and little hyper-parameter tuning required.

3.2.7 Weight initialisation

Weight initialisation is a crucial factor for the development of any good performing neural network. Bad weight initialisation practice may lead to one of the two common problems caused by bad initial weights; which are the vanishing gradient problem and the exploding gradient problem, as mentioned in ref. [30]. The vanishing gradient problem happens when gradient gets smaller as the algorithm progresses down to the lower layers. This effect leaves the lower layer weights unchanged, which leads to the algorithm never learning to converge to the local minimum. The exploding gradient problem happens when the gradient gets large, which results in huge weight updates and the algorithm diverging away from the local minimum.

A solution to these problems was introduced in ref. [31], called the Xavier initialisation. What the authors suggested was to initiate weights by randomly sampling from a distribution with a mean of zero and ensuring that the variance of the input must be similar to the variance of the output. Sampling randomly ensures that the model will learn different functions about the data. This initialisation also requires the gradient to have equal variants when going in a layer and when going out in the reverse direction. That means layers are required to have the same size. This would be hard to achieve for an autoencoder as the algorithm architecture is heavily reliant on layers with different sizes with a constraint in the middle mandatory.

A solution to this limitation is called He initialisation introduced in ref. [24]. It calculates the range of the distribution depending on the number the previous layer's neurons and layer of focus, where the range is given as $[-r, r]$. The range is mathematically obtained by,

$$r = \sqrt{2}\sqrt{\frac{6}{n_{in} + n_{out}}} \tag{3.16}$$

and the variance by,

$$var = \sqrt{2}\sqrt{\frac{2}{n_{in} + n_{out}}}, \tag{3.17}$$

where n_{in} and n_{out} are input connections and output connections of the layer that weights are being initiated for, respectfully.

3.2.8 Regularisation

The common problem experienced in neural networks is overfitting. This is where the algorithm learns the training data well but performs less desirably on test/validation data. The solution for addressing this obstacle would be getting more data, which is not always possible. Alternatively regularisation can be used. The three common regularisation methods are L_1/L_2 regularisation, early stopping and dropout.

L_1/L_2 regularisation works by constraining large weights of a neural network by decaying those certain large weights that lead to overfitting. Early stopping works by stopping training of a neural network as soon as validation error reaches the minimum and starts diverging. Also, the model can be trained for all iterations and save the model that gives the minimum validation error during training.

Dropout as mentioned in refs. [27, 33] works by randomly dropping nodes and their entire connection in a neural network during training. This action prevents the network from being overly co-adapted to a certain set of data. The probability of a node being removed is set randomly but the threshold at which a node is kept or not is set manually. It was found in ref. [33] that dropout regularisation improved the performance of a neural network in a variety of application domains. It also gave better performance while being less computationally expensive compared to L_1/L_2 regularisation.

3.2.9 Hyperparameter tuning

Hyperparameters are the parameters that define the architecture of a machine learning algorithm. Hyperparameters are configured in the building of an algorithm and cannot be learned from training on data. A good selection of hyperparameters can make an algorithm perform efficiently and achieve maximum performance for a given dataset and minimising the validation error as much as possible, i.e ensuring that the model does not overfit. Examples of hyperparameters in this study would be learning rate, number of epochs, number of filters, filter kernel size and the number of neurons at the bottleneck of the autoencoder, to name a few.

Hyperparameter tuning can be done manually and automatically as mentioned in ref. [27]. Manual hyperparameter tuning requires a thorough understanding of the machine learning algorithm used, the problem being addressed, the data being used and the computational resources the researcher will be using. Automatic hyperparameter tuning is used in this study, with the focus being on the grid search optimiser and the random search optimiser. The grid search optimiser performs a brute force search for the best hyperparameters from a given small finite set of variables. This method tends to be computationally expensive as training has to be done for every combination of hyperparameters.

Random search optimisation is more applicable when dealing with large datasets compared to the optimiser mentioned earlier. It is able to explore a larger set of hyperparameters by taking the marginal distribution of hyperparameters, then sample from this distribution. A number of distributions like Gaussian or log-normal distributions can be used. An in-depth study of the comparison between grid search and random search optimisers was done in ref. [32], and showed how well random search performs over grid search. The authors concluded that the random search optimiser is computationally less expensive, it is much faster and finds better hyperparameter values since it explores a larger set. The disadvantage of both these optimisers is that they are not adaptive, which means that the previous outcome during the search does not affect the next.

3.2.10 Early stopping

The goal in deep learning is to avoid underfitting (model parameters are not trained well enough to give a satisfactory performance) by training the model long enough to give a desirable performance but not long enough that it overfits (learns statistical noise in the training data) on the training dataset making, it perform badly on test dataset. A solution would be to train on the dataset until the model's error measure with respect to an independent dataset (often called validation data) begins to degrade.

Early stopping is a common and more effective method to find this compromise. It stops the model from training when overfitting starts to be evident, keeping the generality of the model. There are three components to consider with early stopping. These are monitoring the model performance, a trigger to stop training and a choice of the model to use.

For this project, performance will be monitored using the loss function of both training and validation loss. A trigger method has to be selected carefully as a stochastic optimiser is used. This means that if the model performance is not smoothly improving a trigger has to account for this or the model will underfit. The best option as mention in ref. [45] is to us a delay or patience approach. This means that a model will keep training for a certain number of epochs and terminate if there has not been an improvement in all of them. For model choice ref. [27] suggests the best procedure would be to save the best model of a certain epoch and keep updating until the trigger is triggered. The model with parameters that gave the best performance will be chosen.

Chapter 4

Results and Discussions

This chapter will give details on the data used for this study along with the results obtained and the discussion. Most of the algorithm hyperparameters have been described in the previous chapters and will be referenced from there. The section is divided into two sections:

- The first part discusses testing robustness of the pre-processing method and validating that the CAE gives similar reconstruction error for events in low and high m/p_T. This will show that CAE reconstruction error is indeed de-correlated of mass. This step is crucial before full training the algorithm. A detailed study of robustness is done in the ref. [11] where they train on different low m/p_T phase space and test on a number of high m/p_T phase space.

