Title of the Thesis

Author's name

Masterarbeit in Physik
angefertigt im Physikalischen Institut

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

MMM 2016

I hereby declare that this thesis was formulated by myself and that no sources or tools other than those cited were used.

Bonn, ...
Date Signature

1. Gutachter: Prof. Dr. John Smith
2. Gutachterin: Prof. Dr. Anne Jones

Contents

CHAPTER 1

Introduction

One of my predominant memories from my childhood is Lego. Lego is and was a system of tools that aloud to build a variety of structures from castles to space ships using a variety of stones. The amount of pieces available has grown over the years and that at least in my eyes cost Lego a lot of its original charm. The fascination for a system enabling the user to build almost everything using a very simple set of elemental pieces has stayed with me over the years. Its simplicity and tidiness was not only appealing but also inspiring to me. And it was this fascination I was reminded of when I first learned about particle physics and the standard model of the same. The standard model being a fascinating approach that promised to boil down physics to a surprisingly simple set of pieces and interactions connecting them.
Being allowed to learn about the research process of this field has therefor been a great deal for me.
This work focuses on When it comes to researching at the level of particle physics the usual approach is to generate energy high enough to create new particles. Unfortunately this is a statistic process and high energies and high event count make it an enormous challenge to separate the interesting events from those of no interest. While for some time making cuts and designing detectors in a smart way was sufficient to achieve a high efficiency machine elarning is more and more relevant. In this work the algorithm of an Adversarial Network is used to eparate the tw from the tt bar chanel. The tuning and step by step development of the network strcuture is described to give an outlne of the difficulties to overcome.

Acknowledgements

I would like to thank ...

You should probably use `\chapter*` for acknowledgements at the beginning of a thesis and `\chapter` for the end.

CHAPTER 2

The Standard Model of Particle Physics

Originally in an attempt do unify the electromagnetic, weak and strong force under one theory the standard model of particle physics represents the status quo of particle physics summarizing the elementary particles and their interactions. The model is a gauge quantum field theory and its eternal symmetry is the unitary product group $SU(3) \times SU(2) \times U(1)$ in which the interactions are represented by particles named gauge bosons.

Although failing to answer open questions like the origin of dark matter or neutrino oscillations the Standard Model is a powerful model and has been successful in providing experimental predictions for decades.

2.1 Interactions

2.2 Particles

2.3 Top physics

For more information about the standard model see [3, 4]

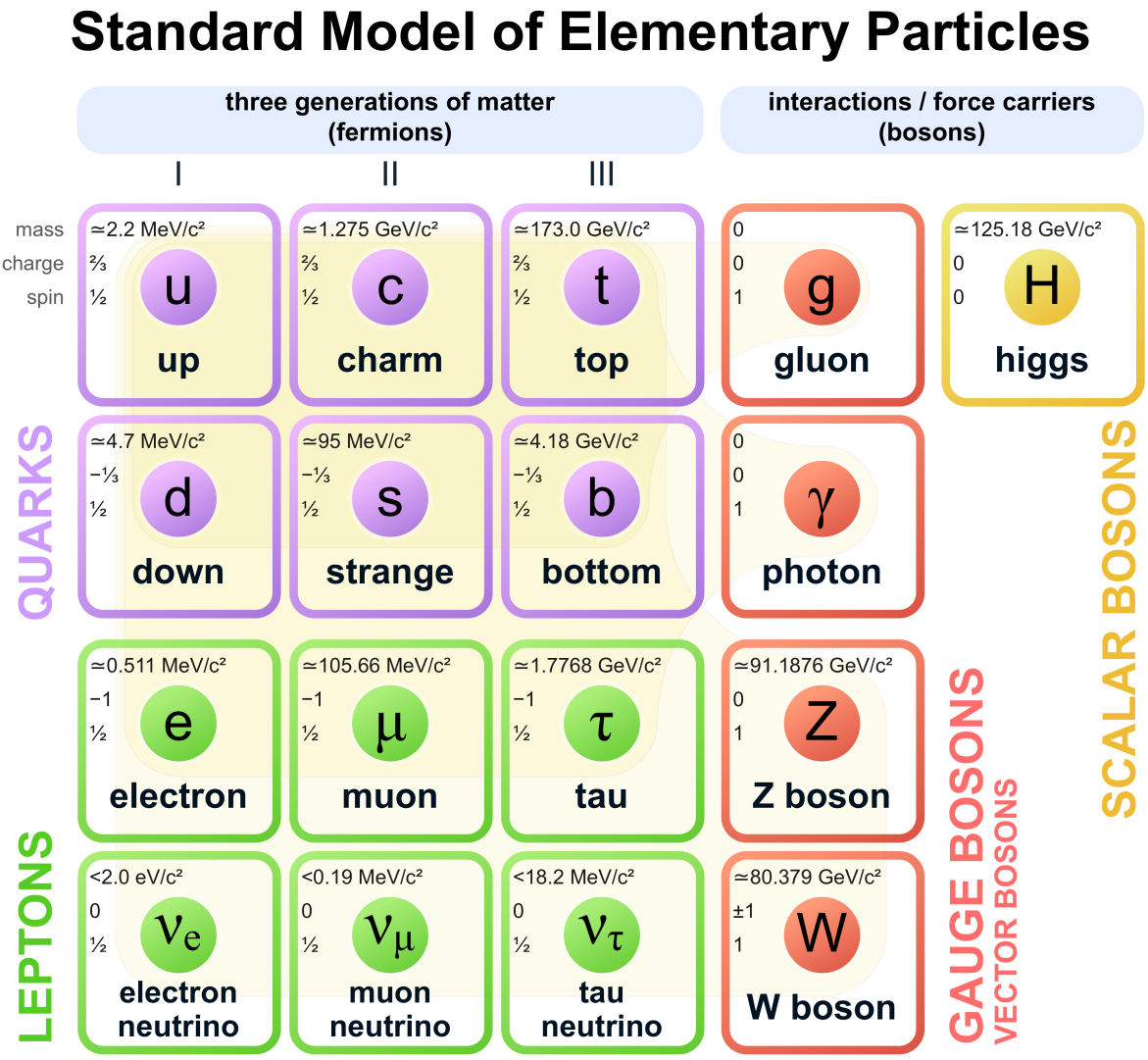


Figure 2.1: Summary table of the Standard Model particles and their properties. The sketch [1] was updated to PDG 2018 data [2]

CHAPTER 3

The LHC and the ATLAS detector

For most researches in modern particle physics there are two main constraints. The first one arises from the statistical nature of decay and creation processes in particle physics. Many of the most interesting events occur extremely rarely and call for large amount of data, more precisely high luminosity, to achieve significant results. Secondly the energy scale of particle physics is enormously high and therefor also the energy needed to allow breaking the structure of particles below the nuclear scale.

This work was carried out using simulations based on the ATLAS [5] detector at the Large Hadron Collider (LHC) [6] which offers both a record energy and luminosity. This chapter gives a summary of both machines and the knowledge necessary to understand the simulations used. First of a summary of the machines' parts and their technical details to are given, followed by a description of how the detector detects particles and how their properties are measured. Simulations of detector events are explained and the event selection for this work is covered.

3.1 Large Hadron Collider

The Large Hadron Collider located at the facilities of the European Organization of Nuclear Research (CERN) close to Geneva was built to extend the frontiers of modern particle physics by delivering high luminosities and reaching unprecedented high energies thereby providing the data for multiple particle physics experiments.

The LHC is a circular particle detector with a circumference of 26.7 km designed to accelerate and collide two counter-rotating beams of protons. The protons are accelerated in bunches of up to 10^{11} protons at energies up to 6.5 TeV and a luminosity of $10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ achieving the record center-of-mass energy of up to 13 TeV. The bunches are pre-accelerated by a number of accelerators before being inserted in the last, so called storage ring. An overview of the acceleration system is given in figure 3.1 and for more detailed information see the LHC design report. [6]. The four interaction points, at which the beams are brought to collision, inhabit the main experiments of the LHC. Two of them are general purpose detectors, namely ATLAS [5] and CMS [7], the third is the LHCb [8] focusing on b physics and lastly ALICE [9] used for investigating heavy ion collisions.Figure 3.2 shows a sketch of the LHC's location and the positions of the four main experiments.

