Investigation of Autoencoders for Jet Images in Particle Physics

JESSICA LASTOW | DEPARTMENT OF PHYSICS | LUND UNIVERSITY | 2021

Investigation of Autoencoders for Jet Images in Particle Physics

Jessica Lastow

Lund University

Department of Physics

Lund University
Department of Physics
Division of Particle Physics
Box 118
SE-221 00 Lund
SWEDEN

Abstract

Dark matter is an invisible type of matter believed to make up 85 % of the matter in the universe, but it has not yet been identified by experiments. According to certain particle physics theories, possible signatures of dark matter are so-called *dark jets*. They are the dark matter equivalent to classic jets, i.e collimated streams of particles. These jets could be produced from the proton-proton collisions at the Large Hadron Collider (LHC). The ATLAS experiment at the LHC is currently searching for such jets.

There are two challenges in discovering dark jets. Firstly, the traces left in the detector by dark jets are not well understood, so the best lead is to search for something anomalous. Secondly, current data storage limitations force us to discard data and this reduces the probability to register a dark jet.

Using machine learning techniques – or more specifically autoencoders – is a proposed solution as autoencoders can perform compression and anomaly detection simultaneously. This master's thesis investigates the use of autoencoders for jet images, a two-dimensional jet representation. A group of different autoencoders are trained separately to learn inherent structures in QCD jets (background, ordinary events). They are then used to recognize boosted W-boson jets (signal, anomalous events) with a different signature. The separation of boosted W-boson jets and QCD jets is a simplified version of the problem of separating possible dark jets from QCD jets.

The autoencoders were able to compress the background jet images threefold with an error of less than 5 % for over 95 % of the data. However, for signal jet images the error was found to be even smaller. This made anomaly detection impossible since the opposite is required for the method to work. The difference between signal and background could be too small for the simple autoencoders to distinguish.

Keywords: data compression, anomaly detection, jet image, autoencoder, dark matter, machine learning, ATLAS, hadron jet, particle physics, high energy physics

Preface

This thesis report was submitted for the degree of *Master of Science in Engineering, Engineering Physics* at the Faculty of Engineering (LTH) at Lund University in February 2021.

Course: Degree Project in Physcis, 30 ECTS (PHYM01)

Principal supervisor: Caterina Doglioni
Assistant supervisor: Antonio Boveia
Examiner: Oxana Smirnova
Student opponents: Olof Englund, Karl Pettersson, Raheleh Mahdivand Avilagh

Acknowledgements

I would like to express my gratitude towards my supervisors Caterina Doglioni and Antonio Boveia and for their guidance and support in the completion of my thesis. I also want to thank my examiner Oxana Smirnova and student opponents Olof Englund, Karl Pettersson and Raheleh Mahdivand Avilagh for providing me with valuable feedback.

I want to especially emphasize the appreciation to my principal supervisor Caterina Doglioni. Not only has she given me encouraging words and incredible insights throughout the entirety of the project, she has also been an invaluable mentor in my future career. I want to thank her for giving me the opportunity to pursue this project and thank her, and Alice Ohlson, for being such inspiring lecturers and giving me no doubt that it is particle physics I want to pursue.

Lastly, I want to express my love and deepest appreciation for my family and friends. They have given me unconditional support and kept me motivated during the times when I needed it the most.

Contents

List of Abbreviations

SM	Standard Model
QED	Quantum Electrodynamics
QCD	Quantum Chromodynamics
CERN	European Organization for Nuclear Research
LHC	Large Hadron Collider
ATLAS	A Toroidal LHC Apparatus
ML	Machine Learning
ANN	Artificial Neural Network

Chapter 1

Introduction

1.1 Background

Dark matter is one of the biggest mysteries in modern physics. We know that this invisible type of matter should exist, but it has not been found yet [1]. One experiment where dark matter particles could be created and detected is ATLAS, which is one of the four main experiments at the world's largest particle accelerator – the Large Hadron Collider (LHC) at CERN in Switzerland [2]. At the LHC, bunches of protons are accelerated to extremely high energies and then brought together to collide [3]. These high-energy collisions can produce new particles that are then detected by ATLAS – some of which could be dark matter [1].

Many possible signatures of dark matter are rare and buried in already known processes (backgrounds). We cannot save the products of every proton-proton collision event as the collision frequency is very high (up to 40 MHz) and the available storage space is not sufficient. This means that we have to discard events – both dark matter signals and background – which limits our sensitivity to these signals. If the data were to be compressed without losing relevant information, then more collision events could be stored with the available storage capacity [4].

However, another problem is that we do not yet know what the signature of dark matter could look like in a detector. If we only search for signatures based on existing theories of dark matter, we may miss interesting but not yet theorized dark matter candidates. If we were able to look for something that is out of the ordinary – something *anomalous* – we would be able to cast a wider net on dark matter.

An *autoencoder* is a type of artificial neural network that can, through training, learn to compress and decompress data. It learns to identify inherent patterns and structures of the data and stores these in a lower dimensionality [4]. There is no universal autoencoder architecture that can compress everything. Autoencoders can only compress data with a certain structure that they have been previously trained on. This characteristic can be utilized: If the autoencoder learns to compress ordinary collision signatures and then is shown something different, it would fail at compressing it. The failure of the autoencoder to compress anomalous events can be quantified, e.g through the reconstruction loss [5]. The reconstruction loss is the difference between the original and compressed quantities. We can then detect anomalous events as those passing a certain reconstruction loss threshold. This feature can be used to find interesting rare events for new physics – perhaps an event that shows traces of dark matter.

1.2 Overall Objective

The overall objective of the thesis project is to test the ability of an autoencoder to perform compression and anomaly detection, i.e distinguish differences between ordinary and anomalous signatures, on certain types of physics processes. The data contains images of jets, which are collimated streams of particles produced in proton-proton collisions. Another objective is to produce well-documented and reproducible code that is accessible for researchers of any level.

1.3 Research Questions

- Is the autoencoder able to compress and decompress images of jets?
- Is the autoencoder able to detect anomalies in images of jets?

1.4 Related Work

Deep Autoencoders for Data Compression in High Energy Physics [4] by Eric Wulff is a proof-of-principle study of using machine learning for compression of ATLAS data. This work successfully proved that autoencoders can be used to reduce the dimensionality of jet data without a significant loss in performance. Wulff used two types of data: only the kinematic properties of the jets (the four-momentum), and also most of the jet variables available to ATLAS (27-dimensional jet data). The baton was then passed on to Erik Wallin who in his bachelor's thesis project *Tests of Autoencoder Compression of Trig-*

ger Jets in the ATLAS Experiment [6] studied compression of 4-dimensional trigger-level analysis (TLA)[1] jet data and computing constraints, and compared it to another kind of compression. Honey Gupta then worked on event-level data in *Deep-compression for High Energy Physics data* [7] for Google Summer of Code 2020. Event-level data means that it includes other objects than just jets, e.g. particles like electrons, muons and photons, but only in terms of four-momentum.

There are two main works that are of interest for jet images: *Jet-Images – Deep Learning Edition* [8] by de Oliveira, et.al. and *Jet-Images: Computer Vision Inspired Techniques for Jet Tagging* [9] by Cogan, et.al. They introduce jet images as a tool for utilizing deep learning and computer vision for tagging of jets.

[1]Trigger-level analysis is a technique where only a partial event is saved from the trigger, e.g only jets, instead of the full detector information [6].

Chapter 2

Theory and Experimental Background

The purpose of this chapter is to give an overview of the underlying theory of the main concepts in this thesis: dark matter, jets and autoencoders. The chapter begins with the theory of the Standard Model and the origin of jets. Then, an introduction of dark matter and proposed dark jets is given. The chapter then continues onto the experimental side of particle physics and introduces the ATLAS Experiment at the Large Hadron Collider (LHC) and its inner workings. This is necessary in order to understand the detection of jets and how this can translate to a jet image. The chapter then switches to the field of artificial neural networks, the theory of which autoencoders rely on. By utilizing autoencoders, compression and anomaly detection can be performed. This will be described towards the end of the chapter.

For the sake of readability some words are bolded to guide the reader to explanations of important concepts. The reader is assumed to have a basic understanding of special relativity (see Appendix A).

2.1 Particle Physics

2.1.1 The Standard Model

The Standard Model (SM) is the current theoretical model that describes the fundamental components of the universe. According to the Standard Model, the components of everything that exists can be reduced to two groups of particles: fermions and bosons. To our knowledge, these are elementary particles, meaning that they have no substructure and are the smallest constituents of matter [10].

Fermions are particles with half-integer spin. For example, every atom in nature is made out of a combination of two composite fermions – the *proton* and the *neutron* – and a fermion called the *electron*. Fermions are then divided into two families: *quarks* and *leptons*. In Table 2.1 is a list of the fermions in the Standard Model [10].

Table 2.1: Fermions in the Standard Model.

Quarks	Up (u)	Charm (c)	Top (t)
	Down (d)	Strange (s)	Bottom (b)

Leptons	Electron (e)	Muon (μ)	Tauon (τ)
	Electron neutrino (ν_e)	Muon neutrino (ν_μ)	Tau neutrino (ν_τ)

Table 2.2: Bosons in the Standard Model.

Force	Electromagnetic	Weak	Strong
Boson	Photon (γ)	$W^{\pm}$-boson, Z-boson	Gluon (g)

Mechanism	Higgs mechanism
Boson	Higgs boson (H^0)

Bosons are particles with integer spin and they include the force carriers for the fundamental forces. These are exchanged between particles during interactions. Each fundamental force has a corresponding boson. Table 2.2 contains a list of the bosons in the Standard Model [10]. The Higgs boson differs from the other bosons as it is not connected to a force. Instead it is connected to the Higgs mechanism. This mechanism explains why bosons have mass.

Each particle has an antiparticle counterpart. An **antiparticle** is an associated particle with the same mass but opposite charges, e.g the electron (e^-) with negative charge and its antiparticle the *positron* (e^+) with positive charge. An antiparticle is denoted with a bar, i.e if q is a quark then $\bar{q}$ is an anti-quark [11].

The **electromagnetic force** acts between electrically charged particles. It is the force that binds the electron to the atomic nucleus and it is the origin of e.g. visible light. The theory of electromagnetic interactions is called **Quantum Electrodynamics (QED)**. The **weak force** is the force responsible for β-decay, which is the radioactive decay of certain atoms. The **strong force** is what binds most matter together, for example the quarks in protons and nucleons in the nucleus. The theory of strong interactions is called **Quantum chromodynamics (QCD)**. In this thesis, QCD will be of special interest. For more resources and information on the material in this chapter, see Ref. [10].

The strong force acts on particles with **color charge**, which is the QCD-equivalent of electric charge in QED. Particles with color charge are quarks and gluons and they can be red (r), green (g), blue (b) or anti-red ($\bar{r}$), anti-green ($\bar{g}$) and anti-blue ($\bar{b}$). Note that the color charge is not the physical color of the particle. It is simply an inherent property like mass or electric charge. The characteristics of color charges give the strong interaction peculiar features like *color confinement* and *asymptotic freedom* [10].

Color confinement means that quarks are confined within so called *colorless states*. Colorless can be seen as the QCD-equivalent of neutral electric charge. Confinement is also why free quarks are never observed, since they are not colorless. Another consequence of color confinement is that **hadrons**, which are compositions of quarks, can only exist in certain configurations. The most common hadrons can either be **mesons**, a quark and an anti-quark ($q\bar{q}$), or **baryons**, three quarks bound together (qqq or $\bar{q}\bar{q}\bar{q}$) [10].

Asymptotic freedom is the strength-distance behavior of the strong force. It is best explained with a classic spring analogy. Consider a spring with each end representing a quark. If the spring is barely extended from its relaxed state then the spring force is small. When the extension of the spring is increased, then the spring force also increases. If the force is strong enough, and in turn have enough potential energy, then the spring snaps and two new ends appear. A similar behavior is seen between quarks. At short distances, the strong force is weak. As distance increases, it becomes stronger. If quarks are separated enough then it is more energetically favorable to create a quark-antiquark pair. In high energy collisions (see more details in Section 2.2), there is enough energy to keep stretching and breaking this spring – creating a stream of quark-antiquark pairs (see Figure 2.1). These collimated streams of particles are called hadronic jets, or simply just **jets** following the so-called **hadronization** of quarks. In Section 2.3, the discussion about jets will continue.

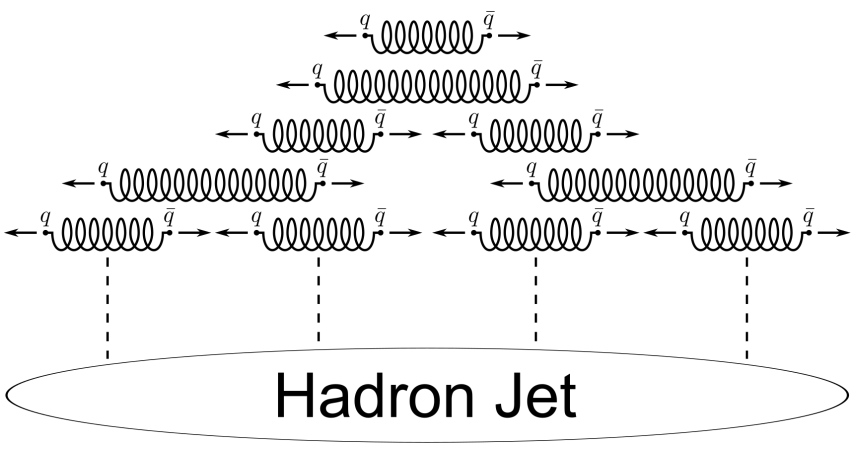


Figure 2.1: A schematic showing the production of a hadron jet using the spring metaphor. © Wikimedia Commons User: Lokal_Profil / CC-BY-SA-2.5.

The Standard Model is a successful model in many ways. It is consistent with most experimental observations in particle physics and has been able to predict many new particles so far. However, it fails to give an explanation to some cosmological problems and fundamental phenomena like gravity. Physicists therefore expect to find phenomena **Beyond the Standard Model (BSM)** and develop theories to e.g. explain the nature of *dark matter* [10].

2.1.2 Dark Matter

The force that binds matter like stars, planets and gas into galaxies is gravity. However, according to mass measurements, the observable mass in galaxies is incapable of generating the gravity needed to hold them together. The swiftly moving objects in the outskirts of the galaxy should be torn away – but this is not what is observed. There must be something else that holds the galaxies together [1].

Dark matter (DM) is a theorized type of matter that solves this discrepancy. It does not interact with the electromagnetic force in any way – it neither absorbs, reflects or emits light. That is why it is called *dark* matter. As a consequence, it is difficult to detect. However, since it is massive it can infer a gravitational pull, thus generating enough force to hold the galaxies together [1].

Finding dark matter is difficult, even though dark matter is *everywhere* and it is more abundant than normal SM matter. Normal SM matter is believed to be only a small fraction of the contents of the universe [1]. We have a large amount of evidence for the existence of dark matter but we know very little about what it is. In fact, we know more about what it is not, than what it actually is. A review of the accumulated evidence for dark matter can be found in Ref. [12] and the history of dark matter in Ref. [13].

In the Standard Model, there are several different elementary particles that make up matter. Quarks are combined into hadrons, that in turn can be combined into atomic nuclei. Add more particles and we get atoms, chemical compounds and in the end we have humans, mountains, stars and galaxies. There is no reason to believe that dark matter would not be equally complex. It is therefore plausible that dark matter particles are part of an unobserved **dark sector** – a collection of new particles, forces and interactions – rather than dark matter consisting of only one fundamental particle [14]. In some theories, this dark sector includes a portal to the Standard Model, making it possible to see traces of it in the SM world. This portal could be a new particle that interacts with both the DM and SM sector, but with a weaker coupling to SM particles [15][16].

One theory is that the dark sector includes an interaction that resembles the strong interaction in the Standard Model. With this in mind, a dark matter model can be constructed with parallels to QCD. Confinement in the dark sector implies that dark matter can be a composite object, e.g. dark hadrons [14][15]. An equivalent to asymptotic freedom would suggest the possibility of hadronization and consequently **dark jets** [14]. However, the dark hadrons and jets are invisible to current detectors, unless they can interact with SM particles. With the assumption that a portal exists to connect the SM with the dark sector, we get a whole new dark category of different possible jet signatures.

2.2 The ATLAS Experiment

The Large Hadron Collider (LHC) [17] is the world's largest particle accelerator. It is 27 km in circumference and is situated at the CERN research facility on the border of Switzerland and France. At the site, about 100 meters underground, there are two separate beam pipes where protons travel close to the speed of light in opposite directions. These beams intersect at four points around the ring where the protons are brought together to collide [3] (see Figure 2.2). The products of each collision are referred to as a collision event, or simply just an **event**. The protons that circulated in the most recent data-taking run of the LHC have an energy of 6.5 TeV [3]. This means that there are 13 TeV of available center-of-mass energy to produce new particles per event [11].