- The second step is to benchmark the reconstruction error of the CAE, aiming to get a much better (in this case a reconstruction error which is the same or lower that obtained using mass correlated data) compared to results obtained using the AE in ref. [11]. Hyperparameter tuning will be used to determine which hyperparameters are important and contribute the most in increasing the performance of the algorithm with respect to the inputs used.

4.1 Data

The data used for this study were simulated using tools and method discussed in section 3.1. For background data, 5×10^6 events were simulated and 2.7×10^6 passed the selection process. For signal data, one million events were generated and only 35674 events passed the selection process of leading large radius jets and p_T of 350

GeV.

Both signal and background data were de-correlated on mass using the method mentioned in section 3.1.1. The parameters used for the pre-processing methods are $m_0 = 0.5$ GeV, $e_0 = 1$ GeV and a jet reconstruction radius of $R = 1$. The jet-images formed as in ref. [46] are viewed as η (rapidity) vs ϕ (azimuthal angles) plane with calorimeter entries as a sparsely filled image, where the filled pixels correspond to the calorimeter cells with non-zero energy deposits. The jet-images are all normalised where the pixel values range from 0 to 1. Figure 4.1 and 4.2 show an average of 30000 40×40 pixel images of both the background and signal respectively.

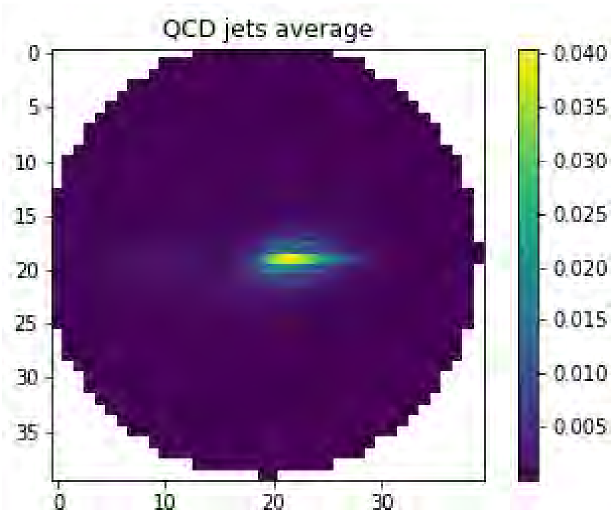

Figure 4.1: Average of QCD jet images.

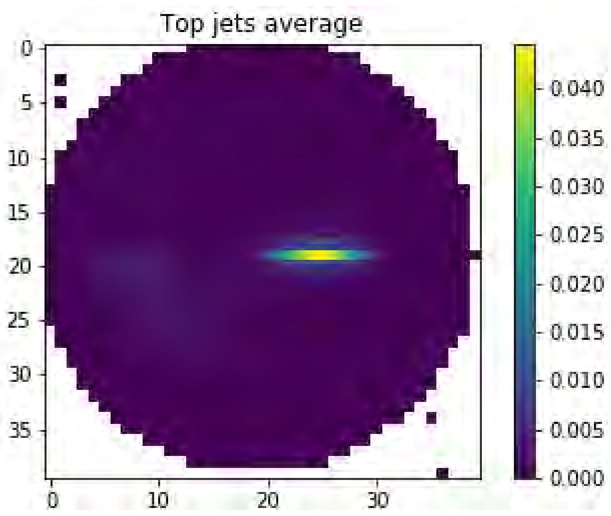

Figure 4.2: Average of Top jet images.

The colour bar on the right side of both images shows the intensity of the energy deposits. Due to computational infrastructure limitations only 4.8×10^5 jet-images were used to train the algorithm for robustness test and 8.9×10^5 backgrounds events where used to train the algorithm for the benchmarking test. 30000 signal and background images were used to test the algorithm. The need for a large dataset is so we reduce the dependency on regularisers and mitigate overfitting as much as possible since we training the algorithm with one class of data.

4.2 Algorithm

The structure of CAE used in this project comprises of hyperparameters and methods discussed in Section 3.2. As this is also a study to determine which parameters give the best performance, most hyperparameters were not set as constants. The initial structure of the CAE was taken from that used in ref. [4] and modified and tuned according to literature, to more improve performance.

The encoder consists of convolutional layers followed by a ReLU or PReLU activation and a max-pooling layer. The convolutional layer consists of 100-128 filter kernels, a 2×2 kernel size, and the kernel weights were initialised using He-uniform initialisation. There is no dimensional reduction at the convolution layer; this task is left to the max-pooling layer. Three sets of the above-mentioned layers are stacked together, then a flatten layer follows to structure the data for the fully-connected neural network layers. A fully-connected layer of 1000 neurons was used before the bottleneck of 30 neurons.

The decoder, as mentioned in Section 3.2 is symmetrical to the encoder. The only difference being the use of up-sampling in place of max-pooling, and de-convolution layer in place of convolution layer. A softmax activation function is used at the output layer so that it can restore the normalisation of the energy cluster in the jet-image and remove negativity. Adam optimiser with a learning rate of 0.001 and decay rate of 1×10^{-5} was used. A generator[1] was used to feed the algorithm a batch of data since the data was too large to be on the RAM (Random Access Memory). Batch sizes varied from $100 - 512$ and 6250 steps per epoch (epoch indicates a complete pass or training over the whole dataset), where used for 40 epochs. Early stopping with minimisation of validation loss as a metric (this means the algorithm will stop training when the validation loss stops decreasing), and a patience of 3 epochs was used (meaning the model stops training if validation loss doesn't decrease for three consecutive epochs).

It will be implemented using the high level API (Application Programming Interface) for deep learning called Keras[2] with Tensorflow[3] backend. Training was done

[1]Generators sample data from storage instead of loading to whole dataset to RAM. This insures that only the required data at a time is on the ram and available for processing unit when needed.