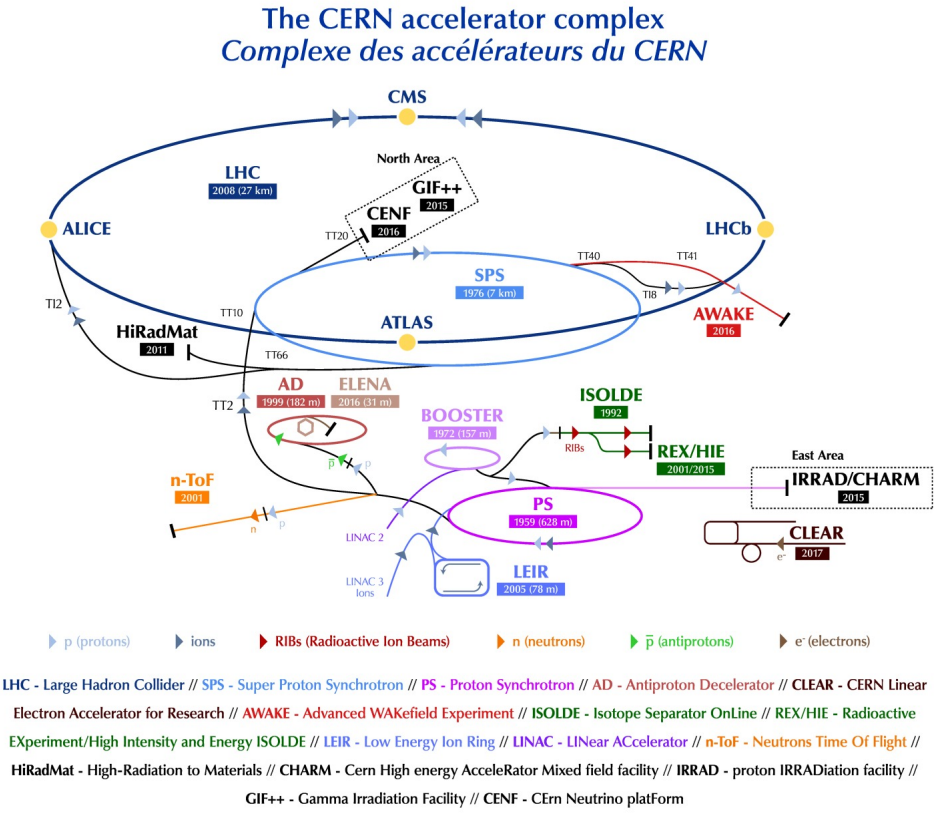


Figure 3.1: Sketch of the accelerator complex of the LHC showing the acceleration systems and the main storage ring with its experiments. [10]

3.2 The ATLAS detector

The ATLAS detector is a general purpose detector meaning it aims at covering a maximum number of final states, enabling reasearchers in may topics of particle physics to use its data.

ATLAS, "A Toroidal LHC Apparatus" has the distinguishing structure of a general purpose detector, its innermost part formed by tracking detectors directly surrounding the interaction point, followed by calorimeters and a further tracking detector for muon detection as the outermost component. All the components are visualized in figure 3.3 including two humans to give an impression of the scale.

The innermost tracking detectors are summarized under the name Inner Detector (ID) and consist of two Silicon detectors namely the Pixel Detector and the Semi Conductor Tracker as well as a straw detector named Transition Radiation Tracker. The Inner Detector allows for precise measurement of not only charged particles' position and thus vertex information, but also for their charge and momentum.

The two calorimeters, being the electromagnetic calorimeter and the hadronic calorimeter, allow to measure the energy of particles by stopping them in the detector material.

The Muon Spectrometer is a further tracking detector identifying particles crossing it as

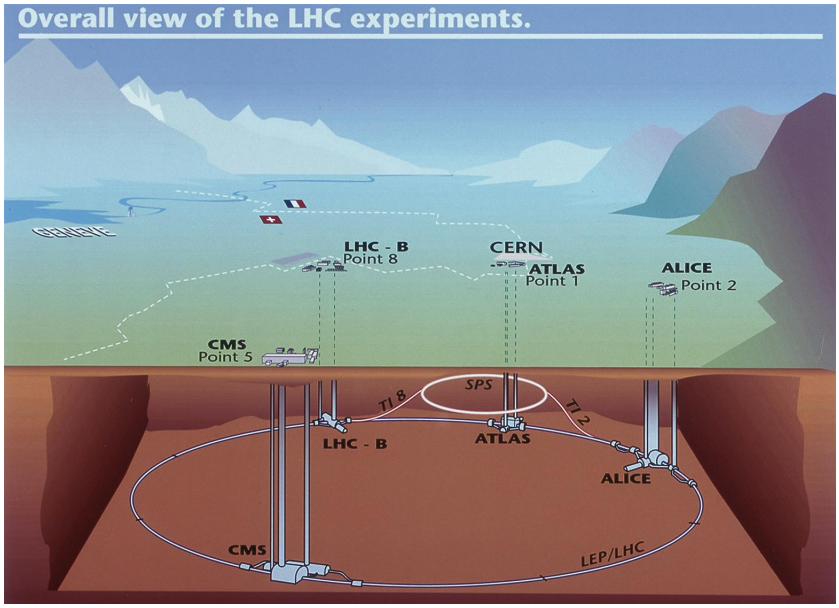


Figure 3.2: Sketch of the LHC ring, the position of the experiments and the surrounding countryside. The four big LHC experiments are indicated (ATLAS, CMS, LHC-B and ALICE) along with their injection lines (Point 1, 2, 4, 8). [11]

muons, as all other charged particles are usually stopped in the other components.

In the following the concept of each detector component is briefly introduced [12] to then summarize how particles can be detected and distinguished. The reconstruction of objects from the detector response is explained and an event selection for tW candidates introduced.

3.2.1 Tracking detectors

Tracking detectors are used to measure a charged particle's trajectory, momentum and charge value, of which two types are used in the ID of the ATLAS detector. The Pixel detector and the Semi Conductor Tracker (SCT) are silicon detectors and the Transition Radiation Tracker (TRT) is a straw-based tracking detector. For all detectors it holds true that they are surrounded by a magnetic field an cover a pseudorapidity range of $|\eta| < 2.5$. The magnetic field results in curved trajectories enabling an estimate of momentum and charge.[14]

Pixel detectors are based on ionisation of charged particles in the semiconductor material. The induced charged is picked up by the detector's pixels providing a position information. To provide a 3-dimensional trajectory the pixel-chips are ordered in 4 layers around the beam pipe where the layer closest to the point of interaction, called Insertable B-Layer (IBL), was added in 2015. It is located only 3.3 cm from the beam pipe and allows to detect vertices very close to the interaction point mainly originating from b quarks giving the layer its name. [15]

The SCT as a silicon microstrip detector is the second silicon-based tracker immediately following on the pixel detector. It consists of modules of four silicon strip sensors organised in four barrel layers and eighteen planar endcap disks.