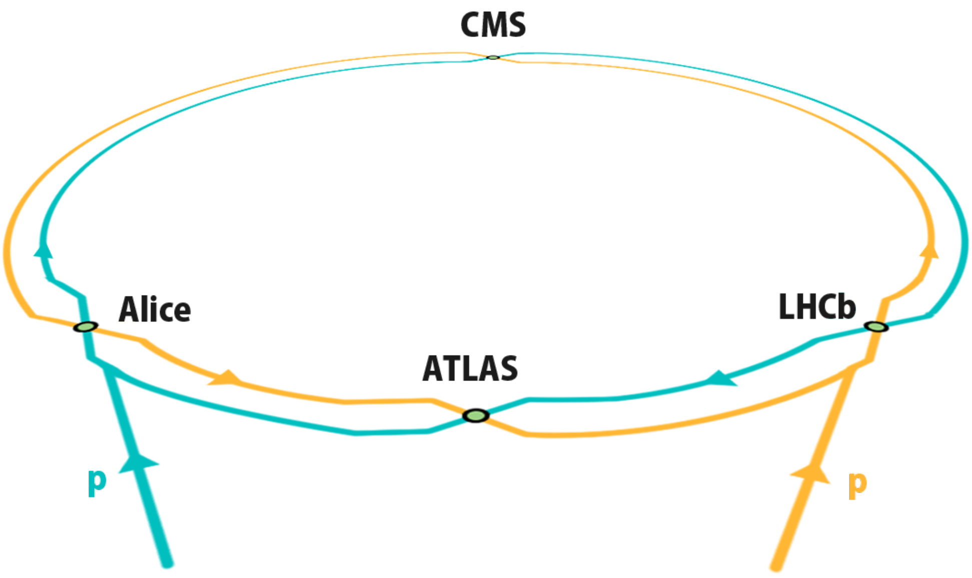


Figure 2.2: An overview of the experiments at the Large Hadron Collider (LHC).

Experiments are located at each collision point. One of them is **ATLAS** [18], a general-purpose experiment with the goal of testing predictions of the Standard Model and discover new particles, for example the Higgs boson and dark matter [19]. Figure 2.3 is a schematic of the ATLAS experiment. ATLAS is the largest detector by volume that has ever been constructed at a particle collider [2].

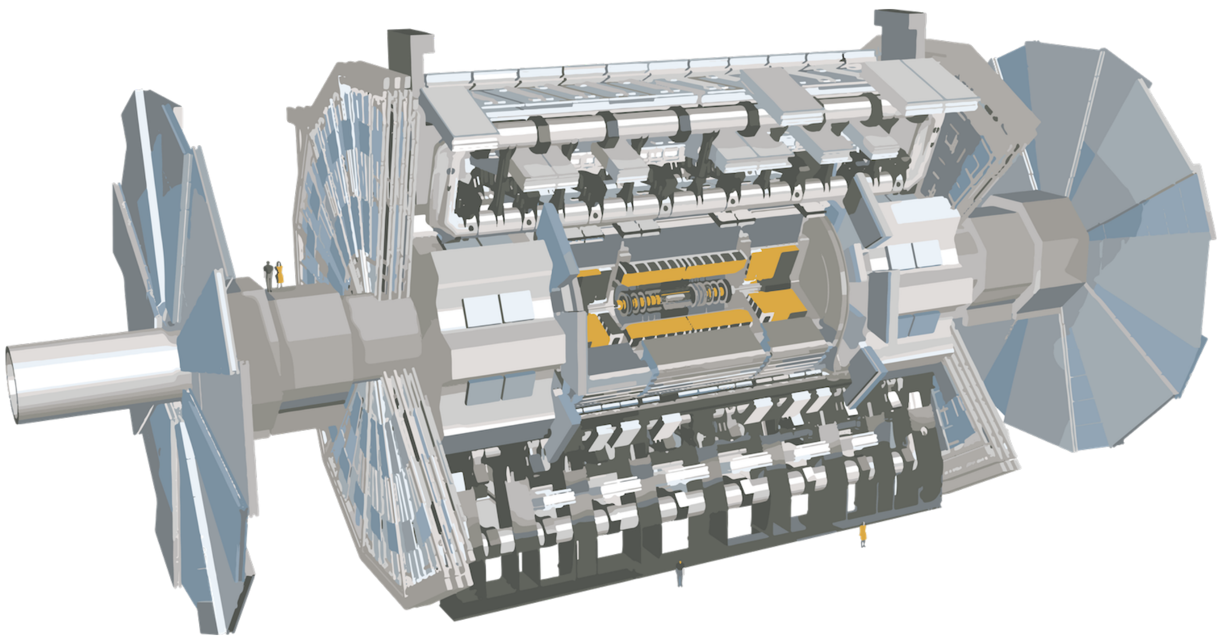


Figure 2.3: A schematic of the ATLAS experiment. The image shows a side-by-side size comparison of the experiment and a human. This image is provided by ATLAS Experiment © 2021 CERN.

2.2.1 The ATLAS Coordinate System

In the ATLAS coordinate system, the x-axis points towards the center of the LHC, the y-axis points upwards and the z-axis is along the beam direction (see Figure 2.4). All axes are orthogonal. The origin of the coordinate system is the collision point [18]. Normally, the spherical coordinate system (radial distance R, polar angle θ, azimuthal angle ϕ) is used instead of the Cartesian coordinate system (x, y, z). Often, the polar angle is replaced by the **pseudorapidity** η, which is defined as [11]:

$$\eta = \ln \left[\tan \left(\frac{\theta}{2} \right) \right] \tag{2.1}$$

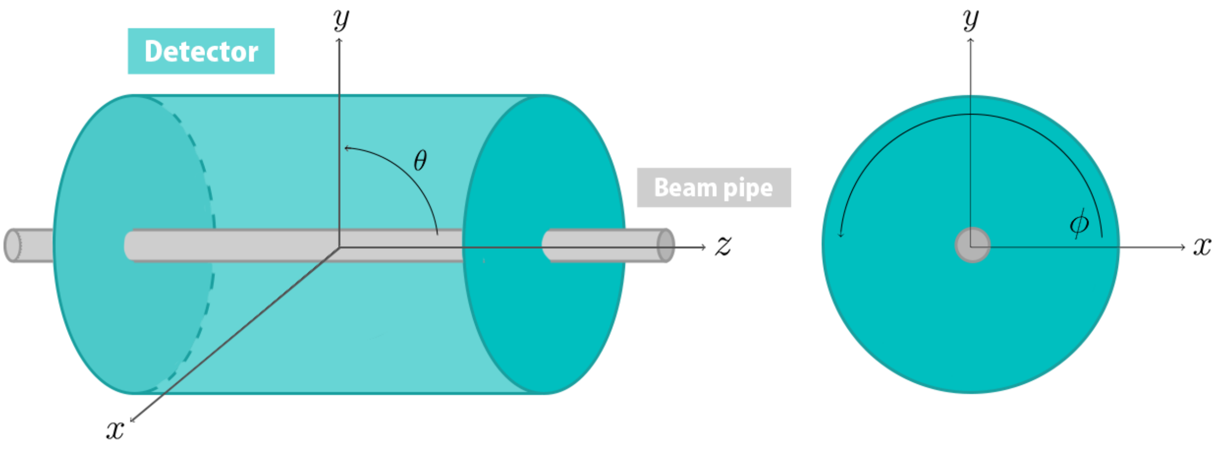


Figure 2.4: The ATLAS coordinate system. The position where a jet deposits energy can be described in Cartesian (x, y, z) or spherical coordinates (R, θ, ϕ).

The xy-plane is referred to as the **transverse plane**. Often in particle physics, the quantity **transverse momentum** p_T is of interest. It is defined as the vector component of the momentum in the transverse plane. The component is perpendicular to the beam axis [18].

2.2.2 Detectors

The ATLAS detector is a multi-layer detector of concentric cylinder layers around the collision point [2]. This design makes it possible to detect particles produced in the collision or produced via subsequent decays. Different particles interact with different layers and they are identified based on where to in the detector they reach and how they leave a mark [11]. Figure 2.5 shows an overview of different particle signatures per layer.

The first layer is the **tracking system**. Here, the charged particles are observed and their momentum is measured. Using computer algorithms we can reconstruct the path they took and thus track them. From the curvature of the tracks and information about the applied magnetic field in the chamber, their momentum can be deduced [10][11].

The second layer is the **electromagnetic calorimeter** which absorbs and detects charged particles through the deposition of energy [10]. The third layer is the **hadronic calorimeter** which is similar to the electromagnetic calorimeter but is able to detect the longer showers from neutral and charged hadrons. The calorimeters are divided into cells in a grid structure. The high granularity allows for a precise detection of where the energy was deposited [20].

The last and fourth layer is the **muon spectrometer** that identifies muons. Even though muons are charged, they are not absorbed by the electromagnetic calorimeter due to their penetration power [10].

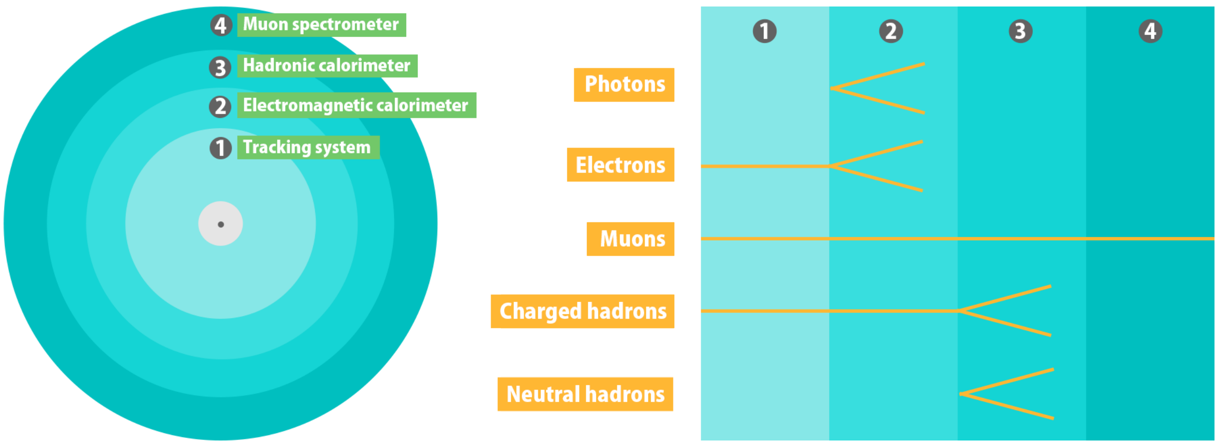


Figure 2.5: Overview of different particle signatures per layer.

2.2.3 Data Selection at the LHC

Each second, about a billion collisions occur inside the detectors at the LHC [3]. This produces a lot of data; up to 60 terabytes (TB) per second [21]. It is impossible to store the entirety of the data produced. To circumvent this, ATLAS uses a trigger system that filters the interesting data, which reduces the amount of data to be stored by several orders of magnitude, from 10^9 to 10^3 [21].

The trigger system keeps the amount of recorded data manageable but it may inadvertently discard interesting events. The challenge is to save enough data to ensure that the rare new physics events are recorded, but still remain within the storage and processing constraints. One idea is therefore to compress the data and use machine learning to do so. Recently, this has been explored with *autoencoders* with promising results [4]. In Section 2.4.3 the discussion about autoencoders will continue.

2.3 Jets and Their Substructure

Apart from particles like electrons and photons, jets (see Section 2.1.1) are also detected at ATLAS. Since the hadronization of quarks produces a collimated stream of particles, the particles hit the detectors near each other and form clusters. These particle clusters are grouped together into one particle-like object – a jet – with a clustering algorithm. The energy assigned to the jet is the sum of the energy of the clustered particles. It is important to note that there is no unique way to cluster jets, and therefore there is a level of arbitrariness to whether a particle is contained within the jet or not. At the LHC, the anti-k_t algorithm [22] is almost always used to reconstruct jets [23]. More jet algorithms can be found in Ref. [23].

Figure 2.6 shows an example of an event with two jets. In cases like this where the jets are back-to-back, it is easy to resolve the two jets. However, there are examples where the two jets are collimated. Depending on the details of the clustering algorithm, these can be identified as one large jet containing two smaller jets. The two smaller jets are called **subjets** of the larger jet [24]. **N-subjettiness** τ_N is a jet property that describes to what degree the jet can be considered to consist of N subjets (due to the arbitrariness of clustering) [25]. The **leading subjet** is the subjet with the largest transverse momentum. The **sub-leading subjet** is the next in order [26].

If a TeV-scale particle produced in a proton-proton collision decays into lighter states, e.g W-bosons ($m_W = 80.379 \pm 0.012$ GeV [27]), then due to the difference in mass the W-boson will have a large transverse momentum [23]. The W-boson is said to be *boosted*. The W-boson can in turn decay in different ways. The most common W-boson decay mode is to hadrons [27], where the hadrons are produced through hadronization of the two quarks in the decay $W \longrightarrow q\bar{q}$. Because of the large boost of the W-boson, the outgoing quarks will not have diverged significantly in relation to each other (small solid angle) and will thus merge into one W-boson jet with two subjets (see Figure 2.7). This type of jet is said to be a **boosted jet** [23].

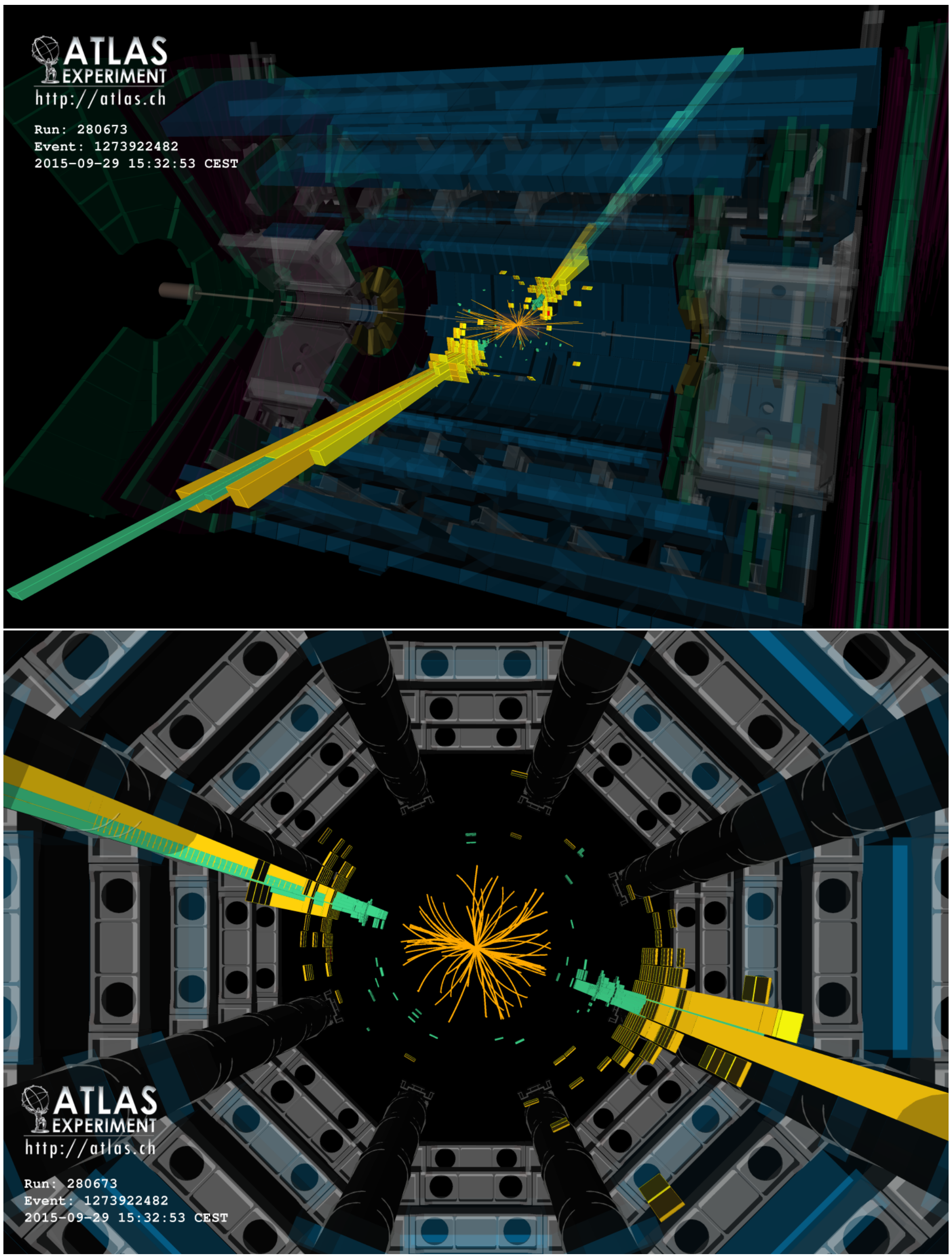


Figure 2.6: A high-energy event with two jets at the ATLAS experiment. This image is provided by ATLAS Experiment © 2015 CERN [28].

Depending on what produces the jets, they will have different internal structures. The W-boson decays into a quark and an antiquark, leading to two jets, as previously mentioned. The energy is shared between these two jets in approximately equal fractions. This leads to two hard cores (high transverse momentum p_T) of the jets measured in the detector. W-jets generally have a two-prong structure. Another class of jets is QCD-jets. Most commonly, they have a single hard core that is surrounded by a diffuse spread of energy due to the radiation of soft gluons. They generally have a one-prong structure [23].

QCD-jets and W-boson jets thus have different substructures and may be distinguished. This is a simplified version of the problem of separating dark jets from standard model QCD-jets. For dark jets there is no guarantee of such a distinct difference, but nonetheless they may still be distinguishable by their substructure.