[2]Keras is a high level open source neural network library written in python. For more information please visit `https://keras.io/`

[3]Tensorflow is a free and open source software library for numerical computation and machine learning programmng. It was developed by Google Brain team. For more information visit the

on a Nvidia GTX 1080 ti Graphic Processing Unit (GPU) with 8 GB GPU RAM. The reason for using Keras with Tensorflow backend was mainly because keras, as a high level API provides a platform which makes the construction of neural networks fairly easy without the need of long codes which may results in errors. Tensorflow is one of the best deep learning API because it is supported by google, which means it is stable, frequently updated, large support platform, easy to debug and it gives computational graphs. Tensorflow is also highly optimise for parallel processing backend software, more especially for GPUs produced by Nvidia which is the GPU that was available during this project. GPU have large number of cores specifically designed to process data in tensor form, which allows for better computation of multiple parallel processes. Additionally, computations in deep learning need to handle huge amounts of data, this makes a GPU's memory bandwidth most suitable.

4.3 Results

As mentioned before, a test of robustness and bench-marking of CAE performance are conducted.

4.3.1 Robustness

The goal of this section is to show that the loss function of the CAE is unaffected by difference in jet mass by using a controlled sample of data. QCD jets from a certain phase space region is used to train the CAE and then test it on QCD jets from another phase space region. Figures 4.3 and 4.4 show the jet mass and transverse momentum distribution of the QCD jets from the Monte Carlo simulation, respectively. From these figures, we can see that there is a mass peak at a certain region, showing how jet mass is clearly a distinguishable feature of particles.

site `https://www.tensorflow.org/`

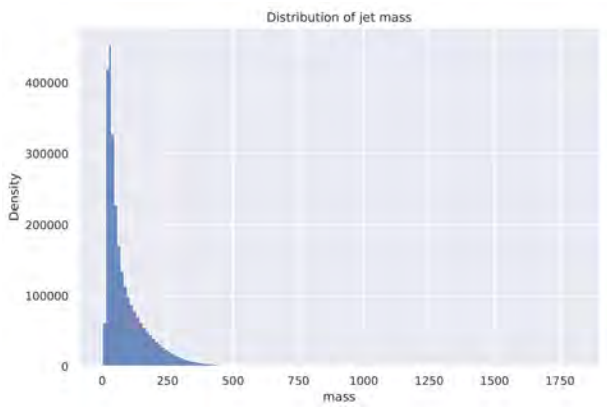

Figure 4.3: Distribution of QCD jet mass.

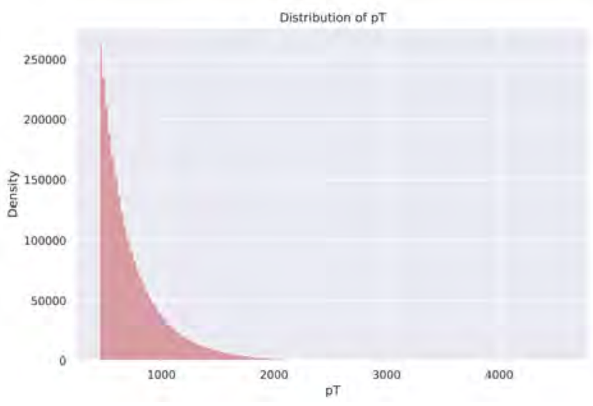

Figure 4.4: Distribution of QCD jet p_T.

Figure 4.5 shows the m/p_T distribution plot of the QCD jets and it can been seen that there are indeed more data at low m/p_T phase space compared to the high m/p_T phase space region.

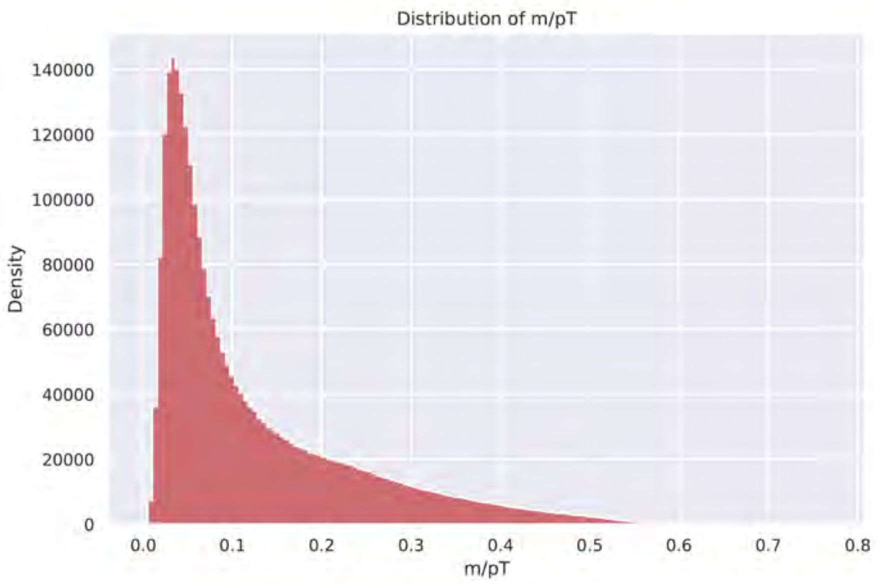

Figure 4.5: Distribution of m/p_T.

For the robustness test, a cut was made at $m/p_T = 0.1$ and all events below that value were used to train the CAE network while values above were used as a test sample.