The TRT is structured in straw tubes each tube being an individual drift chamber with a

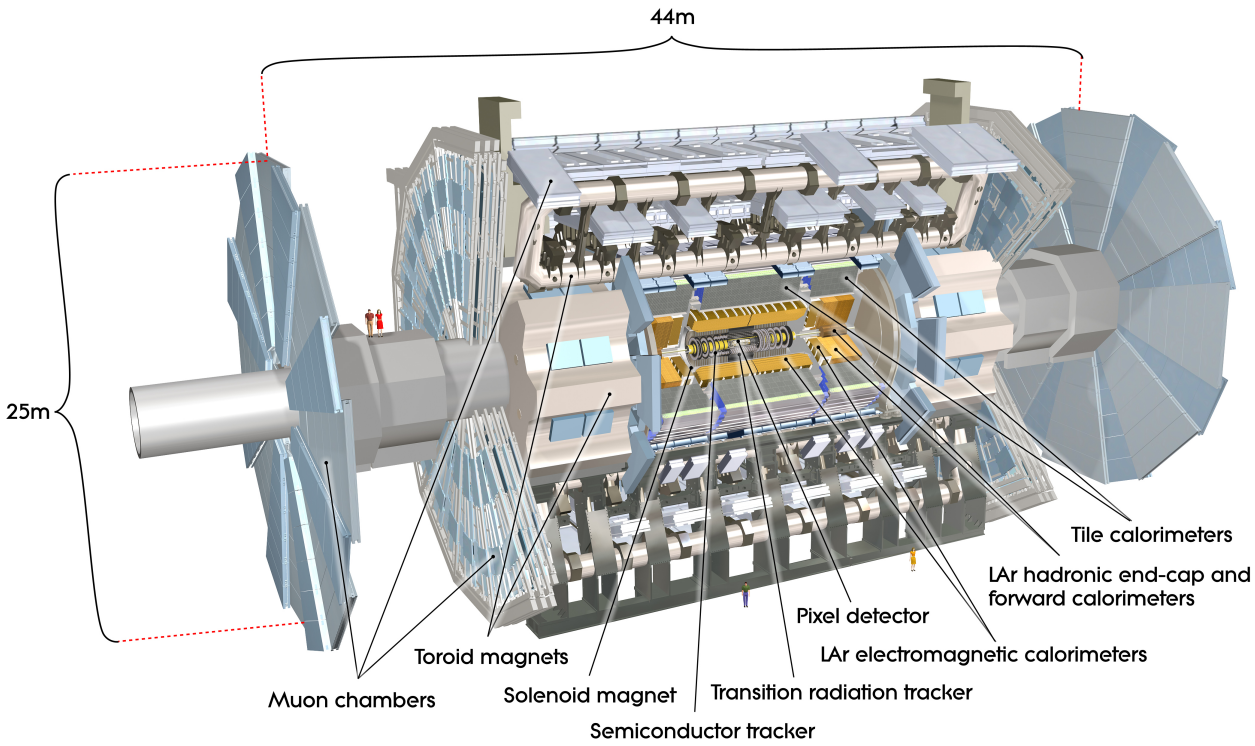


Figure 3.3: Sketch of the ATLAS detector and all its components including two average humans for scale. [13]

strong potential difference due to negatively charged walls. The tubes are filled by a gas mixture (Xe or Ar) causing transversing charged particles to ionize and then by accelerated to the walls. A cascade is initiated and a measurable signal in the potential difference is measured. Between the tubes material is inserted resulting in transition radiation. This radiation has a cross section way higher for electrons thus adding particle information to the track information provided by the TRT.

A particle being detected in a layer of the ID is called a hit. The record of hits gives an estimate on the particle's trajectory and can thereby also give information on the vertex the particle originates from. This vertex information is worth mentioning as for a an experiment with an event count as large as the ATLAS experiment's events interfere and information from so called pileup events can affect the event's information. As pileup originates from different events it can be separated from the event of interest by separating the vertices.

3.2.2 The ATLAS calorimeter system

The ATLAS calorimter system is divided into three main parts. The electromagnetic (EM) calorimeter, compromising a barrel and two end-caps, and the hadron calorimter, built by a tile calorimeter, consisting of a a barrel and two so called "extended barrels", and the hadron end-caps. The third part is the forward calorimeter which additionally focuses on electromagnetic

interaction. The tile calorimeter is scintillator-based apart from that the main part of the calorimeter system is based on liquid argon. The components cover a pseudorapidity range of $|\eta| < 4.9$

Calorimeters determine a transversing particle's energy by exploiting the formation of particle showers. [12] Due to inelastic collisions in the detector's material the energy of the original particle is distributed on a cascade of secondary particles finally stopped by ionization. The resulting charge or photons can be picked up as an estimate of the initial energy.

Electromagnetic calorimeters exploit the energy loss of electromagnetic interacting particles in matter. Mainly photons and electrons loose their energy based on pair production and Bremsstarhlung respectively. The energy loss initializes a cascade of particle decays called an electromagnetic shower.The decay stops when the shower particles do not hold sufficient energy for a decay anymore. The energy of the final state shower particles is picked up by the detector representing the initial particle's energy. The ATLAS ECAL is a sampling calorimeter, built of two alternating layers of absorber and detection layer. In the absorber the showers are induced to then be detected in the detection layers.

As the ECAL uses electromagnetic showers the hadronic calorimeter depends on hadronic shower evolution. Hadronic showers are initialzed due to ionisation or strong interaction with the material's nuclei. If the resulting particles still interact with the material a shower evolves. The hadronic tile calorimeter is made of alternating layers of steel absorbers and scintillators covering a pseudorapidity range of $|\eta| < 1.6$. The hadronic endcap calorimter (HEC) is liquid argon based and covers $1.4 < |\eta| < 3.1$ Due to the larger size of hadronic showers the HCAL occupies more detector space than the ECAL.

3.2.3 The Muon spectrometer

The second tracking detector of ATLAS is the muon spectrometer which is the outermost part of the detector. The task of the spectrometer is to detect charged particles traversing the calorimeter without being stopped or deploying their complete energy, and to do both collect trigger information and information on trajectory and momentum. Due to these two tasks the spectrometer is bifid with the first part being the trigger chamber covering a range of $|\eta| < 2.4$, followed by the high-precision chamber with a range of $|\eta| < 2.7$. The main detector's support feet cause a further gap at about $\phi = 300°$ and $\phi = 270°$.

Normally the only charged particles left to be detected in the muon spectrometer are muons giving the component its name and allowing to provide good trigger information for researches interested in muons in the final event topology.

3.2.4 The ATLAS coordinate system

The ATLAS coordinate system is a right-handed and right-angled coordinate system with the z-axis pointing along the LHC's beam pipe. The corresponding transverse plane is defined by the x-axis pointing towards the ring's centre while the y-axis points upwards. The origin of the system is defined by the nominal point of interaction. The polar angle θ, is the angle between the z-axis and the x-y-plane and the azimuthal angle ϕ is the angle between the x- and the y-axis.

Alternatively, as in this work, an event's topology is described by the azimuthal angle ϕ, the pseudo-rapidity η, and the transverse momentum p_{T}. The pseudo-rapidity replaces the polar angle and is defined as

$$\eta = \frac{1}{2} ln \left[tan \left(\frac{\theta}{2} \right) \right]. \tag{3.1}$$

The transverse momentum is defined by

$$p_{\mathrm{T}} = \sqrt{p_x^2 + p_y^2} \tag{3.2}$$

where p_x and p_y are the momenta along the corresponding axes.

The angular variables are defined within

$$\eta \in [-\infty, \infty], \, \phi \in [-\pi, \pi]. \tag{3.3}$$

This cylindrical system makes use of the shape of the ATLAS detector and the momentum conversation of the transverse plane due to it being perpendicular to all initial beam momenta.

3.2.5 Particle Detection in the ATLAS detector

This section focuses on the detection and distinction of different particle types in the ATLAS detector. The capability and combined information of the detector components is introduced giving an explanation of the general working principle and also of the characteristics defining the events in this work. Figure 3.4 gives an overview of typical particle interactions and detections.

In order to reconstruct the particles in an event low level information is gathered using the direct detector output and then associated to the higher level particle information.

The information from the ID is called a track and contains not only the trajectory but also how consistently a track holds hits in every layer. A track offers momentum and charge information and can be associated with a vertex and a possible energy deposition in a calorimeter. The vertex reconstruction arising from the track information allows to define a primary vertex defined by the highest sum of squared transverse momenta while additional vertices are identified as pileup vertices. Secondary vertices originating from tracks connected to the original vertex can be collected to identify short-lived particles.

The calorimeter data is summarised in clusters. Clusters are neighboring calorimeter cells with energy depositions significantly higher than the expected noise. A cluster is formed around a high energy deposition and can be associated to hadrons or jets or even to a corresponding track.

In the following the higher order objects reconstructed from this basic information are introduced to then explain the decisions made in the event selection for the tW channel.