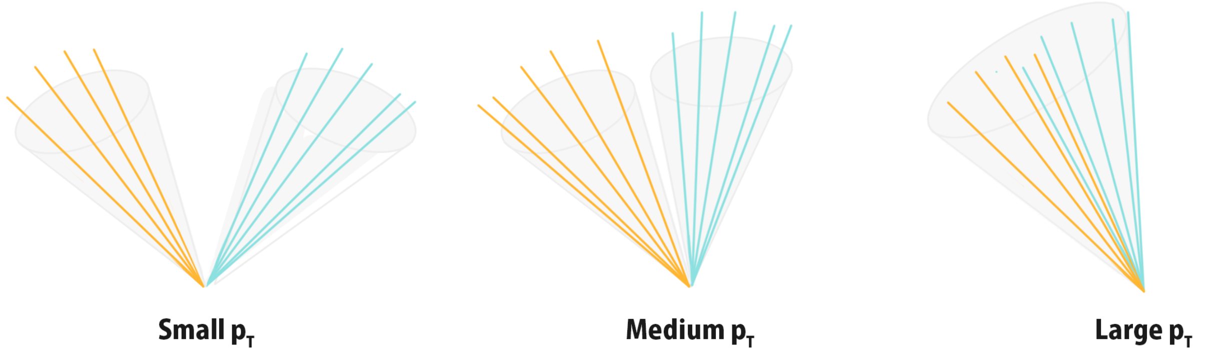


Figure 2.7: A schematic showing how increasing momentum merges two separate jets into one jet with two subjets.

2.3.1 Jet Images

A jet (or any particle) from a proton-proton collision is most commonly represented as a four-vector (see definition in Appendix A.2). In typical Cartesian coordinates and natural units (see Appendix A.2.1) the dynamics of a jet is fully described by the four-momentum (p_x, p_y, p_z, E). However, in practice, using a four-vector on the form (p_T, η, ϕ, m) is more suitable. This is due to the fact that transverse momentum p_T is invariant under Lorentz transformations. The transformation from the Cartesian coordinate four-vector to the new four-vector is:

$$\begin{cases} p_x = p_T \cos(\phi) \\ p_y = p_T \sin(\phi) \\ p_z = p_T \sinh(\eta) \\ |p| = p_T \cosh(\eta) \end{cases} \tag{2.2}$$

In the past few years there have been promising results when using another representation of jets instead of the simple four-momentum, i.e jet images [8][9]. **Jet images** are image representations of jets. A jet image can be seen as the unraveling of the cylindrical calorimeter (see Figure 2.8) in terms of energy deposits. The pixel grid is a pixelation of the calorimeter cell grid and the pixel intensity is set to be proportional to the energy deposited in the cells. The location of the lit up pixels relates to where in the calorimeter the jet left a mark. A typical jet image can be seen in Figure 2.8. Note that instead of using the azimuthal angle ϕ and polar angle θ, the polar angle is replaced with the pseudorapidity η (see definition in Equation 2.1).

The advantage of using jet images over four-vectors is the fact that the images carry information about the jet substructure. Figure 2.9 shows the jet images corresponding to a boosted W-jet and a QCD-jet.

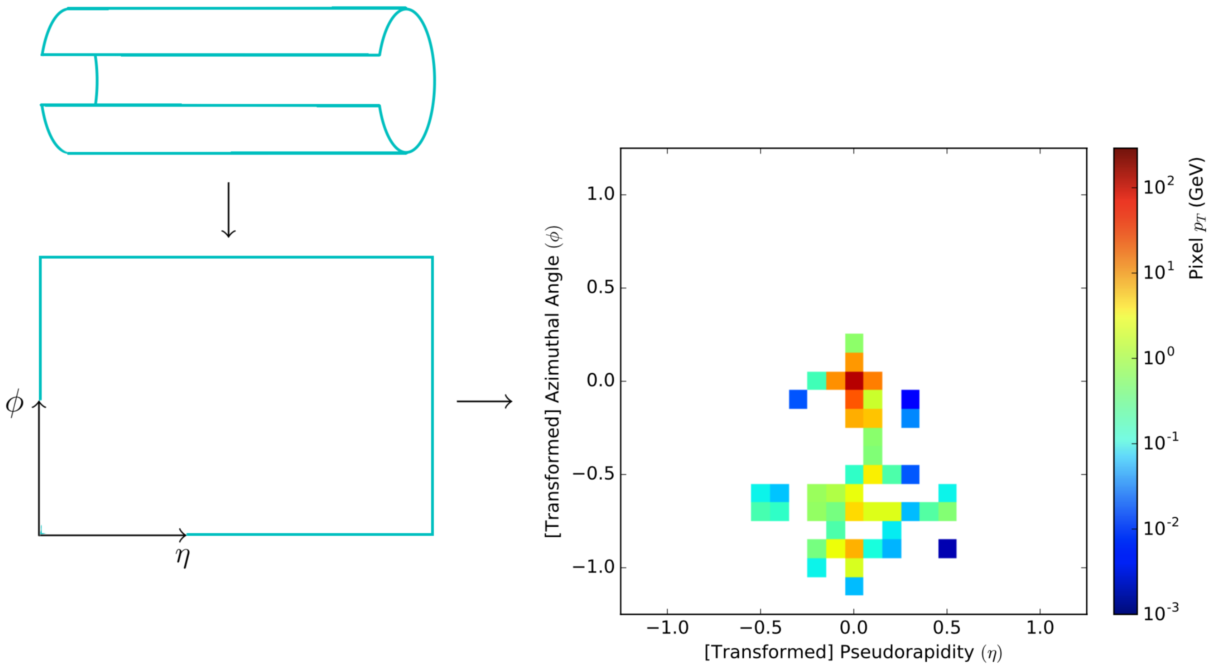


Figure 2.8: A typical jet image and how the unraveling generates it. This jet image is already pre-proccessed according to the procedure described below. The jet image is from Ref. [29]. Used with permission

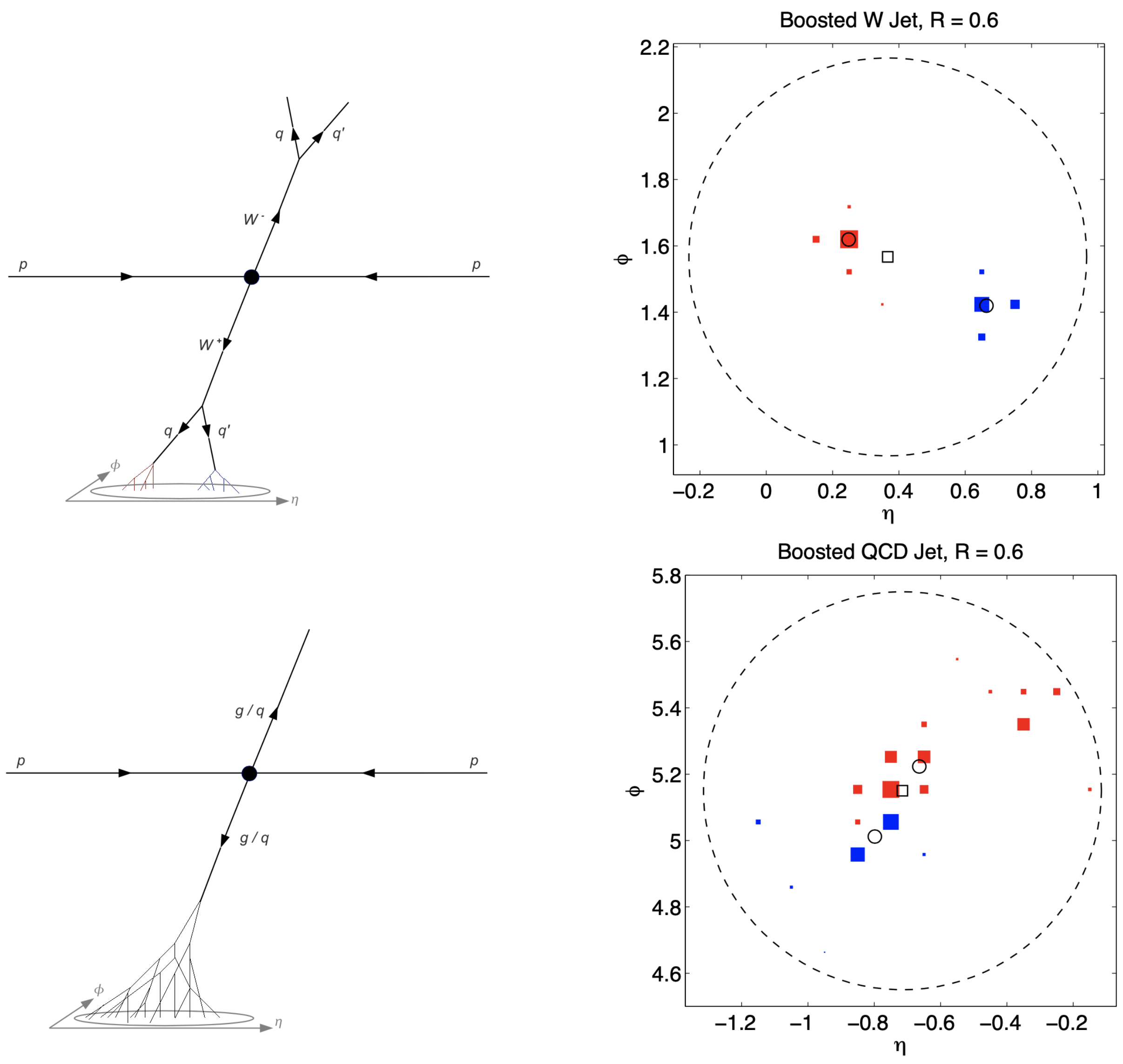


Figure 2.9: A schematic of a boosted W-jet and of a QCD-jet, and the corresponding jet images. The open square shows the jet axis and the open circles the subjet axes of that jet. Note that these jet images are not pre-processed. The image is from Ref. [25]. Used with permission.

Pre-processing

This section will describe the pre-processing procedure from *Jet-Images – Deep Learning Edition* by de Oliveira, et.al. [8] and *Jet-Images: Computer Vision Inspired Techniques for Jet Tagging* by Cogan, et.al [9]. The process is analogous to the pre-processing of images used in facial recognition.

1. **Translation:** In the translation step, the leading subjet is translated to the origin $(\eta, \phi) = (0, 0)$, i.e the middle of the jet image. Note that it is the jet-energy centroid that is translated to coincide with the origin.

2. **Rotation:** Once translated, the jet is rotated around the origin so that the principal axis [30] of the jet is vertical. By rotating every image, the stochasticity of particle decay is removed. If the jet consists of two subjets (e.g as for boosted W-boson jets) then the jet is rotated so that the subleading subjet is directly beneath the leading subjet.

3. **Re-pixelation:** In the re-pixelation step, the pixel intensity is re-distributed among the pixels. This is needed after translation and rotation if the new grid does not line up with the original grid.

4. **Reflection:** In the reflection step, the image is flipped over the ϕ-axis (vertical axis) so that the right side has the highest sum pixel intensity (if this is not already the case).

The pre-processing procedure is essentially the alignment of features so that points of interests are in the same relative pixel location for every image. The analogue of this to facial recognition is the centering of eyes. After the alignment we have ensured that the leading subjet is always in the middle, any subjet directly under and that the remaining highest radiation is always on the right. The purpose of the pre-processing procedure is to help an artificial neural network to train more efficiently [8][9]. Artificial neural networks will be described in Section 2.4.

Pre-processing affects the physical quantities that the image represents. First, consider the translation step. Translations in ϕ are rotations around the z-axis (beam axis). This does not change the pixel intensity as it is a mere rotation of the transverse plane. Translations in η are rotations around the x-axis. This corresponds to Lorentz boosts along the z-axis and as a consequence, such translations do not preserve pixel intensity. A simple solution to this is to redefine the pixel intensity I_i to a Lorentz invariant quantity, e.g from energy to transverse energy, i.e $I_i = E_i \longrightarrow I_i = p_{T,i} = E_i / \cosh(\eta_i)$. In the rotation step, image mass is not preserved and there is no reported solution to this problem in Ref. [8][9]. The image mass is maximally affected when the image is rotated $\pi/2$ [8]. However, the rotation step is reversible. The image could be de-rotated to retrieve the original image mass. The reflection step does not impact the image mass. Note that the initial pixelation of the calorimeter impacts the jet mass resolution. This has a much larger degradation effect on the image mass than the rotation step [8].

Retrieving Mass and Transverse Momentum

The **invariant mass** m of a particle (or a jet) is given by the energy-momentum relation:

$$m^2 = E^2 - |\mathbf{p}|^2 \tag{2.3}$$

where E is the energy and $\mathbf{p}$ is the momentum of the particle. It is a Lorentz invariant quantity. Using the definitions of p_x, p_y and p_z from Equation 2.2, this relation becomes:

$$m^2 = E^2 - (|p_T|^2 + |p_T \sinh \eta|^2) \tag{2.4}$$

By taking a sum over each pixel i, the invariant mass of the jet image becomes:

$$m^2(I) = \left(\sum_i I_i \cos \eta_i \right)^2 - p_T^2(I) - \left(\sum_i I_i \sinh(\eta_i) \right)^2 \tag{2.5}$$

where I_i is the pixel intensity and η_i is the pseudorapidity for pixel i. Note that the pixel intensity for pixel i is defined as $I_i = E_i / \cosh \eta_i$ due to pre-processing. The equation for transverse momentum p_T can be expressed in terms of p_x and p_y, i.e $p_T^2 = p_x^2 + p_y^2$. The equation for the transverse momentum of the jet image becomes:

$$p_T^2(I) = \left(\sum_i I_i \cos(\phi_i) \right)^2 + \left(\sum_i I_i \sin(\phi_i) \right)^2 \tag{2.6}$$

where I_i is the pixel intensity and ϕ_i is the azimuthal angle for pixel i.

2.4 Artificial Neural Networks

Machine learning (ML) is the study of algorithms that seek to make computers *learn from experience.* These algorithms are used in a variety of fields and applications, like biological applications such as tumor detection and DNA sequencing, but also technical applications such as facial recognition, credit scoring and weather forecasts [31]. The main principle for each algorithm is to iteratively build a *model* from a predetermined set of sample data, in order to make decisions and predictions on new data. One subcategory of machine learning that is of particular interest in this thesis is *artificial neural networks.*

Artificial neural networks (ANNs) is a family of networks whose design is inspired by biological neurons in terms of their structure, how they pass signals and how this results in learning. Similarly to how the brain structure is a network of neurons that transfer chemical signals between each other, an ANN is a network of connected nodes that send numerical signals forward [32]. When a neuron is exposed to external stimuli that is strong enough, they are activated and pass the signal along to the connected neurons. For an ANN, the nodes are activated according to a predefined rule that determines their sensitivity. As biological organisms learn, the strength of the connections between neurons change. In an ANN this corresponds to adjusting the weights between nodes [33]. The learning procedure will be explained more thoroughly in Section 2.4.1.

The simplest ANN is the **perceptron**, which is a single artificial neuron that consists of only one node (see Figure 2.10). The node can be viewed as a black box to which one feeds an input $\boldsymbol{x}$, and according to an internal rule it outputs y. The relation between input and output is given by the equation: [32]

$$y(\boldsymbol{x}, \boldsymbol{\omega}) = f(\sum_{k=1}^{K} \omega_k x_k) = f(\boldsymbol{\omega}^T \boldsymbol{x}) \tag{2.7}$$

where f is the activation function and $\boldsymbol{\omega}$ is a vector of weights ω_k [32]. The **weights** determine how much each input signal should affect the output signal [34]. Each input node x_k has a corresponding weight w_k. The weighted sum of incoming signals is the argument to the activation function. The **activation function** determines how strong the input has to be in order to activate. The activation function can essentially be any function but the general rule is that a larger input gives larger output [32]. In this thesis, the activation function used is **Leaky ReLU** (see Figure 2.11) [4].

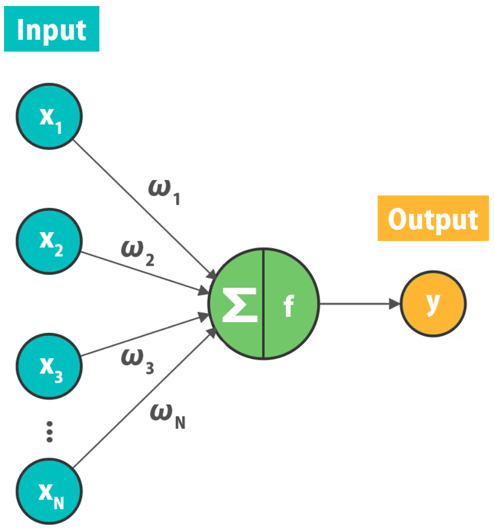


Figure 2.10: The perceptron. The argument of the activation function f is a weighted sum of the inputs. The activation function then gives the output of the perceptron.

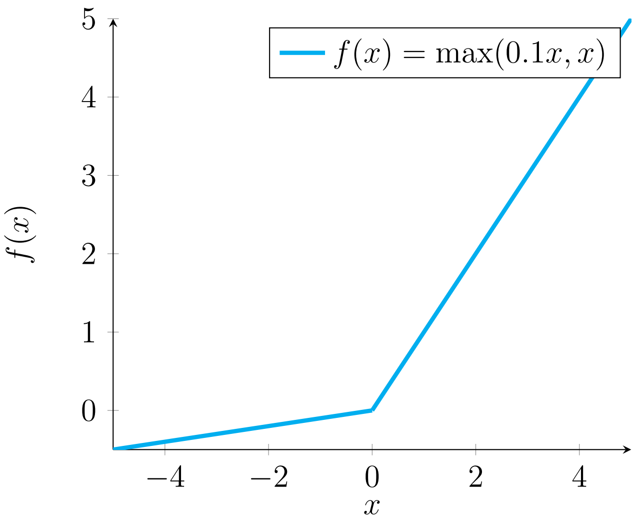


Figure 2.11: The activation function Leaky ReLU.