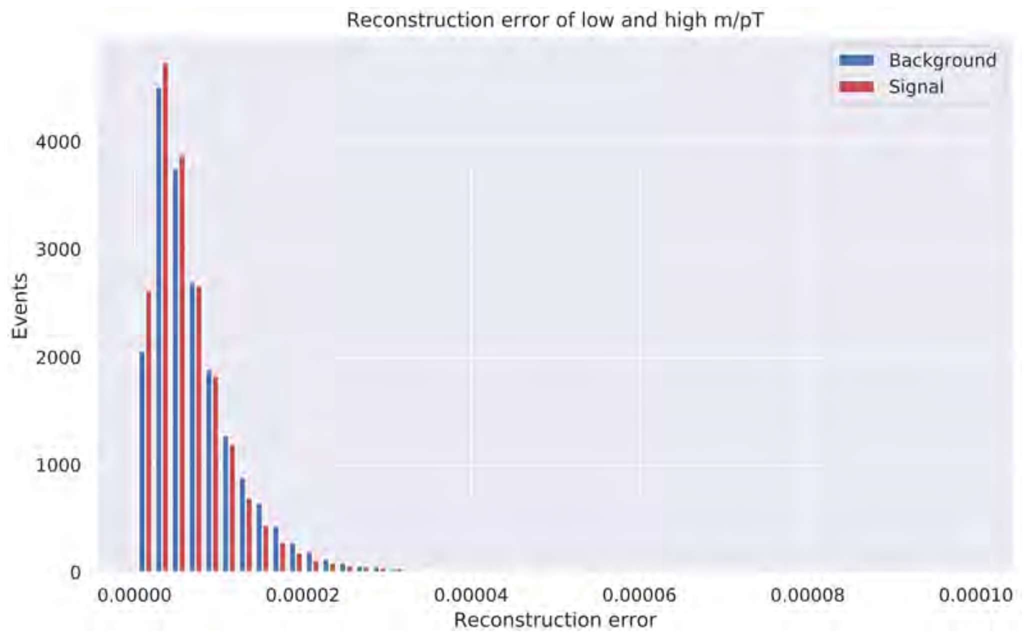

Figure 4.6: Loss function (reconstruction error) of CAE trained on low m/p_T

Figure 4.6 shows a bin to bin comparison of the test data from $m/p_T < 0.1$ which is the background and $m/p_T > 0.1$ which is signal in this case. One can clearly observe that the loss function from different phase space regions. This shows that the CAE loss function is not affected by the jet mass. Fairly similar results were obtained from different CAE hyperparameter configurations.

4.3.2 Reconstruction benchmarking

This subsection gives a benchmark of the performance of CAE trained on background and tested on the signal. As mentioned earlier, the top jet was used as an anomaly for the CAE. The QCD jets are expected to give a lower CAE loss value compared to the anomaly which the algorithm was not trained on. The reason is that QCD jets normally have 1-prong dipole emission structure. Top jets are with respect to QCD jets more complex as they have a 3-prong structure, which is a result of the particle decaying into a bottom quark and W boson, which has 2-prongs. The 3-prong structure is not clearly visible on figure 4.1 but the high density area is the W boson while the washed out area to the left is the bottom jet.

A Receiver Operating Characteristic (ROC) curve is a plot used to determine the diagnostic ability of a binary classifying system as the discrimination threshold is varied. The ROC curve and a bin-to-bin histogram will be used to visualise the results in order to quantify performance to be able to compare with results from other studies. For ROC curve, the top efficiency which is the fraction of top events for a certain threshold selection is represented on the x-axis while the y-axis represents the reciprocal of the background efficiency in log scale. Good ROC curve performance is when the line plot is close to the top right corner and a clear separation between distributions shows good performance on the histogram.

Figure 4.7 gives the reconstruction error distribution for signal and background from CAE.

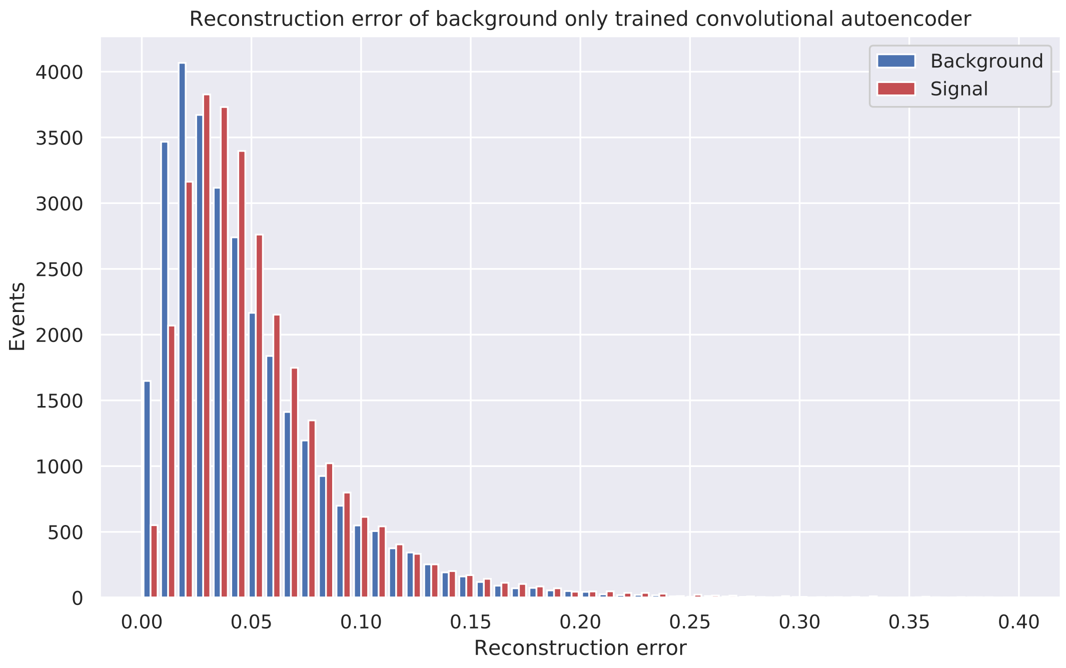

Figure 4.7: Reconstruction error of signal and background by CAE

Contrary to previous studies, one can clearly see that CAE performs very poorly. There is no clear separation between the distributions. One can observe a few background bins peaking more to the left and a few signal bins more to the right. Either way, these are unsatisfactory results because the model has to be trained unsupervised on ATLAS data which as mentioned, is not labelled.

Figure 4.8 visualises the ROC curve of reconstruction error values of CAE. By observing this line plot, one can see that there is no discrimination of signal distribution over background distribution. The area under the curve value is 4.878828, which is another indication of the poor performance.