Electrons are constructed from energy deposits in the EM associated with ID tracks. To improve the decision rule, a likelihood object quantity is constructed from the shape, the ratio of the calorimeter to tracker response, and a set of further variables suitable for a better discriminant. There are three settings for the likelihood object namely tight

medium and loose depending on how restrictive the analysis is. Lastly an isolation quantity is defined based on cones around the track and the EM deposit to further decimate background and fake electrons. [16]

Jets are cones of particles originating from the common hadronisation of a quark or gluon. In the detector they are reconstructed using 3-dimensional topological clusters of calorimeter energy. [17] In addition to this there is further information that can be associated to jets as an ID track or a vertex using a jet-vertex-tagger to minimize the impact of pile-up events and to associate to secondary vertices. For reconstruction the antikt algorithm was used. [18]

Muon reconstruction uses MS hits matched with ID tracks. The choice can be further specified by applying a identification cuts based on MS/ID agreement and integrity of MS hit response. As for electrons isolation can be required. [19]

***b*-jets** are jets originating from the decay of a b quark and therefor a strong discriminant for events containing a t decay. The process of identifying a b-jet is called b-tagging and uses a multivariate discriminant. The topology of b-jets is distinguishable from other jets due to amongst other secondary vertices, vertex alignment of a primary, a secondary b vertex and a tertiary c vertex, the decay length, and the characteristic energy scale. [20, 21]

Missing transverse momentum arises from momentum imbalance in the transverse plane. Momentum in the transverse plane should be preserved due to it being perpendicular to the beam axis and imbalance is an indicator for neutrins escaping the detector. It is calculated using two contributions, one being signals from fully reconstructed and calibrated particles and the other one is information of reconstructed charged particle tracks. [22]

In addition to this the ATLAS trigger system has to be mentioned. Although potentially deserving a chapter of its own for this work it is sufficient to just state and briefly explain trigger information.

Triggers are used to filter events before the actual event selection gets taken into account. Given the incredible luminosity of the LHC such a preselection is important to minimize the data actually processed by the more complicated analysis algorithms and selection schemes. The ATLAS trigger system consists of three trigger, namely the Level 1 (L1), Level 2 (L2), and the Event Filter (EF) where L2 and EF are generally referred to as the High-Level Trigger (HLT).

The L1 is is completely hardware-based and its decision making process is motly based on information from the calorimeter and the muon trigger chambers. The decision step relies on high-p_{T} objects and their multiplicity in an event while also considering missing transverse momentum and the beam condition to provide a first and very broad event selection.

The HLT is based on software and takes the L1 events as input. L2 defines so called regions of interest (ROI) as those regions in the angular plane where the trigger objects for L1 were detected and applies further trgger cuts to these objetcs. The EF fully analyzes the event based on the complete information available.

This trigger information can be used for event selection making sure that certain final state objects are dominant in the topology and also to just apply a first, broad selection.

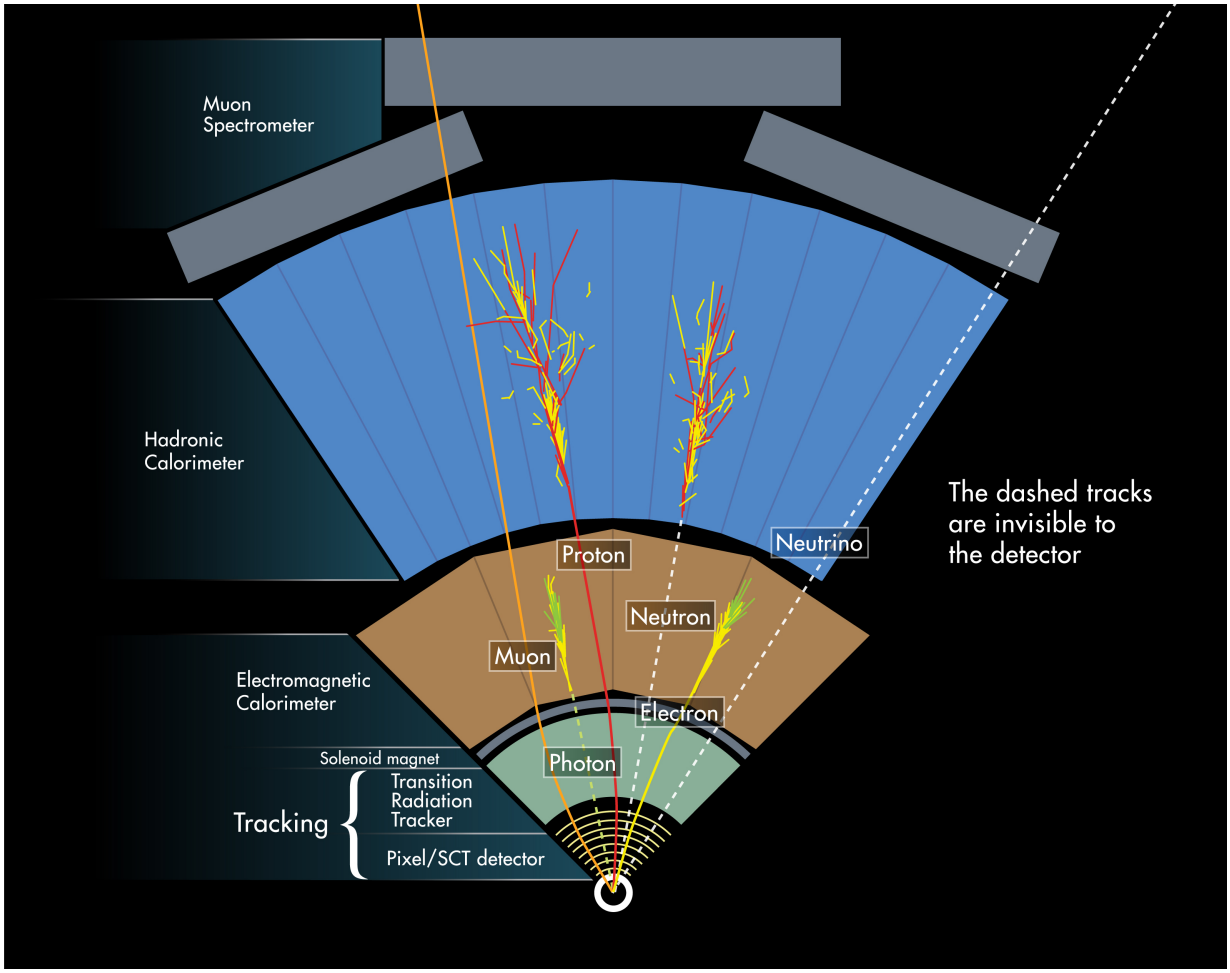


Figure 3.4: Scheme of the ATLAS-detector showing examples of typical particle detections. [13]

3.3 tW event selection

The process separation of interest for this work is tW $t\bar{t}$. Other backgrounds for the tW channel are reduced by applying an event selection:

- A single electron or muon trigger
- Electrons: tightly identified, isolated, $E_{\mathrm{T}} > 26\,\mathrm{GeV}$
- Muons: tight isolation, $p_{\mathrm{T}} > 26\,\mathrm{GeV}$
- Opposite-charge lepton pair
- Leading lepton $p_{\mathrm{T}} > 27\,\mathrm{GeV}$
- Veto for a third lepton $p_{\mathrm{T}} > 20\,\mathrm{GeV}$
- A lepton must match the trigger

- At least one jet with: $p_{\mathrm{T}} > 25\,\mathrm{GeV}$, $|\eta| < 2.5$, tagged at 77 % working point

After this preselection the events are categorized in regions based on the jet and b-jet multiplicities. For this work the region with exactly two b-tagged jets was used denoted as `2j2b`. This region has especially high impact from the NLO interference with a $t\bar{t}$ final state.

3.4 Monte Carlo simulation

A Monte Carlo simulation is a computer based stochastic calculation of a process that in principle could be deterministic but the problem and the amount of statistics requires this solution.