The perceptron is the basic element of any larger ANN. By connecting several perceptrons in different structures one can build a powerful network that can solve many different tasks. In fact, according to the **universal approximation theorem** [35], any continuous function defined on a compact set can be approximated by a network of perceptrons. Such networks are called **multi-layer perceptrons** (MLP). An MLP is defined as a network of connected perceptrons with at least one layer in-between input and output [32]. An example of an MLP can be seen in Figure 2.12. The first layer is called the **input layer** and the last the **output layer**. The layers in-between are called **hidden layers**. This specific network structure can be expressed as 4-8-6-3 where the numbers indicate how many nodes are in each layer.

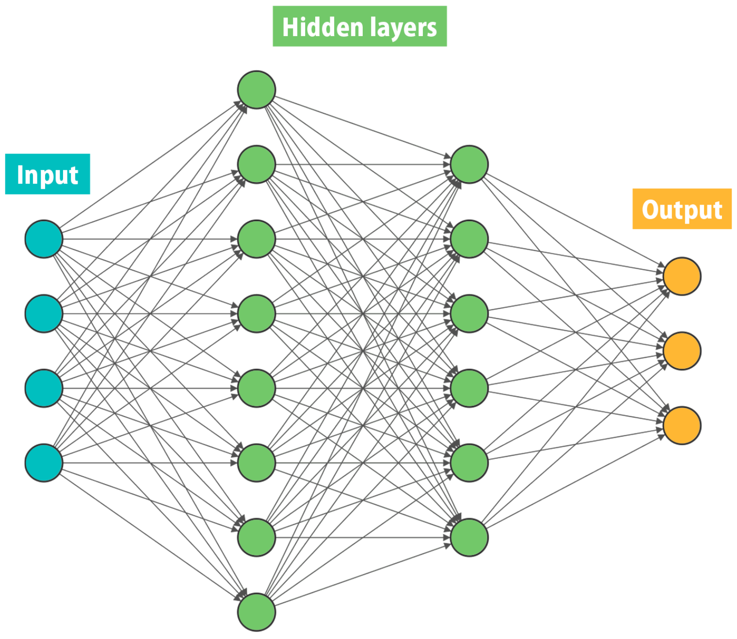


Figure 2.12: An example of an MLP. The first layer is called the *input layer* and the last the *output layer*. The layers in-between are called *hidden layers*. The network structure can be expressed with the code 4-8-6-3.

2.4.1 Training

There are two fundamental categories of learning: *supervised learning* and *unsupervised learning*. **Supervised learning** means that for each input we have a **target**, i.e what we want the output to be. An input-target pair (x_n, d_n) is referred to as a pattern n. When a network learns to perform a task in supervised learning it does so by training on a **data set**, i.e a set of patterns, such that the output eventually equals the target when fully trained. **Unsupervised learning** on the other hand has no targets. Instead, the network learns inherent structures in the data [32]. As an example, consider a data set containing images of dogs and cats. In the case of supervised learning, each image is labeled with either *dog* or *cat*. In the case of unsupervised learning, the same data set is used but there are no labels provided.

Training a network means finding the set of weights that give the smallest error. Depending on what type of task the network is performing, the error is defined differently. The function defining the expression for error is called the **loss function**. In this thesis, the commonly used **mean-squared error (MSE)** loss function $E(\boldsymbol{\omega})$ will be used:

$$E(\boldsymbol{\omega}) = \frac{1}{N}\sum_{n=1}^{N}(y_n - d_n)^2 \tag{2.8}$$

where N is the number of patterns, y_n is the output of the n:th pattern and d_n is the target of that pattern [32].

The basic idea of the training algorithm is to, from a set of initial random weights, iteratively adjust the weights until a loss function minimum is reached. After weight initialization, the training algorithm takes a series of alternating feed-forward and back-propagation steps. In a **feed-forward step**, a pattern from the data set is used as input, which is then fed through the network to yield an output. This is done for each pattern in the data set. After a feed-forward step follows a **back-propagation step** where the gradient of the loss function $\nabla_{\boldsymbol{\omega}} E$ is calculated. Then, according to an optimization algorithm, the weights are updated in relation to that gradient. There is a variety of weight optimization algorithms to choose from. The simplest is called **gradient descent** and it is also the base for most advanced algorithms. In gradient descent, the weights are adjusted by taking a step in weight space in the direction that minimizes loss, i.e $-\nabla_{\boldsymbol{\omega}} E$ (see Figure 2.13). The step size is given by the **learning rate** α. [32] This gives the weight update rule:

$$\Delta\omega_k = -\alpha \frac{\partial E}{\partial \omega_k} \tag{2.9}$$

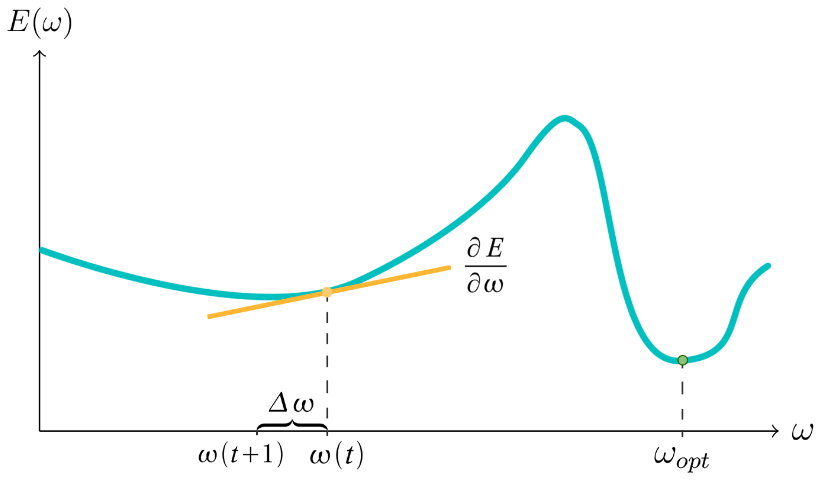


Figure 2.13: Gradient descent with one single weight. The direction of the step in weight space $\Delta\omega$ is given by the derivative of the loss function. The goal is to reach the optimal weight ω_{opt} such that the loss is minimized.

Gradient descent is a rather successful algorithm considering its simplicity, but there are some drawbacks that need to be addressed. For example, if the loss function reaches a plateau then the gradient is small, which makes the steps small and as a consequence it takes a long time to leave the region. This can reduce the convergence speed of the algorithm substantially. Another example is the fact that you can get stuck in a local minimum since it has a zero gradient. This may stop the algorithm prematurely, before we have found the optimal minimum [32]. In this thesis, the optimization algorithm called **Adam** will be used. It is an improvement of gradient descent that for example tackles the two issues mentioned [32]. For an in-depth description of Adam, the reader is advised to delve into the paper *Adam: A Method for Stochastic Optimization* by P. Kingma et. al where it was first introduced [36].

Another drawback with gradient descent is the fact that the entire data set is used to calculate the gradient in each step. This can be a problem when the data set is large since it requires more computing power. A solution to this is to divide the data into mini-batches. The size of the mini-batch is referred to as the **batch size** [4]. When doing this, the network calculates the gradient based on only one mini-batch at a time. This speeds up the training [32]. After one mini-batch gradient descent step is performed we say that the network has completed one **iteration**. When the network has gone through all mini-batches once it has completed an **epoch** [34].

A **hyperparameter** is a parameter that has to be adjusted manually since it is not learned during training [34]. Some hyperparameters have already been introduced, for example learning rate and batch size. In general in machine learning there are a lot of manually set variables – hyperparameters just being one of many – and they are all coupled. Setting these variables is an art and for the lesser experienced it is often a guessing game.

The **1cycle policy** [37][38] is a proposed recipe that removes the task of manually setting a learning rate. In its foundation, the 1cycle policy is a cyclical learning rate (CLR) policy [39] where the learning rate cyclically varies throughout training instead of being fixed. The 1cycle policy consists of only one cycle where the learning rate increases from a small value to a high value, and then decreases back to the small value. After this cycle, it continues to decrease even further for the remaining iterations. Figure 2.14 is shows sketch of how the learning rate changes throughout training in the 1cycle policy.

To use the 1cycle policy, one has to specify the maximum learning rate. The minimum learning rate is set as a fraction of the maximum learning rate. The final learning rate is set to be several orders of magnitude smaller than that. Not only is the 1cycle policy suitable for automatically setting the learning rate, it also greatly speeds up training [38]. The 1cycle policy will be used in this thesis to set the learning rate.

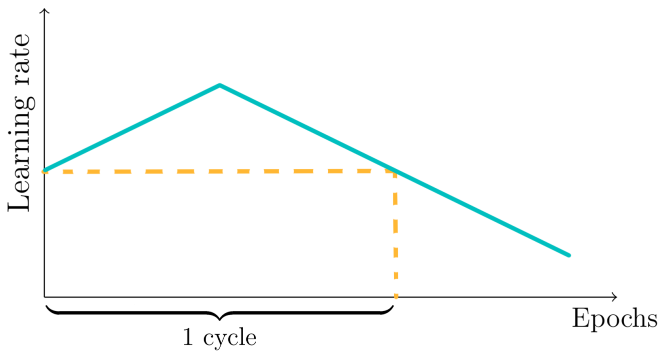


Figure 2.14: A sketch of the learning rate as a function of epochs in the 1cycle policy.

2.4.2 Generalization

Generalization is the ability of a network to perform predictions on previously unseen data. For example, a network that is trained to classify images of dogs is said to have the ability to *generalize* if it can detect dogs in images it has not been exposed to before. A network that has a low generalization performance is not a useful network [33].

To evaluate the generalization performance one usually has two separate data sets: training data and validation data. The **training data** is simply what the network is fed during training. The **validation data** is new, previously unseen data. When this data is fed to the network the generalization performance is given by the loss. If the loss is small then the network generalizes well, and vice versa for a large loss. Note that the need for a validation set does not necessarily mean that you need a completely new data set. A simple way of acquiring validation data is to split the data set into two separate parts and keep one hidden during training. The training data is usually 80 % of the original data set and the validation the remaining 20 % [32].

Overfitting is when the model conforms too closely to the training set, causing the network to deliver poor predictions of new data due to variance in data. Therefore, overfitting causes a low generalization performance [32]. The probability of overfitting increases with the number of nodes and layers in the architecture, and with the number of iterations and epochs during training. A typical sign of overfitting is when the validation loss starts to increase while the training loss continues to decrease (see Figure 2.15). When this occurs the network learns to memorize the data set instead of learning structures. The optimal place to stop training is when both reach a plateau (see Figure 2.15) [4].

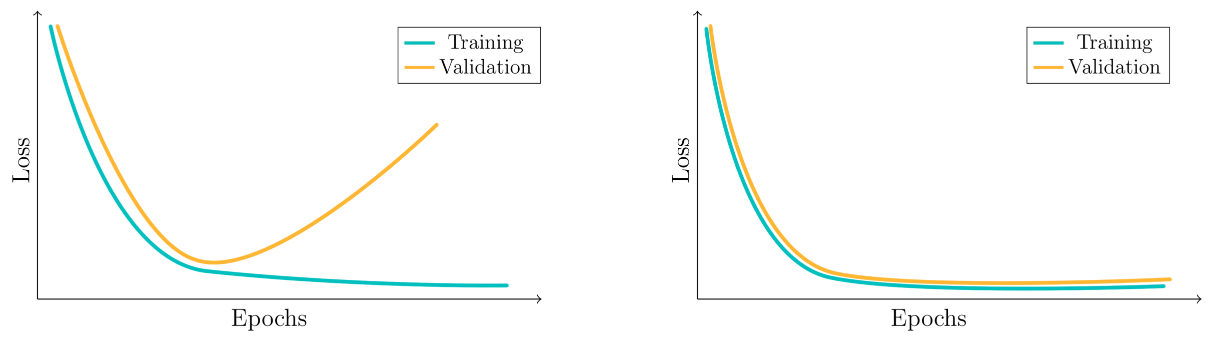


Figure 2.15: Sketch of training and validation loss as a function of training time. To the left we have an example of overfitting and to the right we have optimal performance.

One approach of ensuring generalization is through **regularization**, which is a collection of methods to avoid overfitting. As a general rule of thumb it is better to use a complex architecture with regularization than a simple architecture without. The simplest form of regularization is **early stopping**. As previously mentioned, training with too many iterations and epochs can cause overfitting. Therefore, training is abruptly stopped when the validation loss starts to increase. Regularization can also be done by restricting weights, since there is a correlation between large weights and overfitting [33]. **Weight decay**[1] [40] is a regularizing method where, as training goes on, weights are forced to decay exponentially according to:

$$\omega_{t+1} = (1-\lambda)\omega_t - \alpha \nabla f(\omega_t) \tag{2.10}$$

where ω_t represents the weights at training step t, λ is the weight decay rate, α the learning rate and $f(\omega_t)$ is the gradient of the loss function. Note that λ is a hyperparameter.

2.4.3 Autoencoders

An **autoencoder** is a type of artificial neural network that consists of an *encoder*, a *latent space* and a *decoder* (see Figure 2.16). The encoder encodes the input to the latent space and thus creates a new representation of the input. The decoder then decodes the latent space representation back to the input format and outputs the reconstruction [4][41].

If the latent space has the same dimensions as the input data then the autoencoder is faced with a trivial problem. This means that the network will just learn to copy the data forward. However, if the autoencoder has a latent space bottleneck, then the autoencoder is forced to learn a reduced representation of the data. The encoder may thus be effectively

[1]Not to be confused with L_2-regularization which is often referred to as weight decay. For some optimization algorithms they are equivalent but for Adam they are not [40].

used for compression and the decoder for de-compression. The compressed representation of the input data is the latent space representation [33].

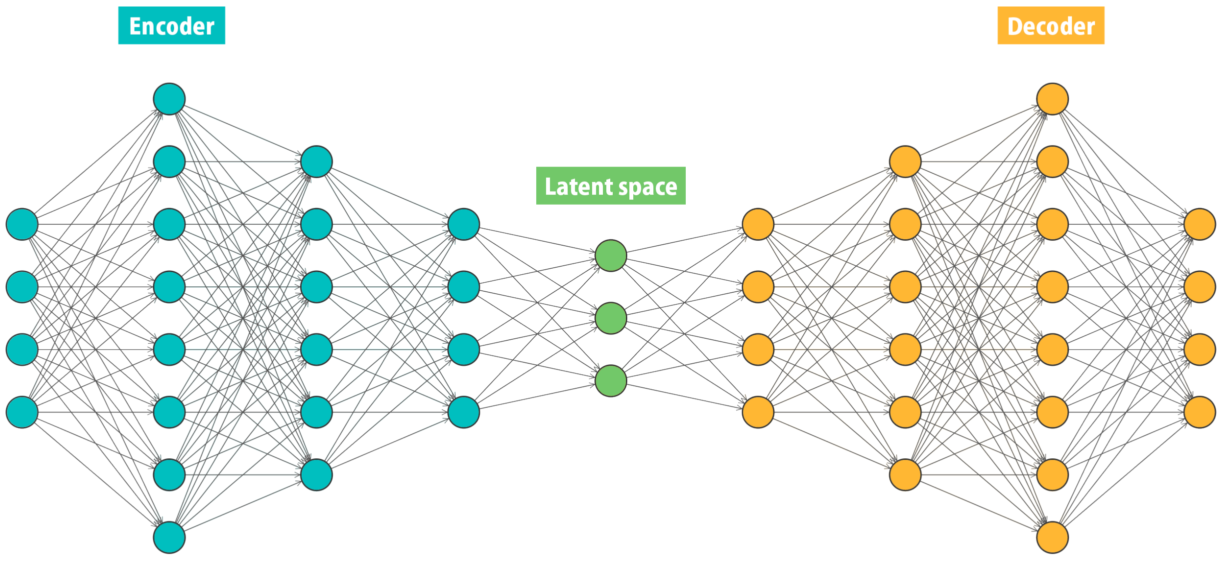


Figure 2.16: An example of an autoencoder network structure and its components.

The architecture of an autoencoder network can vary greatly. The minimum requirement is that the input and output layer have the same dimensions. A common architecture is the *butterfly design*. This entails a network with an odd number of layers and a symmetry plane along the bottleneck (see Figure 2.16) [32]. Deep autoencoders, i.e autoencoders with many hidden layers, is preferable as it has been experimentally shown that they yield better compression [42].

Autoencoders perform unsupervised learning as they do not need labeled data to function. However, since they create their own targets and labels, they can be seen as being self-supervised. If the data inherently contains mostly linear correlations, then a linear activation function is good enough. However, for more complex problems the autoencoder needs a non-linear activation function, e.g Leaky ReLU (see Figure 2.11) [32].

Training an autoencoder is done precisely in the same manner as for an MLP. The loss function used to train an autoencoder is:

$$E(\boldsymbol{\omega}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}(y_k(\boldsymbol{x}_n, \boldsymbol{\omega}) - x_{nk})^2 \tag{2.11}$$

where $\boldsymbol{\omega}$ are the weights, N is the number of data points in the data set, K the number of inputs, y the output and x the input [32]. Note that this is the same loss function as MSE-loss (see Equation 2.8) but the targets are equal to the inputs.