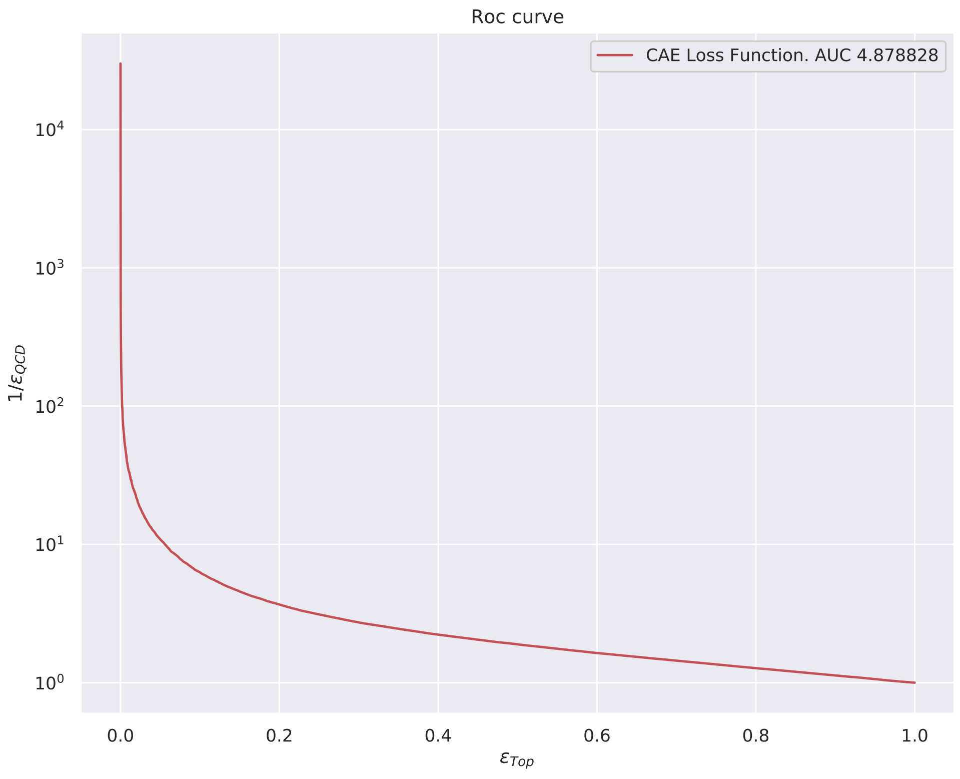

Figure 4.8: ROC curve of CAE

The CAE performance did not change much even after months of hyperparameter tuning. The reasons for the poor performance will be discussed in the following section along with results to support the claim.

4.4 Discussion

As stated before, the CAE was expected to give better performance over fully-connected neural networks or most other dimensional reduction method for image data. Surprisingly, this was not the case, as shown in results from the previous subsection. The performance was poor in comparison to results obtained in ref. [11]. This poor performance can be associated with two main factors, which are:

- The data, and how the pre-processing methods de-correlate the jets of jets mass was a major factor. This is supported by ref. [10] which mentions that jet mass is a significant aspect used for distinguishing jets and tagging particles. On top of this, jet image data are very sparse.

- The second factor would be how a convolutional neural network operates.

As mentioned in subsection 3.1.1, the de-correlation method works by re-scaling the jet by setting the energy of all jet constituents to 1 GeV. Jet mass is calculated using energies of the constituent particles along with opening angles of all constituents[4]. By re-scaling the jet constituents and setting the energy to a constant, the jet images lose an important discriminating factor which, is jet mass correlation. As shown in ref. [4], jet mass correlation affects the performance of the CAE greatly, where a higher jet mass results in a higher reconstruction error. Since we eliminated this correlation, CAE performance has significantly decreased. Another problem is caused by small dynamic range of pixel values for both signal and background image data. Figures 4.1 and 4.2 show a mostly uniform purple colour for the most part of both images. This is problematic when used to train convolutional autoencoders, as it becomes difficult for the algorithm to detect the difference in the structure of jets.

Also, as stated earlier, convolutional layers perform a convolution operation over an input image using a certain number of kernels or filters to learn the best weight to best express features in the input data. After that, the max-pooling layer is used to reduce the dimensionality of an image, as mentioned in subsection 3.2.3. These operations make convolutional networks superior in image recognition are rendered useless by the small dynamic range in pixel values. Because of the initialisation method used for kernels, the values are not so deviated from each other, meaning that most of the filters identify similar features from the image in the convolution operation.

Max-pooling is also rendered ineffective by the similarity of pixel values. Even though top and QCD jets have different structures, max-pooling is unable to identify this feature because it would not matter whether there is 1 or 3 pixels for a given kernel; the same value will still be passed to the next layer and so on. Up-sampling and de-convolutional layer on the decoder part are also affected in the same manner.

The reason suggested above can be proved if a fully-connected autoencoder is trained, validated and tested on the same data, but giving better reconstruction

[4]The momentum of particles is also used and obtained from curvature observed in the tracking chamber.

performance. Results from a crude hyperparameter un-optimised fully connected AE give better performance, as shown in Figure 4.9. One can see the QCD jets were peaked to the left then their distribution falls rapidly while top jets are more to the right of the distribution. This of course did not lead to better results than those mentioned in previous literature but was used to support the reasoning stated above.