Monte Carlo for the ATLAS detector is the simulation of detector events to calibrate the performance and generate an estimator to compare model and measurement. For this work Monte Carlo is used to train neural networks on and provide the necesary truth information. To generate Monte Carlo one has to base it on quantities that are both possible to calculate but also to measure from experiments. This allows the experiment to be calibrated, the theory to be verified and the analysis to be tested on Monte Carlo. As systematics are based on models certain assumptions need to be made. If the model does not predict reality perfectly systematic uncertainties in the analysis arise from these assumptions. The mass of a particle for example is a quantity that has to be fixed to create a simulation. From the theory the mass can only be fixed within its uncertainty giving reason to several different values to base a simulation on.

The important quantities to base a simulation on for collider physics are the crosssection and the decay width. The crosssection states how high the probability for a certain interaction to occur is. The decay width states how high the probability for an initial state to decay into a certin final state is.

To create Monte Carlo simulations for the ATLAS experiment to steps have to be taken. The first is to simulate the collision of two protons and the resulting events. The second task is to simulate how the detector sees these events taking his accuray and ineeficienceies into account. This tasks are performed by ATLAS Monte Carlo.

The collision of two protons is not sufficiently described by just looking at the two valence quarks, namely two u and one d quarks, instead one has to look at higher order states where a quark can radiate a gluon decaying into a quark antiquark pair. The multitude of these processes in a nucleon leads to a sea of quarks and gluons in addition to the three commonly states valence quraks determining nthe macroscopic properties of the proton.

At the energy scale of the LHC the proton can be seen as a pool of these sea quarks rather than as a product of its three valance quarks. Parton density functions (PDFs) are defined to define the probabilty to find a parton of a certain flavour carrying a certain part of the proton's momentum and thereby describing the structure of the proton at high energy scales. The PDFs are determined in designated experiments and as they are not experiment dependent they can be used a a basis for a proton proton collision simualtion.

CHAPTER 4

The tW Channel

CHAPTER 5

Machine Learning

5.1 The concept of machine learning

For many decades computers have been an integral aspect of science, handling large amounts of data, completing tedious calculations and controlling experiments. For the most part the machines were assigned discrete tasks. They followed step-by-step commands previously designed by human users and had expected outcomes. For particle physics in particular, computers have been used to select and process interesting data from large samples, making the processing of data possible at speeds beyond human capabilities. However the selection rules always had to be generated by the user, therefore requiring an in depth understanding of the underlying system. Machine learning presents a way for a program to establish its own decision rules, improving these over several iterations and thereby learning to solve the problem by itself.

There has been a great effort over the last decades trying to implement a way for machines to learn from known quantities and thereby enable them to analyse complex tasks ranging from voice recognition to object classification. The efficiency and validity of a machine learning model is highly dependent on the human understanding of the problem at hand. The prerequisite to a successful model is the tuning of the degrees of freedom and parameters to the complexity of the assignment. This is called *hyperparameter optimisation*, a task often proportional to the learning process itself.

Machine learning can be exemplified by drawing an analogy to human beings. In order to solve a problem, the machine must understand the system, evaluate a decision step and generate new decision steps. Understanding a system means to be aware of all features and possibilities relevant to the task. Humans have their senses to easily break down their observations into useful features and concepts that can then be processed for decision making. A computer has no senses built in and for most tasks this means that the step of filtering information for a relevant subset of features has still to be done by humans or a good preprocessing algorithm. Once a system has been converted to a subset of features usable by a computer, the step of making its own decisions has to be implemented. This can be done by weighting and interconnecting the information using structures inspired by neurons and synapses in the human brain.

The structure and complexity of the network enables it to learn from data. In addition, a

metric is introduced that measures the quality of the model, called the *cost* or *loss function*. This function allows the network to improve iteratively as a decrease in its value is considered an improvement by the network. Combined with an optimiser which suggests further steps, this function is a basis for a network to independently approach a good decision rule for a somewhat uninvestigated topic.

A very commonly used machine learning technique is the artificial neural network which on its own forms a broad field that builds the base of this work. The most important concepts of machine learning will be explained in the context of neural networks.

5.2 Neural Networks

The artificial neural network, or just neural network, is one of the most commonly known approaches to machine learning. Its structure is inspired by the neurons forming the human brain which is also where it gets its name from.
Instead of neurons, a neural network consists of numerous very simple processors, called nodes. These nodes are usually structured into several layers. As presented in fig (todo). In addition to that there are several ways to structure and connect these nodes often times matching a certain problem. In this explanation only the most commonly way is explained. In that case each node of a layer is getting input from each node in the last layer and is outputting to each node of the following layer. Such a network structure is shown in figure 5.1. It describes the step between a node and the previous layer. The underlying math will be explained in detail later. This is called a feed-forward neural network.

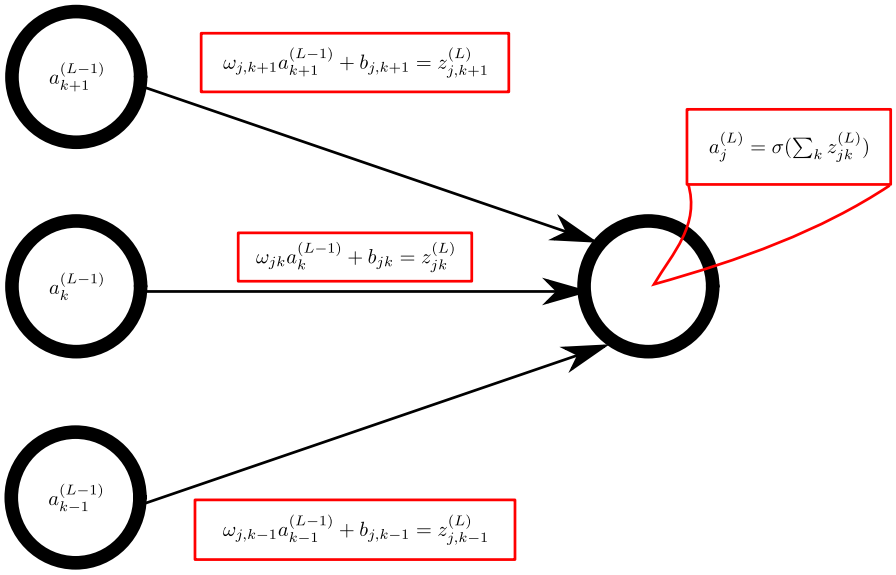


Figure 5.1: Network propagation from layer $(L-1)$ to layer L

The input layer

To understand a task and draw reasonable conclusions first the underlying system has to be understood. Its features need to be found and summarised. The human brain is capable of investigating unknown systems and learn the features that are the most unique or interesting. For that it has its senses to explore the system and process them later. To allow a machine to do something similar the unknown system has to be represented in a way that it is clear for the network what to look out for. This is usually the task that needs the most preprocessing by the user. The simplest case is to submit a list of variables to the network. In particle physics this could be kinematic variables of the final partons in an event. The input to a neural network is just given to the input layer of nodes and then processed through each following layer. For different tasks different layers might deal with different parts of the information but in this work the linear way of giving all information used to an input layer and then processing it is used.

Decision making process

Computers being not much more than very powerful calculators, excel at performing high numbers of clearly defined calculations. The trick is that the task has to be defined clearly. There usually is no uncertainty.
This is very different for the human brain. We rely on a certain uncertainty when processing information through a net of neurons where the output of every neuron is taken as input for the surrounding neurons. The challenge of machine learning is to represent this fuzziness by many somewhat discrete calculations. In the neural network the neurons and their fuzzy interaction is represented by the nodes which are very simple processors. Like neurons each node can use input from many other nodes to create a new output signal. Thereby the input information can be linked to each other in numerous ways. Combined with a weighting system this allows to create complex models and match a variety of problems.