2.4.4 Anomaly Detection

An **anomaly** is a data point – or a group of data points – that differs from the majority enough to suspect that it was generated from another mechanism [5]. Detection of such anomalous data is useful in many applications; ranging from credit-card fraud, where the anomaly could be abnormally large sums or many transactions, to medical diagnoses, where an anomaly could be a sign of a disease. In order to detect an anomaly, one has to first model what *normal* implies. Everything that does not conform to this model is considered anomalous [43].

Anomaly detection, i.e identifying anomalies in data, can be done with an autoencoder, with the assumption that what is classed as normal has a similar underlying structure. When training on normal data, the autoencoder will learn to compress and decompress, eventually with a small reconstruction loss. However, if the network later is fed anomalous data it would give a significantly larger reconstruction loss since the structure is different. The reconstruction loss can be considered an *anomaly score*. This can be converted to a probability, i.e with what certainty is this data point anomalous. A more complex continuation is to combine the autoencoder with a network used for binary classification. This extension will classify each datapoint in the data set as either normal or anomalous [5]. One can also just set a threshold value for the classification based on the anomaly score.

A common metric for evaluating the classification accuracy is the **confusion matrix**. In the case of supervised learning, each data point has a predicted class and a true class. The **true class** is the correct classification and the **predicted class** is how the network de facto classified the data. The confusion matrix is a table displaying the classification performance in terms of true and predicted classes.

In the case of binary classification (class 1 = positive, class 2 = negative), the confusion matrix is a 2×2 matrix. The matrix elements are: true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) (see Figure 2.17). **True positives** are the data points that were correctly predicted to be positive. Conversely, **true negatives** are the correctly predicted negatives. **False positives** are falsely predicted positives. **False negative** are falsely predicted negatives.

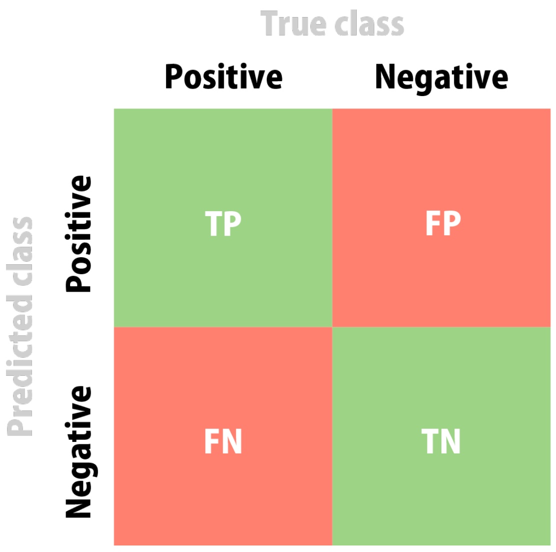


Figure 2.17: A confusion matrix of a binary classification problem. It is a matrix of four elements: true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN).

In the case of anomaly detection, a network is tasked to recognize anomalous data among normal data. Suppose the data set contains 10 normal data points and 12 anomalous and after feeding the data through the trained network, it predicts 8 data points as anomalous. Of the 8 data points recognized as anomalous, 5 actually are anomalous (true positives), while the other 3 are in fact normal data points (false positives). In total, 7 anomalous data points were not found (false negatives), and 7 normal data points were correctly predicted (true negatives).

Chapter 3

Tools

A large portion of this thesis project is to continue the development of a Python package for compression of high energy physics data (especially jets) with autoencoders. The package in question is open-source and available on GitHub. The main developers have been Eric Wulff, Erik Wallin and Honey Gupta which have developed the package in relation to their thesis work and project work [4][6][7].

The package relies heavily on the PyTorch and fast.ai [44][45] Python libraries. For a list of the most important libraries used, see Table 3.1. After Gupta's work, the developers of fast.ai did a from-scratch rewrite of the library. As a consequence, the entire package had to be migrated to the new version. Migrating the essential code was the first task of this thesis project.

Table 3.1: Summary of most important libraries used.

Library	**Version number**
fast.ai	2.1.2
PyTorch	1.7.0
Pandas	1.1.4
matplotlib	3.3.2
NumPy	1.19.3
Jupyter Notebooks	1.0.0

To enable future developers to build upon this work and use the code, each developer has to provide documentation to their code. Most of the scripts in the package are written in Jupyter Notebooks [46]. This allows for the scripts to be divided into cells of code where each cell can be run separately. It is possible to write headings and text to describe the different parts. With a well written documentation, the steep learning curve of having to interpret someone else's code is removed. It makes research accessible and allows for larger collaborations between many different parties.

Previously, the package had only been developed for compression of HEP data. With this thesis work, the functionality of working with compression of jet images was added together with anomaly detection (as well as documentation). The code developed for this project is divided into four notebooks:

- PROCESS_JET_IMAGES: Contains the pre-processing of the images. The .HDF5-file is unpacked into a PANDAS dataframe. This dataframe is then packed into different .PKL files. Among the .PKL-files there are e.g the validation and training set. The notebook also contains visualization of the jet images and distributions, and an optional truncation step.

- TRAIN_JET_IMAGES: Contains the code for training of networks and the definitions of all network models and hyperparameters. The predictions from the model and the model itself is saved to .PKL-files and a .PTH-file respectively. The option of loading a previously trained network model exists. All trained models in this thesis can be found on GitHub page for the package. The notebook ends with calculations of the MSE losses.

- ANOMALY_JET_IMAGES: Contains the anomaly detection related code. The predictions from the training notebook are loaded and the optimal threshold is found. Then the confusion matrix is calculated.

- ANALYSIS_JET_IMAGES: Includes the code for all plots in this thesis, e.g calculations of m and p_T from the jet images and plots of their distributions.

These notebooks will also be published on Zenodo.

Chapter 4

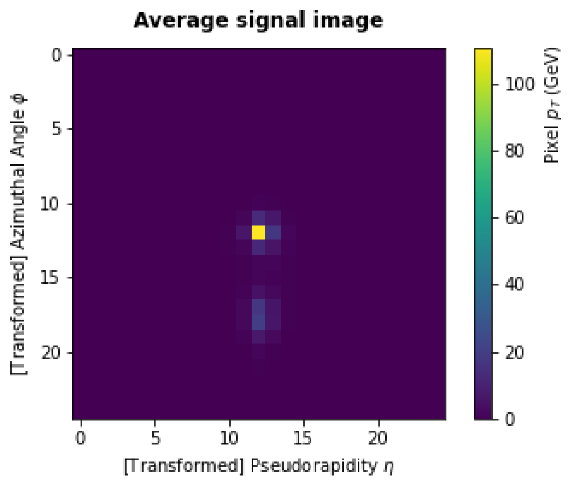
Method

In this chapter the methodology of testing the autoencoder network with jet images will be described. The first section describes the data set used. The networks are constructed with similar hyperparameters but with different architectures. The common hyperparameters will be described in the second section and the different architectures in the third section. The third section also includes the general procedure for each network setup.

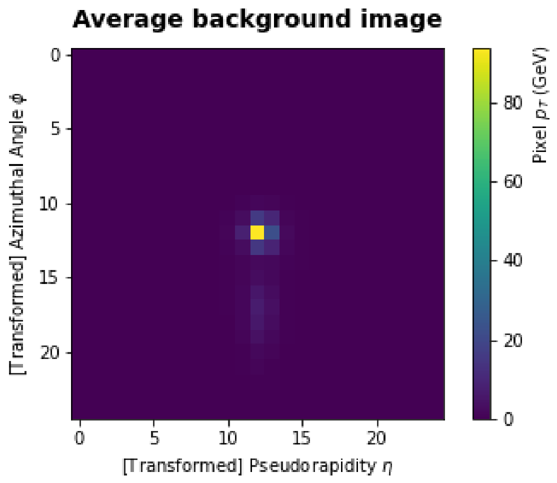
4.1 The Data

The data set used in this thesis is an artificially generated set of jet images. They were generated using a *Location Aware Generative Adversarial Network* (LAGAN) that learnt to mimic patterns from simulated high energy collisions and generate new similar data [29]. An extensive description of the generation can be found in *Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis* by de Oliveira, et al [29].

The data set contains 872 666 jet images of size 25×25 pixels (a pixel grid with $\Delta\eta\times\Delta\phi = 0.1\times0.1$ where $\eta\times\phi\in[-1.25,1.25]\times[-1.25,1.25]$). Half of the images are jets from high energy W-bosons (signal) and half are from quark- and gluon-initiated (QCD) jets (background). In Figure 4.1 the average signal and background image is shown. Each jet image also has a corresponding array of ten variables, calculated from the original jet. Here, mainly the first four will be considered: coordinates η and ϕ, mass m and transverse momentum p_T (see Figure 4.2). The other variables are related to jet substructure. These four variables will hereafter be referred to as the **original variables**. The distributions of these variables for all jet images will be referred to as the **original distributions**.

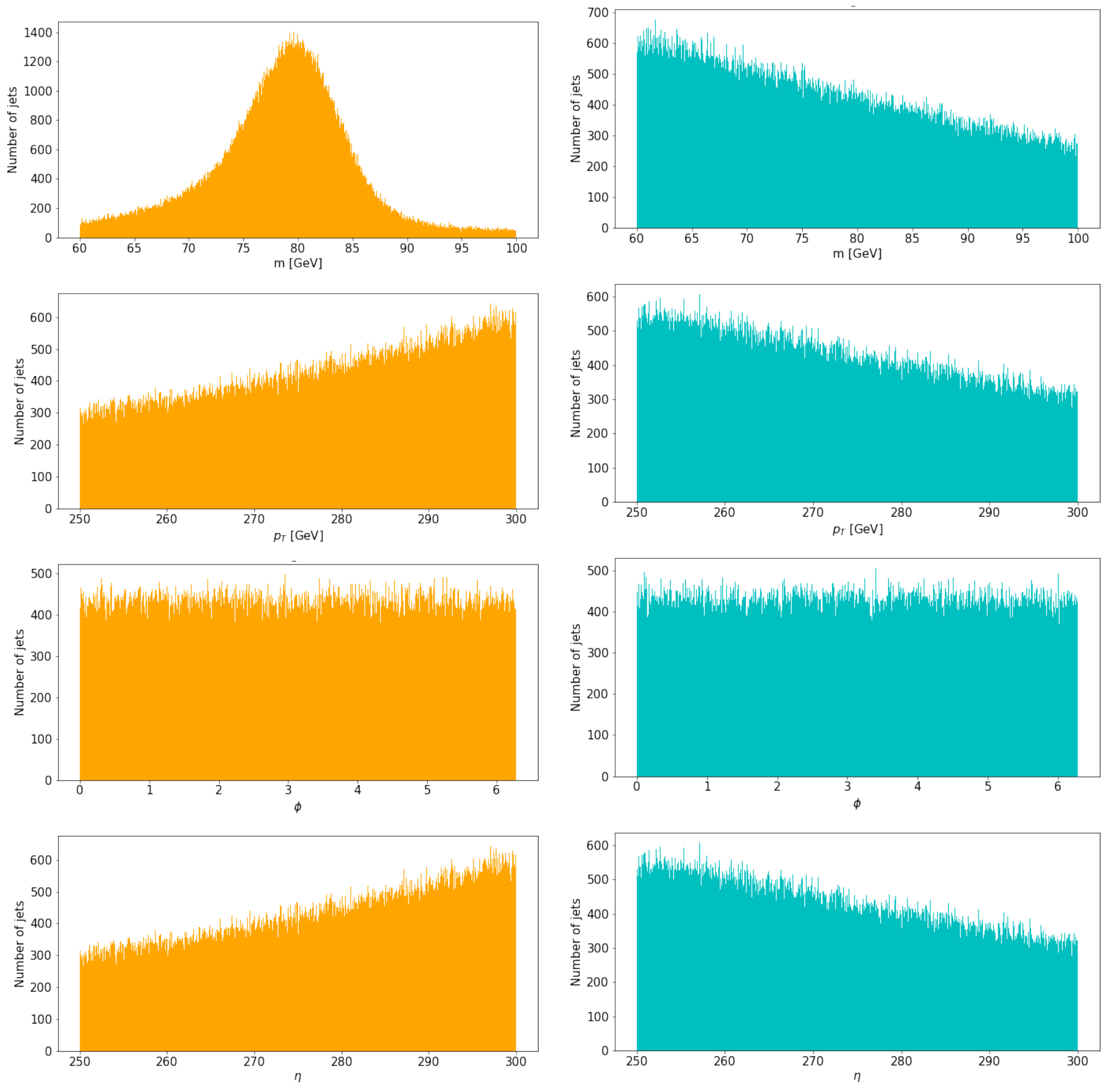


(a) Average signal jet image



(b) Average background jet image

Figure 4.1: The average signal (a) and background (b) jet images in the data set.



(a) Signal distributions (b) Background distributions

Figure 4.2: Signal and background distributions for mass m, transverse momentum p_T, azimuthal angle ϕ and pseudorapidity η. These distributions are *original distributions*, meaning that they are calculated from the original jet.

The images are already pre-processed according to the procedure described in Section 2.3.1. The pixel intensity is thereby given by the transverse momentum p_T of the cell. The mass and transverse momentum for each jet is in the range 60 GeV $< m_{jet} <$ 100 GeV and 250 GeV $< p_{T,jet} <$ 300 GeV, respectively. Jet clustering was done with the anti-k_t algorithm with a radius of $R = 1$ and then re-clustered with $R = 0.3$ k_t into subjets. The images are sparse with about 10 % non-zero elements.

Table 4.1: Summary of the jet image data set.

Number of images	872 666
Signal/background ratio	50/50
Image size	25×25 pixels
Jet variables	η, ϕ, m, p_T
Pre-processed	Yes

The data set is divided into two subsets: training data (80%) and validation data (20 %). Note that for each network, only the jet images are used as input. The corresponding jet variables are used for comparison purposes only. Using both the images and variables on the same footing was tested but (as expected, due to the difference in pixels and jet variables) it gave poor results. Since the network is only able to take a vector as input, the images are reshaped from a 25×25 matrix into 625×1 vector.

4.2 Network Construction

The networks are constructed with a common set of hyperparameters but with different architectures (number of layers and number of nodes per layer). The hyperparameters are selected from Ref. [4].

Even though the architectures are different, all setups have the butterfly design. Due to complex correlations in the data, the activation function used is Leaky ReLU (see Figure 2.11) and the loss function used is the MSE loss function (see Equation 2.8). The training is done by employing the optimization algorithm Adam where the learning rate is set according to the 1cycle policy (see Section 2.4.1). In the implementation of the 1cycle policy in FAST.AI, the maximum learning rate α_{max} is a user-set parameter. Here, the maximum learning rate is set to the default value used in FAST.AI, i.e 10^{-3}. The minimum and final learning rate are set to $1/25$ and $1/10^5$ of α_{max}. Both training and validation data are divided into batches of 256 jet images. The training is done for 30 epochs. Empirically it was shown that this was sufficient to achieve convergence for each network setup. A summary of the static network hyperparameters can be seen in Table 4.2.

Table 4.2: Summary of the static network hyperparameters.

Activation function	Leaky ReLu
Loss function	MSE
Training algorithm	Adam
Learning rate	1cycle policy
Weight decay	1e-6
Batch size	256
Number of epochs	30
Training/Validation ratio	80/20

The training was done on a MacBook Pro (2016) with a 2.9 GHz Dual-Core Intel Core processor, 8 GB 2133 MHz LPDDR3 memory and Intel Iris Graphics 550 1536 MB integrated graphics unit. No GPUs were utilized for training.

4.3 Approach

The main idea of this work is to feed jet images from the data set to networks with different architectures. We will start with the simplest setup and then increase the network complexity step by step. An overview of the different setups used can be seen in Table 4.3.

The first setup is a trivial network. Since there is no bottleneck, no compression or decompression will be done. This network should output a copy of the input. Then, we move on to networks with bottlenecks. The second setup will use a network with a large bottleneck and the third and last with a smaller bottle neck. The actual bottleneck sizes are chosen arbitrarily, leaving a more systematic study for future work.

Table 4.3: Summary of setups. The numbers indicate how many nodes are in each layer. The bottleneck size is given by the middle layer.

	Type	**Network architecture**
Setup 1	No bottleneck	625-625-625-625-625
Setup 2	Large bottleneck	625-300-200-300-625
Setup 3	Smaller bottleneck	625-300-100-300-625

The analysis starts with testing the pre-processing degradation of the original jet variable distributions. This is done by calculating the corresponding distributions from the jet images with Equations 2.5 and 2.6, and comparing them to the original distributions.

Then, for each setup, the ability of the network to compress jet images is evaluated. This is done by calculating the validation loss. In addition, the distributions of mass and transverse momentum from inputted images versus outputted images are compared. These distributions are also calculated with Equations 2.5 and 2.6.

For the non-trivial setups, the network will be fed background data during training and then fed with a mix of background and signal to perform anomaly detection. The anomaly detection will be done by thresholding the reconstruction loss. If the network is able to distinguish the signal (anomaly) from the background (normal) then the next step is to fine tune the thresholding to obtain the optimal discrimination by comparing confusion matrices.