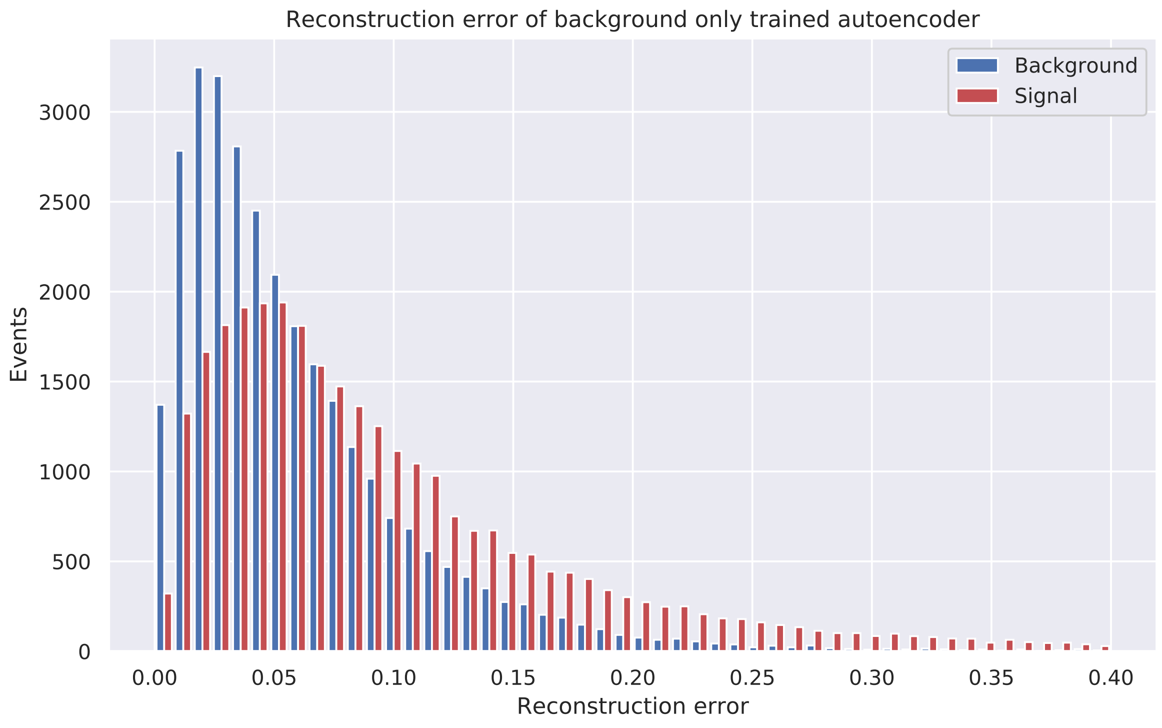

Figure 4.9: Reconstruction error of signal and background by AE.

The ROC curve in figure 4.10 shows that fully connected AE does give better performance compared to CAE. The area under the curve for AE is more than twice that of CAE.

A couple of hyperparameters were observed to improve CAE performance when used. For example batch size set at 128 inputs seemed to give a faster convergence to the local minimum. The dropout decreased the performance of the network and is not recommended for fairly shallow or medium architectures. PReLU gave better results during training compared to the ReLU activation function but increased the training time. This was because it accommodated negative values which reduced the effects of vanishing gradient observed when training with ReLU. The vanishing gradient happens because the pixel values used to train are very small numbers and can lead to dying neurons in some cases when using ReLU. He initialisation was used as a standard after giving constant decrease in loss function without being stuck at a local minimum.

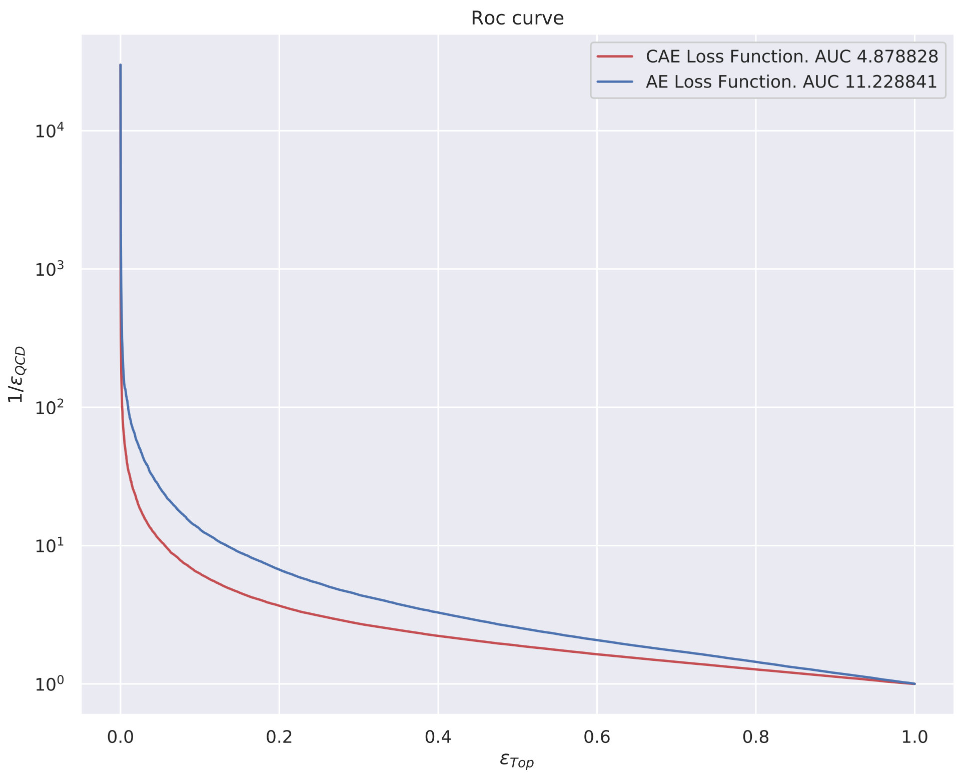

Figure 4.10: ROC curve of AE and CAE.

The following chapter gives a conclusion for the project. It also discusses limitations experienced and recommendations that can be explored in future studies.

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

This project explored the use of convolutional autoencoder as an anti-QCD tagger. The tagger was trained on low m/p_T region where data is abundant and tested on high m/p_T where data is scarce. Simulated mass de-correlated jet image data pre-processed with the method introduced in ref. [11] were used. Light quarks and gluon jets were used to train the algorithm while the top jets were used as anomalies to test the CAE. Mass de-correlation allows for searching deviations from QCD jets where training data is minimal and the unsupervised learning approach means the tagger can be implemented directly on data. The CAE reconstruction error (loss function) was used as a metric to indicate anomalous top jets from QCD background jets. QCD jets were expected to give lower reconstruction error while top jets give a higher value.