The input of every node is the weighted output of all previous nodes as shown in equation (5.1). z_j^L is the input to the j-th node in the L-th layer. ω_{jk}^L is the weight from the k-th node in the previous layer to this node, a_k^{L-1} is that node's output and b_k the relevant bias, representing a possible intercept of the functionality. The sum indicates that all k previous nodes contribute to the input to the j-th node.

$$z_j^L = \sum_{k=0}^{N} \omega_{jk}^L a_k^{L-1} + b_k \tag{5.1}$$

The weight allows to predict which variables are linked to each other or allow for better decision rules when combined but also easily to weight the importance of features. Furthermore the output of each node is a non-linear combination of the input. This means the output can range from just one and zero to an exponential function and is called the activation function of a node. A common choice is the sigmoid function as presented in equation (5.2). [23]

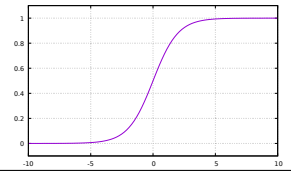
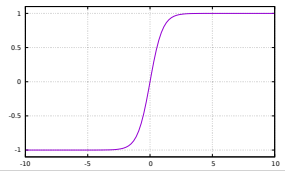
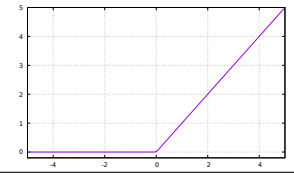
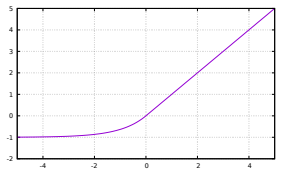
Name	Function	Plot
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	
Tangens Hyperbolicus	$f(x) = \frac{2}{1+e^{-2x}} - 1$	
Rectified Linear Unit, ReLU	$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	
Exponential Linear Unit, ELU	$f(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	

Table 5.1: Selection of activation functions taken from the Keras documentation. [23]

$$a_j^L = \sigma(z_j^L) = \frac{1}{1 + e^{z_j^L}} \tag{5.2}$$

The sigmoid function has output between 0 and 1 which is often times what one wants for nodes especially for the final layer as we are looking for predictions of outcome probabilities. A selection of further activation functions is shown in table 5.1.

For this thesis especially the exponential linear unit or *elu* and the rectified linear unit or *relu* were tested. *Elu* is good for converging the cost to zero rather fast and has the possibility of negative output. *relu* allows for the same benefit as sigmoid while requiring less computational power for the simply linear output for positive values.

Of course the network does still not know the task assigned to it but if we add a cost function to estimate the quality of a decision rule created by a certain combination of the input we can easily make the network approach a better decision rule and hopefully the solution of the problem as the cost function just needs to be minimised. In this work the cost function will be called the loss of the model.

In supervised learning the network is trained with a set of known data. Each event of the training set has a label representing the true outcome value referred to as just label or truth label. Comparing this truth information to the network output makes it very easy to calculate the loss as the deviation of the network output from the desired truth output. A possible loss function is the crossentropy or just binary crossentropy for a binary output result. As in this work the

result is binary, signal and background, the binary crossentropy is the natural choice. Equation (5.3) shows the underlying function where p is the estimated probability for the outcome $\hat{y}$ and y is the label for a correct or incorrect guess.

$$C = -(y \log p + (1 - y) \log(1 - p)) \tag{5.3}$$

The loss is the most important observable for the training quality. This quality must not be mixed up with the overall quality of the model as for the loss only the data trained on is taken into account but not a test data set.

Optimizers - Choosing the next step

The probability interaction of the nodes combined with the loss function enables a network not only to create a model but also to evaluate it. The last missing part is an algorithm that can estimate a step to a model that further minimises the loss. One could certainly do this randomly until the network finds a very low costed decision rule if infinite computational power were provided but that would neither be an efficient nor the desired learning process.

The output of each node in the final layer is defined by the weighted and biased information of the previous nodes and lastly the activation function. For one connection therefore there are three variables that have impact on the loss. Summarising this information for all nodes in a vector defines the loss-vector. The gradient of this vector is an estimator for the impact of each parameter on the overall loss and thereby gives a prefered direction for the model. Updating a network's parameters based on this gradient is called backpropagation. The algorithm works as follows:

1. A certain set of input variables is iterated through all layers of a network resulting in an estimator $\hat{y}$ at each output node.

2. The sum of deviations from the true value at all output nodes is determined as the loss C of the setup. The loss just defines how good the model is.

3. The gradient of the loss is calculated as the partial derivative of all network parameters

$$\frac{\partial C}{\partial a_k^{L-1}} = \sum_{j=1}^{N} \frac{\partial z_j^L}{\partial a_k^{L-1}} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial C}{\partial a_j^L}$$

4. The parameters are then updated backwards through the layers following the negative loss gradient.

This backpropagation algorithm is the backbone of the neural network's learning process.

The decision step based on the gradient defined above is specified by the networks optimizer and deserves a bit more attention. There is different choices of optimisers trying to accommodate different problems as well as some parameters important to understand and tune for an effective training. The length of a learning step has to match the problem's topology to properly let the

model converge. First we define the gradient g in a more general way. The batch size is m and stands for the amount of data processed to evaluate the next step. f is the network for a current configuration or model θ and the output $\hat{y}$. θ summarises all the parameters optimised by the network during the training. The output target provided by the truth information is y.

$$g = \frac{1}{m}\nabla_\theta \sum_j L(f(\hat{y}^j;\theta), y^i) \tag{5.4}$$

The configuration θ is then updated using the gradient and a constant η called the learning rate as it determines the step size for each update.

$$\theta\prime = \theta - \eta g \tag{5.5}$$

Optimisation processes like this are gradient descent based optimisers and can be considered the basis of all optimisers. Depending on the choice they might be based on the whole training sample or just a mini batch of the sample. The most basic form has the learning rate as its only hyperparameter. A good learning rate should be small enough to avoid oscillations but high enough to approach a minimum efficiently fast. A good estimate is given by the Robbins Monro condition:

$$\sum_k \eta_k = \infty \tag{5.6}$$

$$\sum_k \eta_k^2 < \infty \tag{5.7}$$

As the choice of learning rate will not be perfect for every part of the problem's topology, momentum ν can be introduced as a second parameter to the optimizer. [23] The effect desired is that the stepsize becomes greater when the slope is long and the minimum is still far away and to be shorter when approaching the minimum. Momentum scales each step by how aligned previous steps were. That means it will allow avoiding local minima or moving slowly along a slope by enlarging steps at the beginning of the training but also will slow down at the end of the training when the steps become shorter. It promises to speed up the training with less risk of large oscillations which a large learning rate would probably result in. Momentum also takes a single scaling hyperparamter α and is updated each step following:

$$\nu' = \alpha\nu - \eta\frac{1}{m}\nabla_\theta \sum_j L(f(\hat{y}^j;\theta), y^j) \tag{5.8}$$

$$\theta' = \theta + \nu' \tag{5.9}$$

Alternatively one can use Nesterov momentum [23] which is a more advanced adoption of momentum and updates the step a further time after applying the gradient:

$$v' = \alpha v - \eta \frac{1}{m} \nabla_\theta \sum_j L(f(\hat{y}^j; \theta + \alpha \times v), y^j) \tag{5.10}$$

$$\theta' = \theta + v' \tag{5.11}$$

Lastly it can be helpful to decrease the learning rate of the network stepwise while approaching a minimum to avoid oscillations or leaving the minimum in general further. This can be accomplished by the hyperparameter of learning rate decay. It just decreases the learning rate in each iteration t by a small hyperparamter ϕ following the assumption that smaller steps are sufficient close to the minimum. [23]

$$\eta' = \frac{\eta}{1 + \phi t} \tag{5.12}$$

Adaptive optimisers

In addition to the purely gradient based optimisers there are adaptive optimisers. Learning rate and momentum as previously described are difficult to tune to every part of the training process as the topology of the problem might rapidly change. Therefore adaptive optimisers update their parameters based on the training process. There is a set of adaptive optimisers. [23] Often the adaptive optimiser of choice is ADAM. [24] ADAM updates both, its learning rate and its momentum over the course of the training based on an exponentially decaying average of past gradients and past, squared gradients. The average makes sure that the parameters keep getting updated based on past steps. They should be decaying as otherwise the parameters would rapidly shrink. The decay of the averages is defined by a hyperparameter β.

$$\hat{g}^2 = \frac{\sum g^2}{1 - \beta_1^t} \tag{5.13}$$

$$\hat{g} = \frac{\sum g}{1 - \beta_2^t} \tag{5.14}$$

The model's parameters are then updated according to:

$$\theta' = \theta + \frac{\eta}{\sqrt{\hat{g}^2} + \epsilon} \hat{g} \tag{5.15}$$

ADAM is often considered a very good algorithm as it contains many corrections to hyperparamters during the training and thereby allows for easier optimisation but it also needs more computational power.