Chapter 5

Results & Discussion

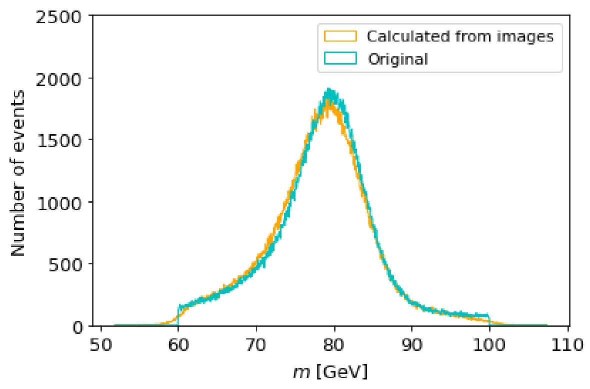
The first part of this chapter shows the results of the pre-processing degradation of the original jet variable distributions, as it is a prerequisite to understand some of the features in the following distributions. Thereafter, a section about compression results for each setup follows. Lastly, the results from the anomaly detection are discussed. In the following, the QCD-jet images will be referred to as *background*, and the W-boson jet images as *signal*.

5.1 Pre-processing Degradation

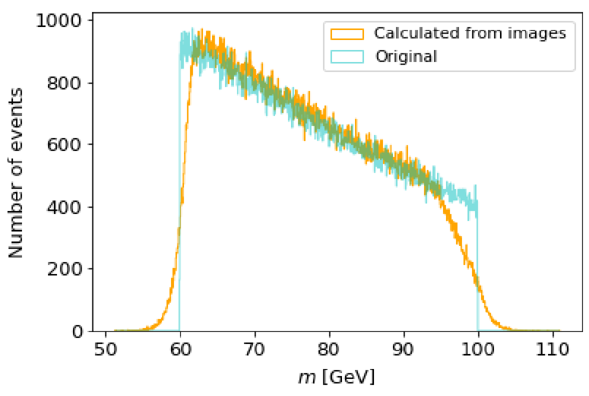
Figure 5.1 shows the mass m and transverse momentum p_T distributions from the jet images (calculated with Equations 2.5 and 2.6) and from the original jet variables, for signal and background respectively. Note that these distributions have not been processed by a network yet.

The distributions have approximately the same shape. Most importantly, the calculated signal mass distribution follows the original normal distribution with a peak around 80 GeV, which corresponds to the W-boson mass ($m_W = 80.379 \pm 0.012$ GeV [27]). This is expected as all signal jets are coming from W-boson decays, and the invariant mass of the two subjets (and therefore of the jet that encompasses them) corresponds to the mass of the W-boson. The most apparent difference in the distributions is the distortion at the edges. The reason for this can be traced back to the pre-processing. As mentioned in Section 2.3.1, the pixelation has the largest effect on the distributions (see Figure 5.2). The pixelation reduces the resolution of the image, which is correlated to the resolution of the quantities in the distribution. This explains the blurring of the sharp edges in the distribution tails, as well as the blurring of the signal peak. The resolution effect is visually smaller for the signal mass distribution compared to the other distributions. However, this is simply a consequence of the cropping of the original distributions to a certain m, p_T range. Three of the original distributions are skewed distributions and cropping them results in large, sharp edges. Cropping the tails of a unimodal normal distribution gives small edges.

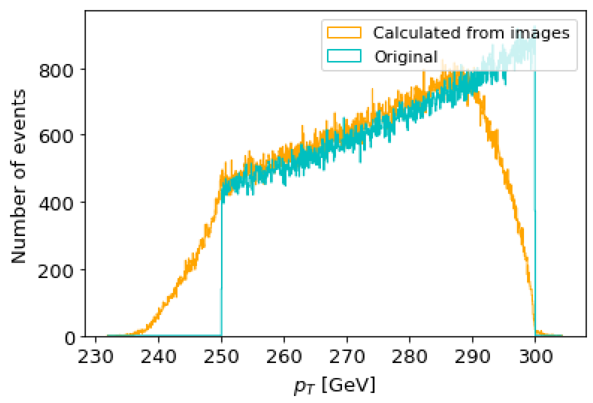
Because the calculated distributions and original distributions differ even without being affected by a network, the network cannot be expected to give a reconstruction that reproduces the original distributions. However, the network setups should be able to reproduce the calculated distributions.



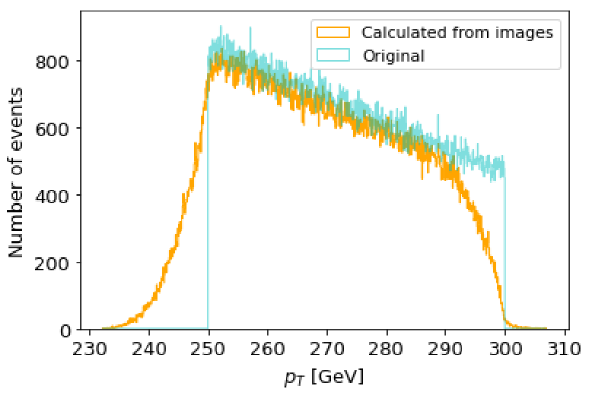
(a) Mass distribution for signal



(b) Mass distribution for background



(c) Transverse momentum distribution for signal



(d) Transverse momentum distribution for background

Figure 5.1: Comparison of mass m and transverse momentum p_T distributions calculated from the jet images and the original distribution, for both signal and background.

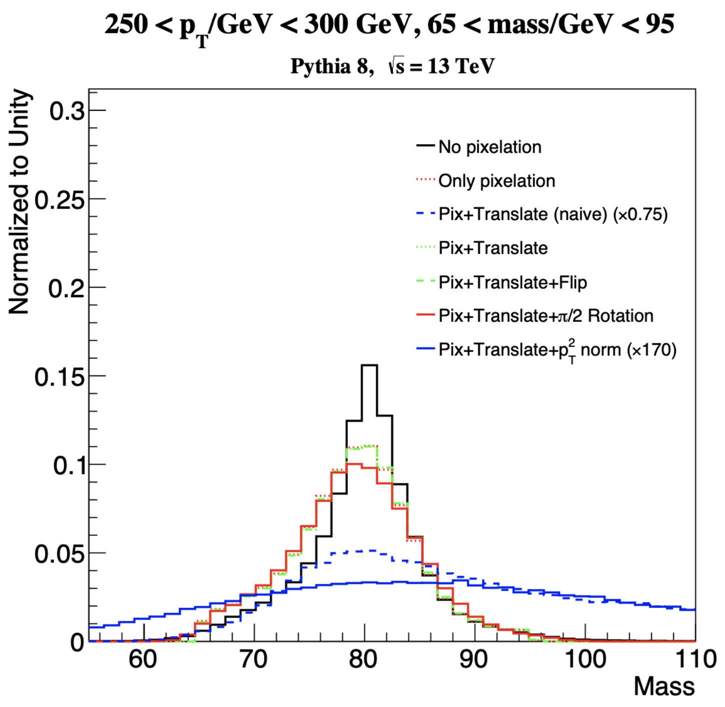


Figure 5.2: Degradation of the signal mass distribution with respect to each pre-processing step. The initial pixelation (*Only pixelation*, red dotted) highly impacts the original data (*No pixelation*, black). Translation and reflection (*Pix+Translate+Flip*, green dotted) does not affect the distribution but rotation has a slight impact (*Pix+Translate+$\pi/2$ Rotation*, red). The image is from Ref. [8]. Used with permission.

5.2 Compression

5.2.1 Setup 1: Network Without Bottleneck

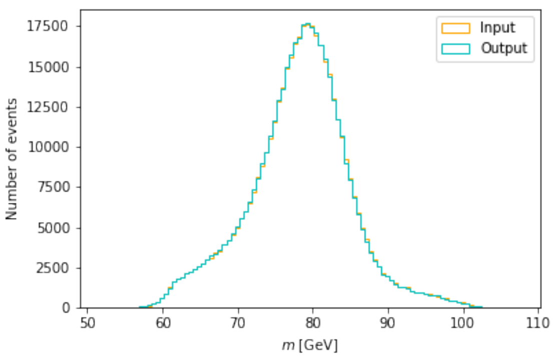
The first setup, called Setup 1, has a network architecture without a bottleneck. For this reason, the reconstruction loss is expected to be negligible. The network should be able to learn to copy the data forward since it is faced to solve a trivial problem. This setup sets a baseline for the best possible outcome of the bottlenecked networks. Figure 5.3 shows the distributions for mass and transverse momentum for both signal and background after passing it through the background-trained network, and Figure 5.4 shows the corresponding residuals.

The residual is given by the difference between input and output, and then divided by input. A perfect reconstruction would have the shape of a Dirac delta function. From the standard deviation of the residual, the reconstruction error[1] can be obtained. Two

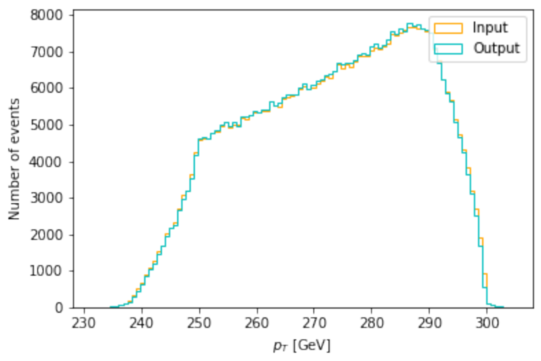
[1]Not to be confused with reconstruction loss. The reconstruction loss is a metric used for the autoencoder training and reconstruction error is related to the width of the residuals.

standard deviations represent the reconstruction error for over 95 % of the data, and this metric is used to compare each setup.

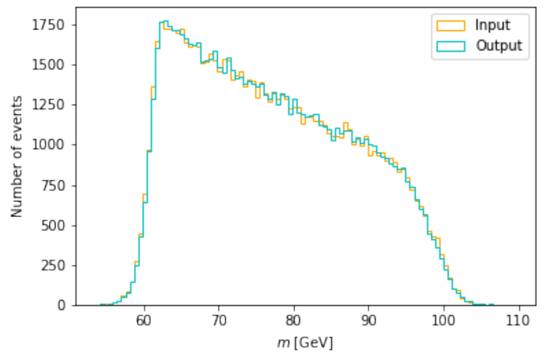
The MSE validation loss is 6.6×10^{-4}. This resulted in over 95 % of the signal images having a reconstruction error within 0.84 % for both mass and transverse momentum, and within 1.7 % respectively for background. Transverse momentum is reconstructed better than mass. The residuals for transverse momentum are within 0.3 % for both signal and background. Each residual peak is narrow has an expectation value with a negligible distance from $\mu = 0$, as expected. The reason for the error not being even smaller is because the hyperparameters are not optimized.



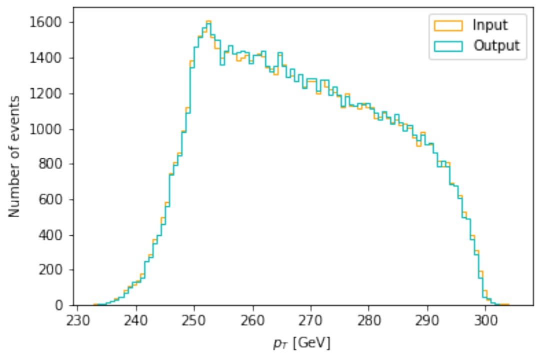
(a) Mass for signal



(b) Transverse momentum for signal

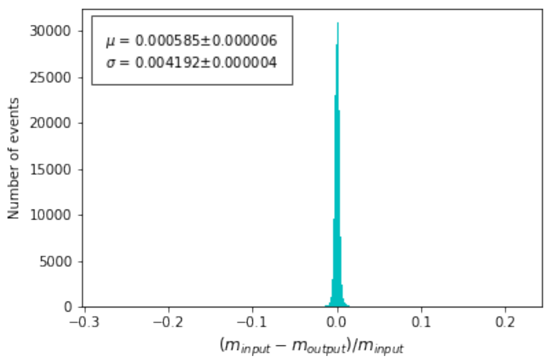


(c) Mass for background

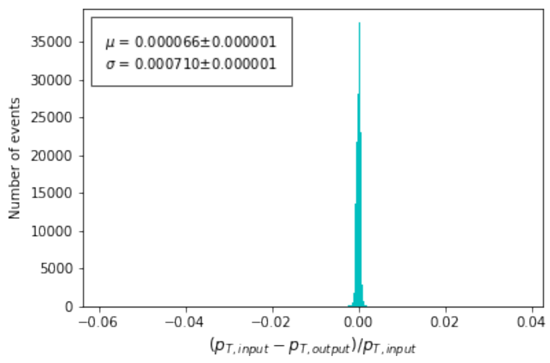


(d) Transverse momentum for background

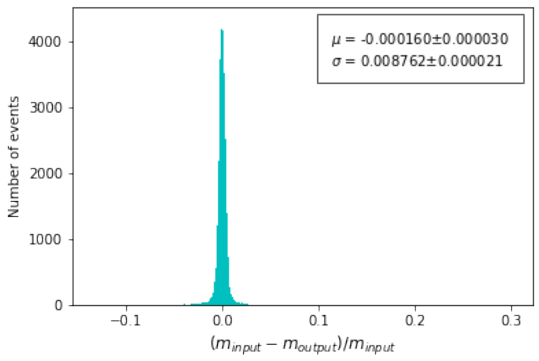
Figure 5.3: Mass m and transverse momentum p_T distribution overlay comparison of input and output of the autoencoder for both signal and background for setup 1.



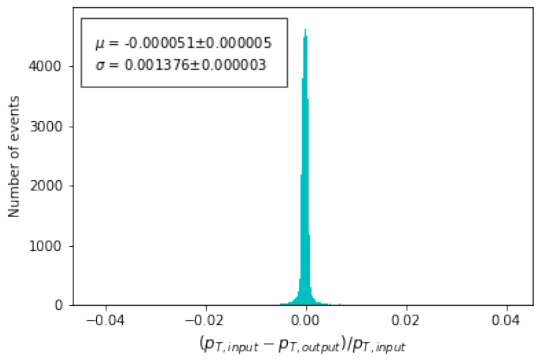
(a) Mass residual for signal



(b) Transverse momentum residual for signal



(c) Mass residual for background



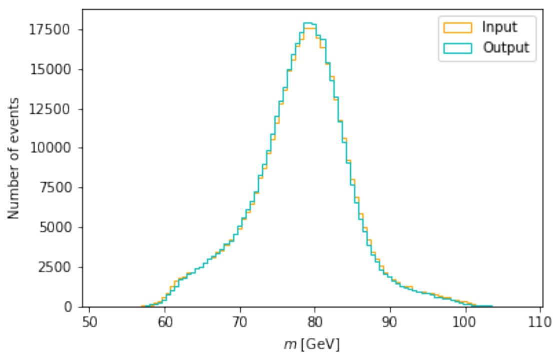
(d) Transverse momentum residual for background

Figure 5.4: Mass m and transverse momentum p_T residuals of input and output of the autoencoder for both signal and background for setup 1.

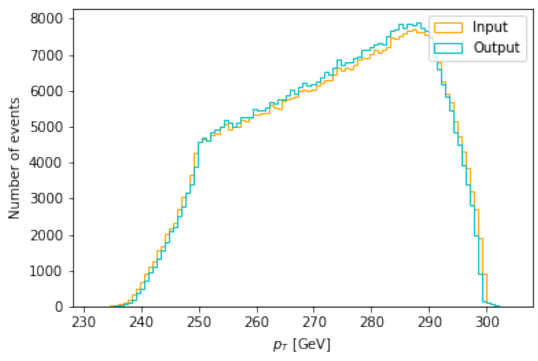
5.2.2 Setup 2: Network With Large Bottleneck

This setup does not solve a trivial problem. Instead, it is faced with the task of compressing the 625 input variables to 200 variables, i.e a compression to about a third of the original size. Figure 5.5 shows the distributions for mass and transverse momentum for both signal and background after passing it through the background-trained network, and Figure 5.6 shows the corresponding residuals.

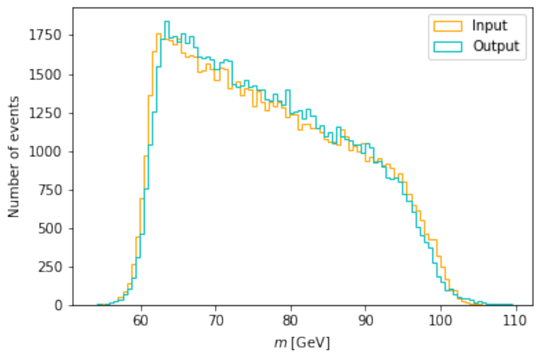
The MSE validation loss is 7.6×10^{-3}. This resulted in over 95 % of the signal images to having a reconstruction error within 3 % for both mass and transverse momentum, and within 5 % respectively for background. Each residual peak has an expectation value with a negligible distance from $\mu = 0$. This network gives a reconstruction loss that is a about one magnitude larger compared to the trivial network in setup 1.