The results obtained from the study were not what was expected as seen in the previous section. The CAE performance was affected by the small dynamic range in pixel values for both QCD and top jets images which made the convolution and max-pooling/up-sampling operation less effective. This reason for poor performance can be supported by results from fully-connected autoencoder, which performed better with a crude un-optimised architecture. There were other limitations that affected the performance of the algorithm. These are discussed in the next section.

5.2 Limitations

The major limiting factor was computational power. Access to GPU was limited to one cluster node, which had a time limit of 120 hours. This limited the ability to run full hyperparameter tuning optimisation as one run would normally take 100 hours. Another problem was that dropout (mentioned in subsection 3.2.8) significantly reduced the performance of the algorithm and was not idea in reducing overfitting. This meant that a large dataset was needed to be used so the model doesn't fit which made the training time more than the allocated time per cluster node in result stopping training before the algorithm converges. Up-sampling is not the best interpolation method as it pixilated the whole image while the input image was sparse. Interpolation method, which pairs well with max-pooling called unpooling, was not available on the Keras deep learning packages used in this project. Building a package efficiently would have taken more time and extended beyond the scope of this project.

5.3 Recommendations

There are a number of methods that can be used to improve the design of a robust anti-QCD tagger. The first recommendation would be the use of coloured jet images as done in ref. [48]. Coloured in this context means additional information to the jet image instead of conventional energy deposits grey scale jet images. Colour represents transverse momentum of neutral particles, transverse momentum from tracking chamber and the number of tracks per pixel. The use of colour jet images improved performance in other studies and should work for this case.

The use of detector-level simulated data is recommended in future studies as it simulates a particle's passage and interactions through different layers of the ATLAS detector. Detector simulations include pile-ups which are caused by additional proton-proton collision of other protons from the bunch, and underlying events, which are caused by other parton interacting in proton-proton collision either than the main interacting partons. These often affect the performance of the tagging algorithm and should be tested.

Another method to explore would be using Lund jet planes mentioned in ref. [49] instead of conventional jet images. Lund plane serves as a phase space within jets mapped to a triangle in a two dimensional plane that shows the logarithmic transverse momentum and angels of any given emission with respect to its emitter. A primary Lund plane only contains position of emissions as different colours should be used.

The machine learning section of the study has multiple features that can be explored given one has an advanced GPU at their disposal to train and perform hyperparameter optimisation. Un-pooling can be used instead of up-sampling as it stores the position of results from max-pooling, restoring them to the exact position in the decoder part of the autoencoder. Hyperparameters like larger batch size, learning rate and different gradient optimisers can be explored at a wider range. Advanced CNN architectures like AlexNet in ref. [50], Fast R-CNN in ref. [51], ResNet in ref. [52] and GoogLeNet in ref. [53] can be explored and converted into autoencoders (this means changing the architecture of a classification deep learning algorithm to a unsupervised algorithm resembling the structure of an autoencoder) to see if they may result in better output reconstruction of the input. Convolutional long short-term memory recurrent neural network as mentioned in refs. [54, 55] can also be explored since it has shown promising results when trained and implemented on Lund jet plane inputs.

Appendix

More ROC curve results

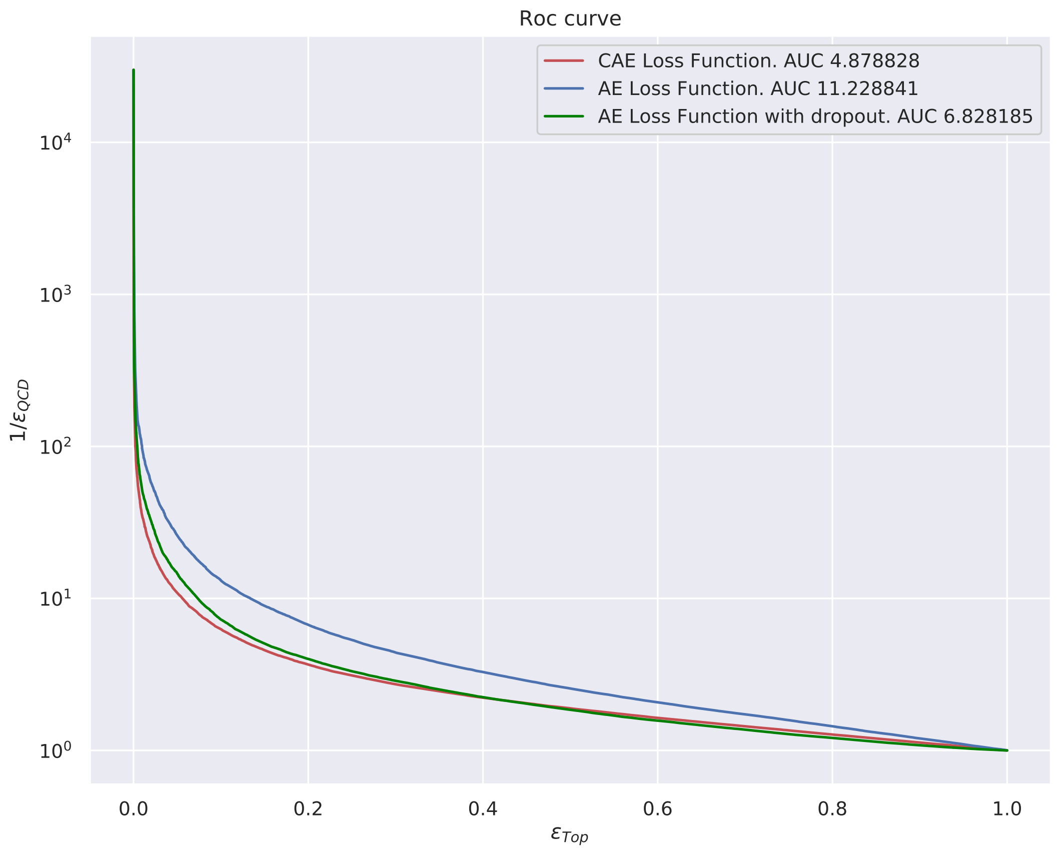

Figure 5.1: Results showing how poorly algorithm with dropout perform.