5.3 Regularisation and Optimisation

Fluctuations and noise in the training sample can be a big problem for a model trained on the sample as a neural network might pick noise and random fluctuations up as features for the decision rule.
This basically is the process of overfitting. The network observes way more features to work with than those actually present in reality or even in a different test sample. The most extreme scenario is that your network is large and deep enough to pick up every single feature in the training sample. If that happens the training error becomes very low and indicates a very good decision rule. For a different sample this decision rule is at most very unreliable but probably strictly wrong resulting in a high test sample error. The network just picked up and remembered every single feature in the training sample instead of general correlations. It becomes a mask of the sample.

A possible way to solve this issue is to stop the training early or to find a good way when to stop the training. This way noisy features might not yet have been picked up by the training. Then again this might also lead to a suboptimal result of the overall training as one cannot be sure that the correct features always get picked up first.

More sophisticated approaches are called regularisations of a neural network. The most commonly used solution is a so called Dropout layer described in the subsection 5.3.1. Additionally a batch normalisation can have an effect of regularisation and is therefor introduced in this section as well.

5.3.1 Dropout

Dropout is attempting to keep the network from relying on subdominant features too much by removing different nodes in each iteration. This forces the network to build models that are not based on strong correlations between nodes. The weights become less interdependent. To simplify it a lot it means training several neural networks depending on which nodes are turned on during a training epoch. It keeps the training in motion for a large number of epochs.

Dropout is added to each layer of a network and can also be restricted to a subset of layers too. It slows down the training as the additional motion slows down the process of finding a minimum but it also accelerates each epoch slightly as it simplifies the network architecture.

5.3.2 Batch normalization

In supervised learning the training result is heavily dependent on the set of data the network is trained on. This means that the performance might change a lot when the test data is very different. Imagine a classifier distinguishing between pictures that show cars and pictures that do not show cars. If the training set contains predominantly green cars the colour green might end up as a strong indicator for the classification car. In general the colour green will not be as dominant and the network will perform slightly worse when trying to classify cars of a different colour. Formally such a change of input is called a covariance shift.
A way to reduce the effect of covariance shift is batch normalization. The output of nodes in general and the weight of a connection in a neural network is not necessarily limited allowing

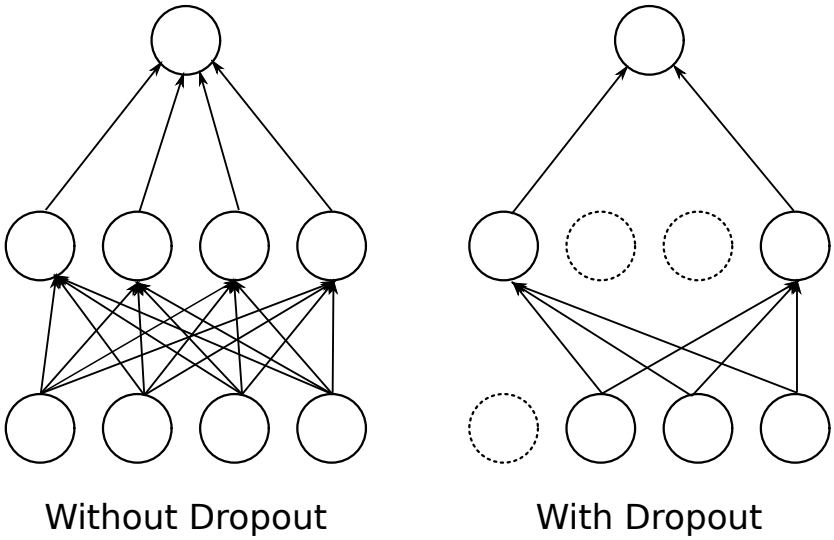


Figure 5.2: Sketch of network before and after the inclusion of dropout. On the left hand side dropout is not applied and all nodes are connected. On the right hand side the dashed circles are nodes excluded by dropout and therefor not connected to the other nodes.

for certain connections to be really dominant and overshadowing less dominant features. We want to avoid this as the dominance of some features might just be present in the training sample. This can be achieved by normalizing the output of each layer in the network to the total output and thereby minimising the effects of strongly overrepresented features. This is done by normalizing each output to the mini-batch mean μ_B and the mini-batch standard deviation σ_B^2.

$$\mu_B = \frac{1}{m}\sum_i x_i \tag{5.16}$$

$$\sigma_B^2 = \frac{1}{m}\sum_i (x_i - \mu_B)^2 \tag{5.17}$$

$$x_{i,norm} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{5.18}$$

5.4 Adversarial Neural Networks

This main part of this work is the examination and training of an adversarial neural network. An adversarial neural network consists of a classifier and a second network that tries to regularise the output of the first classifying network.

In this section the concept of an adversarial neural network is motivated and the underlying mathematics as originally stated in paper [25] are presented. For more information about an approach directly tested on physics see the paper "Learning to Pivot with Adversarial Networks". [26]

5.4.1 The adversarial neural network

Neural networks have been very efficient for classifying tasks but less successful for generative tasks. This was the original problem that gave birth to the idea of a generative adversarial network. Generative networks often times have output that is very easy to distinguish from real samples. The solution suggested is adding a classifier that tries to distinguish between generated samples and real samples. As long this adversary is able to accomplish this task the first network fails at its generative task. Training the two networks against each other disincentives the generative network from using the features not dominant in real samples.

In this work the first network is not a generator but a classifier separating signal events from background events in a Monte Carlo simulation. These simulations contain systematic uncertainties and different samples represent a set of plausible data generation processes. The classifier should not be too dependent on variables with high systematic uncertainties as the network cannot account for differences in the training and testing sample or even in rel data. If the classifier has these strong dependencies on systematic uncertainties it might lead to a high co-variance shift.

Instead of a generated sample and a truth sample, a so called nominal sample and systematic samples are used as input for the second network. Systematic samples have slightly different distributions than the original samples because of changes to the variables with the systematic uncertainties. Training the second network on determining whether it is looking at a nominal or a systematic sample allows to estimate how dependent the model is on variables with high systematic uncertainties. Training the classifier against the adversarial network promises to reduce the effect of systematic uncertainties on the model. If the topology of the problem allows for it this should make the model generated by the classifier pivotal. That means it does not depend on the unknown values of the nuisance parameters.

Mathematically this comes down to a minimax decision rule or a competition between two neural networks. Let us call the classifier $Net1$ and the adversary $Net2$ then the problem becomes:

$$\min_{Net1} \max_{Net2} V(Net1, Net2) = \mathbb{E}_{x\sim\rho_{data}}[logNet1(x)] + \mathbb{E}_{z\sim\rho_{sys}}[log(1 - Net2(z))] \tag{5.19}$$

$V(Net1, Net2)$ is the combined value function for the two adversary networks. The first network is trained to be an optimal classifier represented by $logNet1(x)$ while the second network is trained to distinguish between the nominal and systematics distribution $z \sim \rho_{sys}$ represented by $log(1 - Net2(z))$.

In theory the first classifier should be trained slowly and kept close to its optimum while the second network slowly learns and allows the first network to adapt to it. This is achieved by training the two networks successively over multiple iterations using a combined value function. As this value function a combined loss function is used. It is just the difference between the two separate loss functions with a hyperparameter λ to control the impact of the adversary as shown in equation (5.20).

$$\mathcal{L} = L_{net1} - \lambda L_{net2} \tag{5.20}$$

In the first step of each iteration the first network is trained using the combined loss function $\mathcal{L}$. In the second step the second network is trained using its simple loss function λL_{net2}. Each of the networks has the usual set of hyperparameters to optimise explained in detail in the previous sections missing.