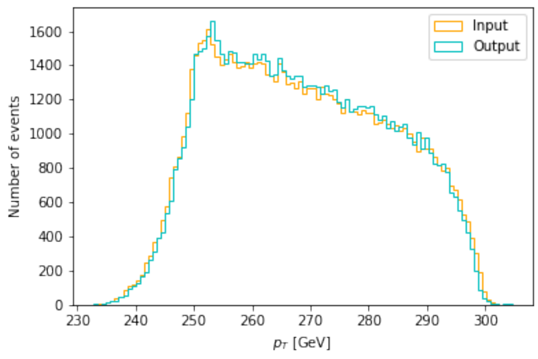
(a) Mass for signal



(b) Transverse momentum for signal

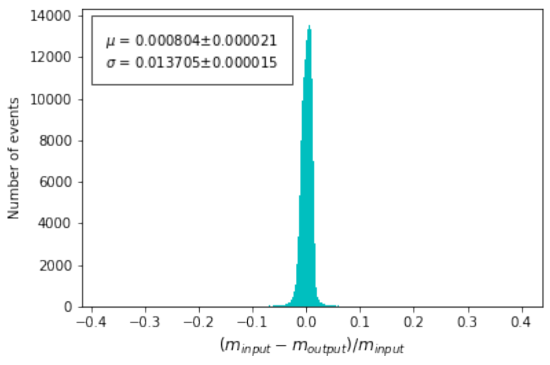


(c) Mass for background

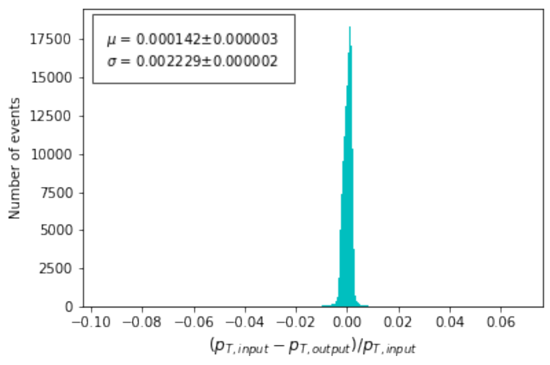


(d) Transverse momentum for background

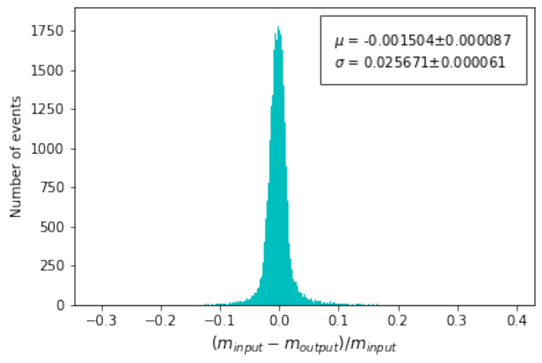
Figure 5.5: Mass m and transverse momentum p_T distribution overlay comparison of input and output of the autoencoder for both signal and background for setup 2.



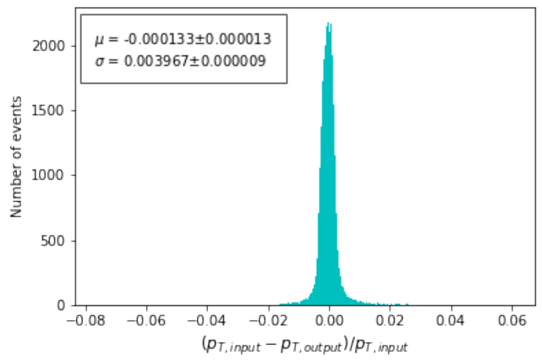
(a) Mass residual for signal



(b) Transverse momentum residual for signal



(c) Mass residual for background



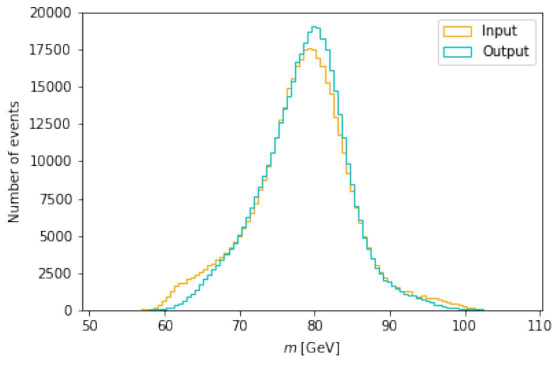
(d) Transverse momentum residual for background

Figure 5.6: Mass m and transverse momentum p_T residuals of input and output of the autoencoder for both signal and background for setup 2.

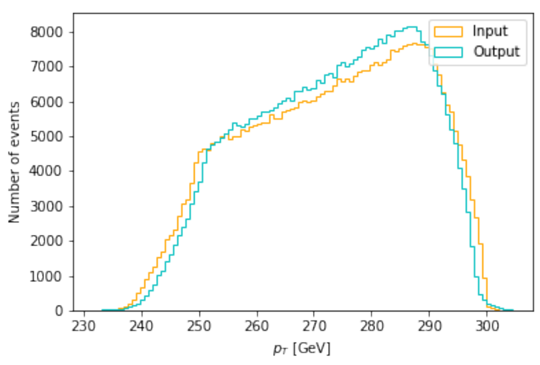
5.2.3 Setup 3: Network With Smaller Bottleneck

Given that it has a bottleneck of 100 nodes (starting with 625 nodes in the input layer), this setup forces the network to compress the input variables to an even lower dimension. This would correspond to a compression to about a sixth of the original size. Figure 5.7 shows the distributions for mass and transverse momentum for both signal and background after passing it through the background-trained network, and Figure 5.8 shows the corresponding residuals.

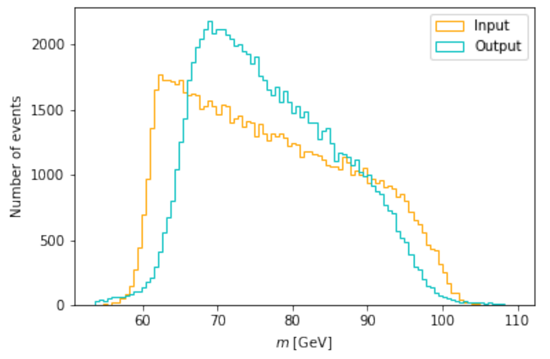
The MSE validation loss is 2.6×10^{-2}. This resulted in over 95 % of the signal images to have a reconstruction error within 8 % for both mass and transverse momentum, and within 15 % respectively for background. This network gives a reconstruction loss that is about two orders of magnitude larger compared to the trivial network in setup 1, and about one order of magnitude compared to setup 2.



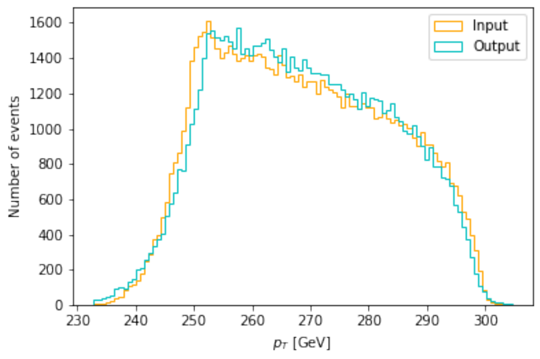
(a) Mass for signal



(b) Transverse momentum for signal

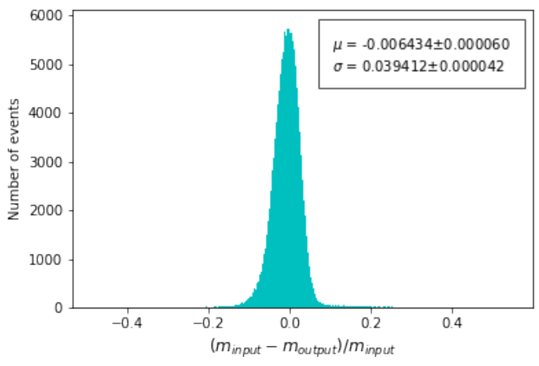


(c) Mass for background

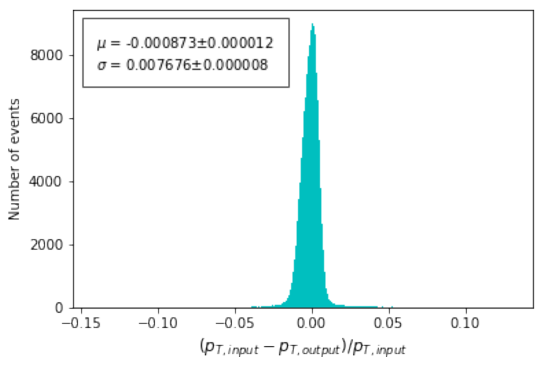


(d) Transverse momentum for background

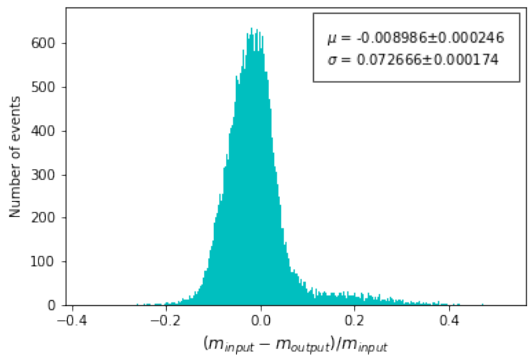
Figure 5.7: Mass m and transverse momentum p_T distribution overlay comparison of input and output of the autoencoder for both signal and background for setup 3.



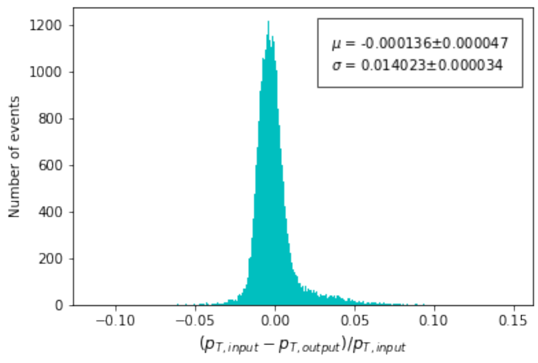
(a) Mass residual for signal



(b) Transverse momentum residual for signal



(c) Mass residual for background



(d) Transverse momentum residual for background

Figure 5.8: Mass m and transverse momentum p_T residuals of input and output of the autoencoder for both signal and background for setup 3.

5.2.4 Summary of Reconstruction Errors

Table 5.1 contains a summary of the reconstruction errors for each distribution and setup. For each setup, the reconstruction error is about two times larger for background compared to signal. The networks are generally better at reconstructing transverse momentum than mass. The ratio of reconstruction errors is approximately the same for each setup, i.e 6-1-12-2. The largest contribution to the error is from the background mass distribution. The smallest contribution to the error is from the signal transverse momentum distribution.

Table 5.1: Summary of reconstruction errors (for over 95 % of the data, i.e within 2σ) in percentages for mass m and transverse momentum p_T, for both signal and background respectively, per setup.

		Signal		**Background**	
	Description	m	p_T	m	p_T
Setup 1	No bottleneck	0.84 %	0.14 %	1.8 %	0.28 %
Setup 2	Large bottleneck	2.7 %	0.45 %	5.1 %	0.79 %
Setup 3	Small bottleneck	7.9%	1.5 %	15 %	2.8 %

5.2.5 Anomaly Detection

Anomaly detection with an autoencoder relies on the fact that anomalous data gives a larger reconstruction loss than for normal data. However, our results show the opposite for each setup (see Table 5.2), when testing a network trained on background with a signal sample. The reconstruction loss ratio between background and signal is larger than two for each setup, while it was expected to be much smaller than one. This translates to the network being better at reconstructing the boosted W-boson jet signatures than the QCD-jet signatures, even though it was trained to recognize the inherent structures of QCD-jets.

Table 5.2: Reconstruction loss for background (normal) and signal (anomalous) for each setup.

	Loss for background	**Loss for signal**	**Ratio**
Setup 1	0.00066	0.00012	5.7
Setup 2	0.0076	0.0019	4.0
Setup 3	0.026	0.0092	2.8

While intuitively unexpected, this may be caused by the simple architecture of the network being poorly suited for this task. Additionally, if the signal and background images are too similar (see Figure 4.1 for average signal and background image), the autoencoder will interpret them coming from the same underlying distribution of images. The background images could be a sample from this distribution with larger variance than for the signal sample considered. As a consequence, the MSE loss would be larger for the sample with the larger variance.

The images, for signal and background respectively, that give the largest contribution to the MSE loss have high-p_T subleading jets (see Figure B.2). The images that give the smallest contribution have conversely subleading jets with lower p_T (see Figure B.3). The network seems to have learned to recognize the one-prong substructure and diffuseness of QCD-jets. When shown images with two hard cores, i.e the signature for boosted W-boson jets, the reconstruction error is larger. However, splitting the sample into two sub-samples with different N-subjettiness (τ_{21}) [25] did not yield any additional explanation for the phenomenon.

In any case, this thesis is the first step towards the use of this particular network structure on jet images. The developed code, documented and released, will be useful for further studies in this direction.

Chapter 6

Conclusions

6.1 Conclusions

The autoencoder network was able to compress and decompress jet images successfully, both in the case of QCD jets and in the case of hadronically decaying W-boson jets. The best compression performance was seen with the network in setup 2 (bottleneck of size 200) where the data was compressed to a third of the original size. The reconstruction error was within 5 % for over 95 % of the data. This corresponds to a significant reduction in data size with just a simple autoencoder, and as a consequence 3 times more data could be stored. Increasing the compression by a factor of two – which corresponds to being able to store 6 times more data – resulted in a significantly larger error, i.e within 15 %.

The autoencoder networks, however, were not able to distinguish background (that they were trained on) from signal. Therefore with the current configuration it is not possible to perform anomaly detection on the jet images provided. Anomaly detection relies on the fact that reconstruction error for anomalous should be larger than for normal data, while the opposite was seen in with our data set. The signal (anomalous) data gave an order of magnitude smaller reconstruction error than for background (normal). The reason behind this is that jet images of signal and background could be too similar for a simple autoencoder to distinguish.

6.2 Future Outlook

This thesis provides a base for further explorations of anomaly detection in jet images with autoencoders. A natural next step is to split the signal and background sample into sub-samples with similar characteristics, and identify what causes the signal to have such a small reconstruction error with a background-trained network. Furthermore, one could perform a hyperparameter scan, i.e finding the optimal set of hyperparameters by systematically testing different combinations. This will most probably improve the overall performance of the network. Exploiting the sparsenesss of the jet images could also improve the performance.

Since the standard autoencoders were not able to differentiate features successfully in the anomaly detection, one could add complexity. One example of adding complexity is through switching the fully connected layers to convolutional layers. A convolutional neural network (CNN) is a type of artificial neural network that is especially well suited for image inputs. They learn local spatial relationships between input nodes (nearby pixels) in order to perform feature detection. A mix of a CNN and an autoencoder is called a convolutional autoencoder (CAE) [33]. Using a convolutional autoencoder could potentially make anomaly detection possible.

If these changes would prove to be successful and anomaly detection is made possible, then the next step would be to test new data, e.g a data set with QCD-jets and simulated dark jets. The network would train on background QCD jets and then try to distinguish the presence of dark jets via testing in a similar fashion as what was done for boosted W-boson jets in this thesis.

References

[1] CERN. Dark matter. URL https://home.cern/science/physics/dark-matter. (accessed: 2020-11-23).

[2] CERN. Detector & technology. URL https://atlas.cern/discover/detector. (accessed: 2020-12-08).

[3] CERN. Facts and figures about the LHC. URL https://home.cern/resources/faqs/facts-and-figures-about-lhc. (accessed: 2020-11-25).

[4] Wulff, E. Deep autoencoders for data compression in high energy physics (2020). URL https://lup.lub.lu.se/student-papers/search/publication/9004751.

[5] Engström, O. Deep learning for anomaly detection in microwave links: Challenges and impact on weather classification (2020).

[6] Wallin, E. Tests of autoencoder compression of trigger jets in the atlas experiment (2020). URL https://lup.lub.lu.se/student-papers/search/publication/9012882.

[7] Gupta, H. Deep-compression for high energy physics data (2020). URL https://zenodo.org/record/4012511.

[8] de Oliveira, L., Kagan, M., Mackey, L., Nachman, B. & Schwartzman, A. Jet-images — deep learning edition. *Journal of High Energy Physics* **2016** (2016). URL http://dx.doi.org/10.1007/JHEP07(2016)069.

[9] Cogan, J., Kagan, M., Strauss, E. & Schwarztman, A. Jet-images: Computer vision inspired techniques for jet tagging. *Journal of High Energy Physics* **2015** (2015). URL http://dx.doi.org/10.1007/JHEP02(2015)118.

[10] Martin, B. & Shaw, G. *Particle Physics*. Manchester Physics Series (Wiley, 2017), 4 edn.

[11] De Angelis, A. & Pimenta, M. *Introduction to Particle and Astroparticle Physics: Multimessenger Astronomy and Its Particle Physics Foundations* (Springer International Publishing AG, Cham, 2018).

[12] Roos, M. Dark matter: The evidence from astronomy, astrophysics and cosmology (2010). 1001.0316.

[13] Bertone, G. & Hooper, D. History of dark matter. *Reviews of Modern Physics* **90** (2018). URL http://dx.doi.org/10.1103/RevModPhys.90.045002.

[14] Kahlhoefer, F. Strongly interacting dark sectors at the LHC. URL https://indico.cern.ch/event/922632/contributions/4098266/attachments/2143260/3612622/LLP8.pdf. (accessed: 2020-11-24).

[15] Doglioni, C., Helen-Genst, M. & Kulkarni, S. Snowmass 2020 - letter of interest. studies of dark shower benchmarks. URL https://www.snowmass21.org/docs/files/summaries/EF/SNOWMASS21-EF10_EF9_Kulkarni_Suchita-149.pdf.

[16] Hansen, E. B. Searches for new physics phenomena with jet final states in the ATLAS detector (2020). URL https://cds.cern.ch/record/2748331.