Bibliography

[1] Atlas Collaboration. Performance of top-quark and W-boson tagging with ATLAS in Run 2 of the LHC. *arXiv preprint arXiv:1808.07858*, 2018.

[2] Muyuan Ke, Chunyi Lin, and Qinghua Huang. Anomaly detection of logo images in the mobile phone using convolutional autoencoder. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1163–1168. IEEE, 2017.

[3] Deepak Kar. *Experimental Particle Physics*. IOP Publishing Limited, 2019.

[4] Marco Farina, Yuichiro Nakai, and David Shih. Searching for new physics with deep autoencoders. *arXiv preprint arXiv:1808.08992*, 2018.

[5] ATLAS Collaboration. Identification of boosted, hadronically-decaying W and Z bosons in $\sqrt{s}$= 13 TeV Monte Carlo Simulations for ATLAS. Technical report, ATL-PHYS-PUB-2015-033, 2015.

[6] ATLAS collaboration. Boosted hadronic top identification at ATLAS for early $\sqrt{s}$= 13 TeV data. Technical report, ATL-PHYS-PUB-2015-053, 2015.

[7] ATLAS Collaboration. Identification of boosted, hadronically decaying w bosons and comparisons with atlas data taken at $\sqrt{s}$= 13 TeV. *Eur. Phys. J. C*, 76:154, 2016.

[8] Davison E Soper and Michael Spannowsky. Finding physics signals with shower deconstruction. *Physical Review D*, 84(7):074002, 2011.

[9] Tilman Plehn, Michael Spannowsky, Michihisa Takeuchi, and Dirk Zerwas. Stop reconstruction with tagged tops. *Journal of High Energy Physics*, 2010(10):78, 2010.

[10] ATLAS Collaboration. Performance of mass-decorrelated jet substructure observables for hadronic two-body decay tagging in ATLAS. Technical report, Tech. Rep. ATL-PHYS-PUB-2018-014, CERN, Geneva, 2018.

[11] Tuhin S Roy and Aravind H Vijay. A robust anomaly finder based on autoencoder. *arXiv preprint arXiv:1903.02032*, 2019.

[12] Theo Heimel, Gregor Kasieczka, Tilman Plehn, and Jennifer Thompson. QCD or What? *SciPost Physics*, 6(3):030, 2019.

[13] 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. *Computer Physics Communications*, 191:159–177, 2015.

[14] Simone Alioli, Paolo Nason, Carlo Oleari, and Emanuele Re. A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX. *Journal of High Energy Physics*, 2010(6):43, 2010.

[15] Johan Alwall, R Frederix, S Frixione, V Hirschi, Fabio Maltoni, Olivier Mattelaer, H-S Shao, T Stelzer, P Torrielli, and M Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *Journal of High Energy Physics*, 2014(7):79, 2014.

[16] Gregor Kasieczka, Tilman Plehn, Michael Russell, and Torben Schell. Deep-learning top taggers or the end of QCD? *Journal of High Energy Physics*, 2017(5):6, 2017.

[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[18] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[19] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[20] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)*, pages 1–5. IEEE, 2018.

[21] C-C Jay Kuo. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41:406–413, 2016.

[22] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference On Computer Vision*, pages 2146–2153. IEEE, 2009.

[23] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on Image-Net classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[25] AL Maas, AY Hannun, and AY Ng. Rectify nonlinearities improve neural network acoustic model. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

[26] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks*, pages 92–101. Springer, 2010.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* Cambridge Massachusetts, MIT press, 2016.

[28] Christopher M Bishop. *Pattern recognition and machine learning.* New York, Springer, 2006.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, Inc., 2017.

[31] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[32] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[35] Fu-Jie Huang MarcâAurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPRâ07). IEEE Press*, volume 127, 2007.

[36] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

[37] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

[38] Seungyoung Park, Myungjin Kim, and Seokwoo Lee. Anomaly detection for HTTP using convolutional autoencoders. *IEEE Access*, 6:70884–70901, 2018.

[39] Manassés Ribeiro, André Eugênio Lazzaretti, and Heitor Silvério Lopes. A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, 105:13–22, 2018.

[40] Volodymyr Turchenko, Eric Chalmers, and Artur Luczak. A deep convolutional auto-encoder with pooling-unpooling layers in Caffe. *arXiv preprint arXiv:1701.04949*, 2017.

[41] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[42] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[43] Philippe Thévenaz, Thierry Blu, and Michael Unser. Image interpolation and resampling. *Handbook of Medical Imaging, Processing and Analysis*, 1(1):393–420, 2000.

[44] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[45] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the Trade*, pages 55–69. Springer, 1998.

[46] Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwarztman. Jet-images: computer vision inspired techniques for jet tagging. *Journal of High Energy Physics*, 2015(2):118, 2015.

[47] Sebastian Macaluso and David Shih. Pulling out all the tops with computer vision and deep learning. *Journal of High Energy Physics*, 2018(10):121, 2018.

[48] Patrick T Komiske, Eric M Metodiev, and Matthew D Schwartz. Deep learning in color: towards automated quark/gluon jet discrimination. *Journal of High Energy Physics*, 2017(1):110, 2017.

[49] Frédéric A Dreyer, Gavin P Salam, and Grégory Soyez. The lund jet plane. *Journal of High Energy Physics*, 2018(12):64, 2018.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[51] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[53] Jeffrey Dean. Large-scale deep learning for building intelligent computer systems. 2016.

[54] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[55] Kazi Nazmul Haque, Mohammad Abu Yousuf, and Rajib Rana. Image denoising and restoration with CNN-LSTM Encoder Decoder with direct attention. *arXiv preprint arXiv:1801.05141*, 2018.