[17] Evans, L. & Bryant, P. LHC machine. *Journal of Instrumentation* **3**, S08001–S08001 (2008). URL https://doi.org/10.1088/1748-0221/3/08/s08001.

[18] Aad, G. *et al.* The ATLAS experiment at the CERN large hadron collider. *JINST* **3**, S08003. 437 p (2008). URL https://cds.cern.ch/record/1129811. Also published by CERN Geneva in 2010.

[19] CERN. About the ATLAS experiment. URL https://atlas.cern/discover/. (accessed: 2020-11-23).

[20] Aad, G. *et al.* Topological cell clustering in the ATLAS calorimeters and its performance in LHC run 1. *The European Physical Journal C* **77** (2017). URL http://dx.doi.org/10.1140/epjc/s10052-017-5004-5.

[21] CERN. Trigger and data acquisition. URL https://atlas.cern/discover/detector/trigger-daq. (accessed: 2021-01-29).

[22] Cacciari, M., Salam, G. P. & Soyez, G. The anti-kt jet clustering algorithm. *Journal of High Energy Physics* **2008**, 063–063 (2008). URL https://doi.org/10.1088/1126-6708/2008/04/063.

[23] Marzani, S., Soyez, G. & Spannowsky, M. Looking inside jets: an introduction to jet substructure and boosted-object phenomenology. *Lecture Notes in Physics* (2019). URL http://dx.doi.org/10.1007/978-3-030-15709-8.

[24] Kogler, R. *et al.* Jet substructure at the Large Hadron Collider. *Reviews of Modern Physics* **91** (2019). URL http://dx.doi.org/10.1103/RevModPhys.91.045003.

[25] Thaler, J. & Van Tilburg, K. Identifying boosted objects with n-subjettiness. *Journal of High Energy Physics* **2011** (2011). URL http://dx.doi.org/10.1007/JHEP03(2011)015.

[26] Scott, D. J. & Waalewijn, W. J. The leading jet transverse momentum in inclusive jet production and with a loose jet veto. *Journal of High Energy Physics* **2020** (2020). URL http://dx.doi.org/10.1007/JHEP03(2020)159.

[27] Tanabashi, M. *et al.* Review of Particle Physics. *Phys. Rev. D* **98**, 030001 (2018).

[28] Collaboration, A. ATLAS Event at 13 TeV - Highest Mass Dijets Resonance event in 2015 data (2015). URL https://cds.cern.ch/record/2113239. General Photo.

[29] de Oliveira, L., Paganini, M. & Nachman, B. Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis (2017). 1701.05927.

[30] Sylvester, J. Xix. a demonstration of the theorem that every homogeneous quadratic polynomial is reducible by real orthogonal substitutions to the form of a sum of positive and negative squares. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **4**, 138–142 (1852). URL https://doi.org/10.1080/14786445208647087.

[31] The MathWorks, I. What is machine learning? URL https://www.mathworks.com/discovery/machine-learning.html. (accessed: 2020-11-27).

[32] Ohlsson, M. & Edén, P. *Lecture Notes on Introduction to Artificial Neural Networks and Deep Learning (FYTN14/EXTQ40/NTF005F)* (2020).

[33] Aggarwal, C. C. *Neural Networks and Deep Learning: A Textbook* (Springer Publishing Company, Incorporated, 2018), 1st edn.

[34] Skansi, S. *Introduction to Deep Learning - From Logical Calculus to Artificial Intelligence* (Springer Publishing Company, Incorporated, 2018), 1st edn.

[35] Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural Networks* **2**, 359–366 (1989). URL `https://www.sciencedirect.com/science/article/pii/0893608089900208`.

[36] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization (2017). `1412.6980`.

[37] Smith, L. N. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay (2018). `1803.09820`.

[38] Smith, L. N. & Topin, N. Super-convergence: Very fast training of neural networks using large learning rates (2018). `1708.07120`.

[39] Smith, L. N. Cyclical learning rates for training neural networks (2017). `1506.01186`.

[40] Loshchilov, I. & Hutter, F. Decoupled weight decay regularization (2019). `1711.05101`.

[41] Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **313**, 504–507 (2006). URL `https://science.sciencemag.org/content/313/5786/504`.

[42] Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016). `http://www.deeplearningbook.org`.

[43] Aggarwal, C. C. *Outlier Analysis* (Springer International Publishing, 2017), 2nd edn.

[44] Howard, J. & Thomas, R. fastai. URL `https://docs.fast.ai`. (accessed: 2021-02-23).

[45] Howard, J. & Thomas, R. fastai. URL `https://github.com/fastai/fastai/tree/master/`. (accessed: 2021-02-23).

[46] Jupyter, P. About us: Some information about the jupyter project and community. URL `https://jupyter.org/about`. (accessed: 2021-02-23).

Appendix A

Special Relativity

This appendix gives a brief introduction to special relativity, four vectors and natural units.

A.1 Lorentz Transformation

The principle of **special relativity** is that the laws of physics are identical in all inertial reference frames. An inertial frame of reference is a reference frame with no acceleration [11].

Consider a person (P1) standing still on a train station. P1:s frame of reference is denoted S. Consider another person (P2) sitting on a train passing by the station on a train traveling at a constant speed v[1]. This person's reference frame is denoted S'. Since P1 is standing still and P2 is traveling at a constant speed, S and S' are inertial reference frames.

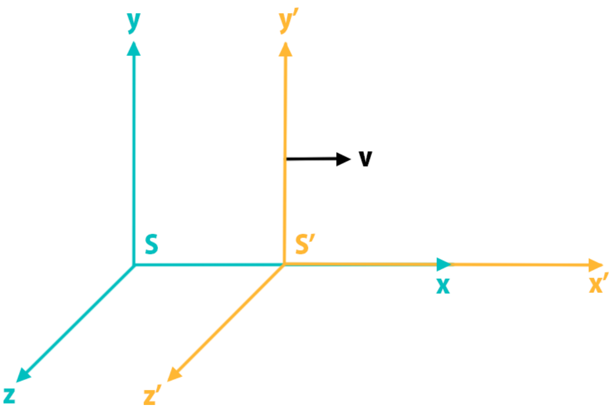


Figure A.1: In relation to S, S' travels at a speed v along the x-axis.

[1]Consider the axes of the the coordinate systems to be shared and the x-axis is along the train tracks. According to that definition, the train is traveling at a constant speed in the x-direction.

If you want to describe an event that occurs at a certain position and time on the train (in reference frame S') from the perspective of P1 (in reference frame S) you have to transform the coordinates. In special relativity, this transform is called the **Lorentz transformation**. For the two inertial reference frames S and S' as in the example, the Lorentz transform would be:

$$\begin{cases} x = \gamma(x' + \beta ct') \\ y = y' \\ z = z' \\ ct = \gamma(ct' + \beta x') \end{cases} \tag{A.1}$$

where x, y, z are the spatial coordinates, t is time, c is the speed of light, $\beta = v/c$ and $\gamma = 1/\sqrt{(}1 - \beta^2)$. [11]. Note that the components orthogonal to the direction of motion are not affected by the transformation. The reference frame S' is said to be **Lorentz boosted** in relation to S in the x-direction. If a quantity is invariant under the Lorentz transformation, it is **Lorentz invariant**, i.e it is the same in all inertial reference frames.

A.2 Four-vectors

An event is characterized by the four variables t, x, y, z that together describe both *when* and *where* it occurs. This can be viewed as a vector in a four-dimensional space. A four-dimensional space mathematically formulated for special relativity is called **Minkowski space**. The four-dimensional vectors in Minkowski space are called **four-vectors**.

In normal three dimensional space, distances are invariant. The distance between two arbitrary objects is the same no matter the perspective. The mathematical formulation of spacetime is done for this is to also be the case. But, since the speed of light has to be a constant (postulate), the algebra with four-vectors is different. In the mathematical formulation of spacetime the coordinates of a four-vector $A^\mu = (A_1, A_2, A_3, A_4)$ are

$$A_1 = x, \quad A_2 = y, \quad A_3 = z, \quad A_4 = iA_0 = ict \tag{A.2}$$

where $i^2 = -1$ is the imaginary unit. This way distances Δs are invariant but not the coordinates Δx_1, Δx_2, Δx_3 and Δx_4 themselves. A distance in space time Δs is:

$$(\Delta s)^2 = (\Delta A_1)^2 + (\Delta A_2)^2 + (\Delta A_3)^2 + (i\Delta A_0)^2 = \Delta x^2 + \Delta y^2 + \Delta z^2 - c^2\Delta t^2 \tag{A.3}$$

Compare this to $\mathbf{r}^2 = x^2 + y^2 + z^2$, i.e the distance from a point (x,y,z) to the origin in 3D space.

For now we have just considered one type of four-vector, i.e (x, y, z, ict). We can apply the Lorentz transformation to any four-vector. Another common four-vector is **four-momentum** $p^\mu = (p_x, p_y, p_z, E/c) = (\mathbf{p}, E/c)$ which relates energy and momentum in spacetime. [11]

A.2.1 Natural Units

In particle physics, **natural units** are used instead of the standard international system of units (SI). In natural units, the scales are adjusted to be more suitable for the scales in particle physics and the commonly used constants $\hbar = c = 1$. This is convenient because then energy, momenta and mass are all expressed in electron volts (eV) and, lengths and time are expressed in eV^{-1} [11]. Throughout this report, all units are natural units. Using natural units simplifies the expression of four-vectors, e.g. $p^\mu = (p_x, p_y, p_z, E/c) \longrightarrow p^\mu = (p_x, p_y, p_z, E)$.

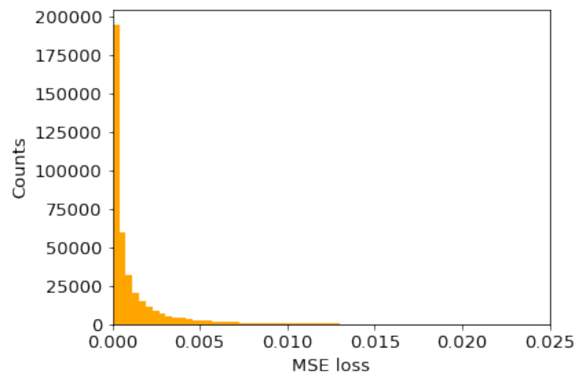
Appendix B

Additional Plots

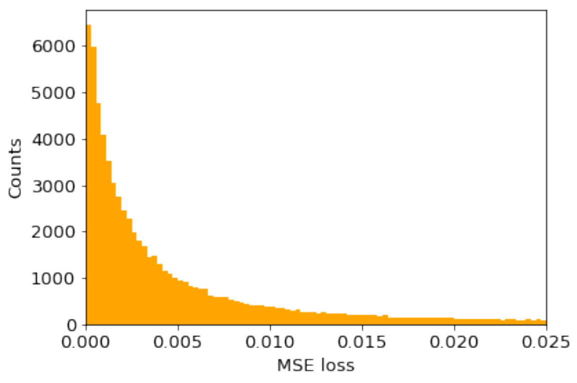
This appendix includes some additional plots to chapter 5.

B.1 MSE Loss Distribution

Figure B.1 shows the MSE loss distribution per image for signal and background for setup 3 (smaller bottleneck, see 4.3). This is to confirm that background does not have outliers that increase the average MSE loss. As can be seen, the MSE loss for signal is generally higher than for background.



(a) Signal

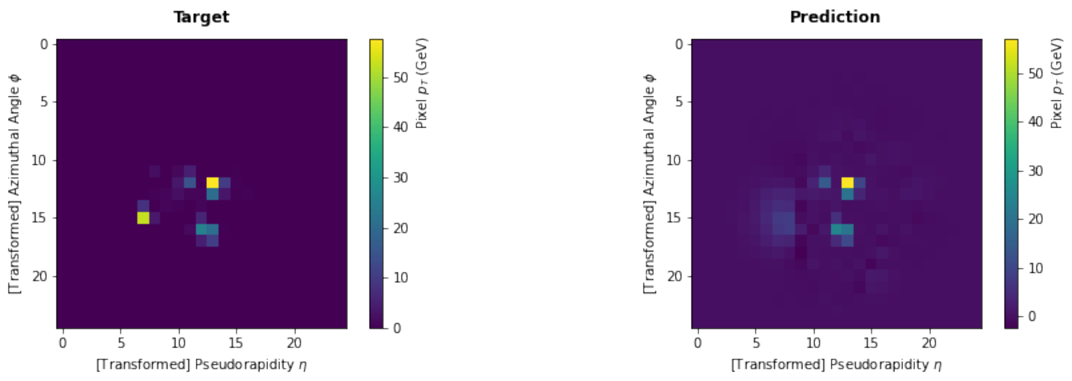


(b) Background

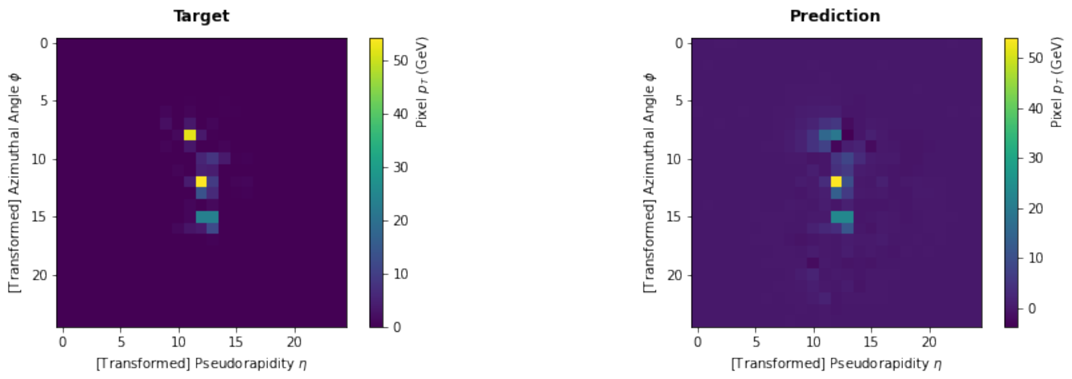
Figure B.1: MSE loss distribution for signal and background for Setup 3. The tails beyond 0.025 are negligible.

B.2 Images With Smallest and Largest Loss

Figure B.2 and B.3 shows the images that have the largest (and smallest) contribution to the MSE loss for setup 3. The target and predicted image is shown. The images with the largest MSE loss have two subjets with high p_T while the smallest have a subjet with a lower p_T.

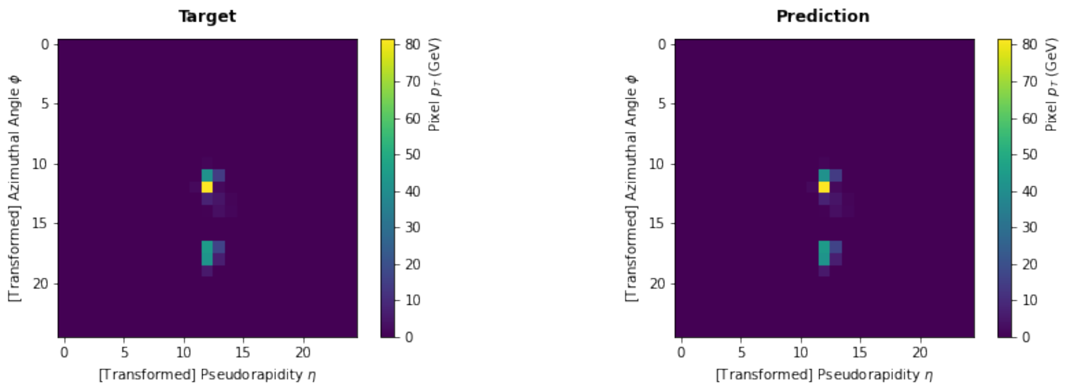


(a) Target and prediction comparison for signal.

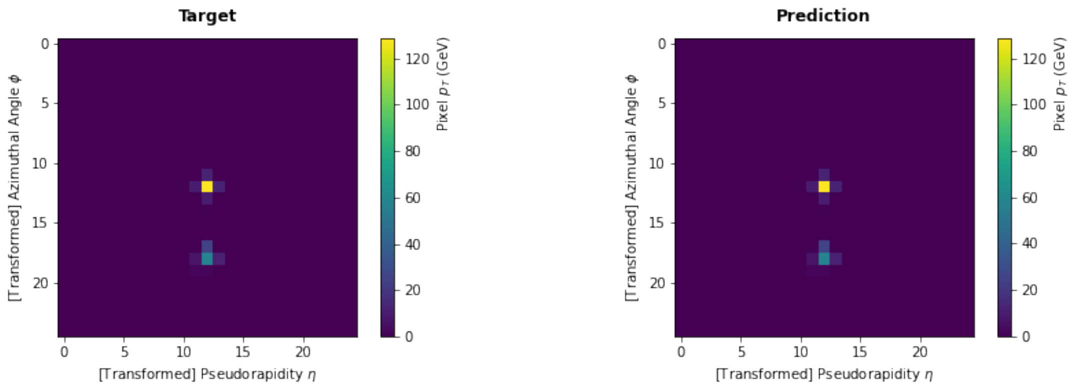


(b) Target and prediction comparison for background

Figure B.2: Target and prediction comparison for the image that has the largest contribution to the MSE loss, for both signal and background.



(a) Target and prediction comparison for signal.



(b) Target and prediction comparison for background

Figure B.3: Target and prediction comparison for the image that has the smallest contribution to the MSE loss, for both signal and background.