In this work the adversarial network is setup by building a classifying network. The information of the classifier is then fed into both the classifier's output layer and the adversarial network creating a second model based on the first network's model. The networks are then trained successively controlling the combined and separate losses. Figure ?? shows a sketch of the setup.

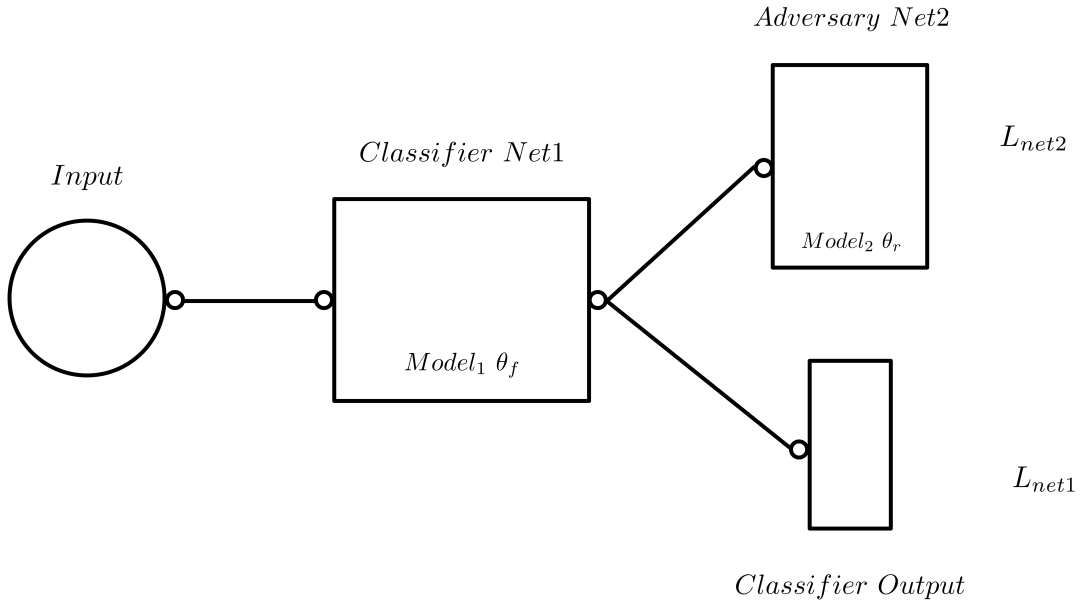


Figure 5.3: Sketch of the network setup. The main part is the classifier. The model generated is fed into both the second network and the classifier's output. This way the two losses L_{net1} and L_{net2} are generated. Furthermore the training of the classifier immediately affects both output models.

The hyperparameter λ is set to tune the impact of the second network on the model. A large λ leads to a very pivotal model but can also decrease the overall quality of the classifier.

In order to better understand the figures shown in chapter 7 the three loss values of an adversarial neural network will be introduced as the last part of this chapter. The first loss belongs to the classifier and is expected to first increase as the adversary finds a good model and to decrease once a more pivotal model is found. The second loss displays the adversary's performance ideally decreasing at first to then increase and saturate as a pivotal model renders it impossible to extract any information. Lastly the combined loss is displayed showing the overall performance and decreasing as good models are found for both the classifier and the adversary. Figure 5.4 shows an example for this loss functions taken from the paper "Learning

to Pivot with Adversarial Networks". [26]

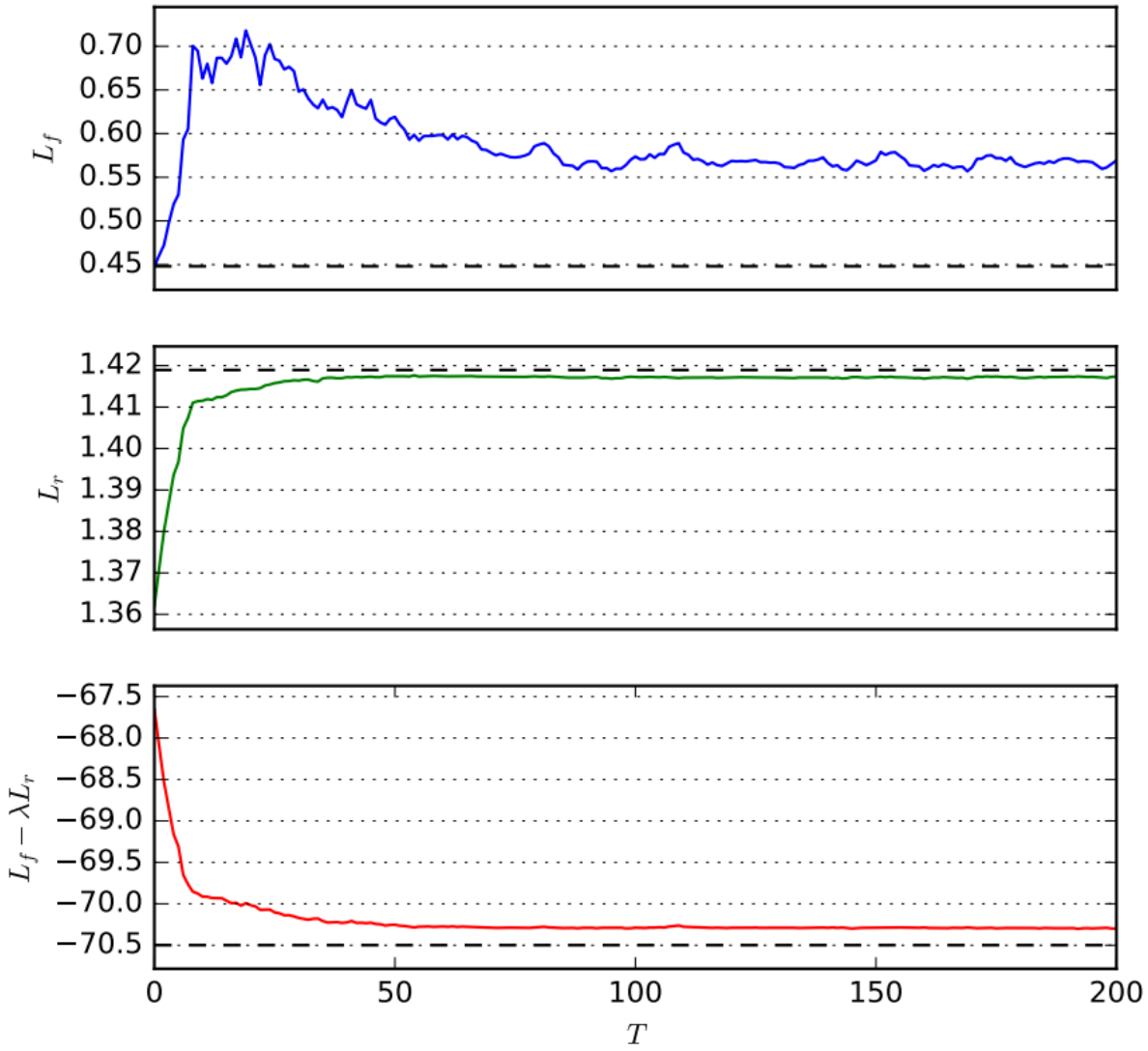


Figure 5.4: Three losses of an adversarial network for $\lambda = 10$ taken from a toy example. [26] From top to bottom the classifier, adversary and combined loss is presented. T is the number of training iterations.

CHAPTER 6

Hyperparameter optimisation of a classifying neural network

The basis od the adversarial neural network is a common neural network trained on signal/background classification. Before the adversary, second network is added the network is optimised on its own to make sure that its setup is sufficient for the classification task. During the adversarial training this setup can be updated if the structure is not optimised for the additional task of a model independent of systematics.

This chapter describes the hyperparameter optimisation of the first network. The first section deals with the input information. The second section explains the choice of the architecture followed by the setup of the optimiser. Lastly regularisation of the network is described.

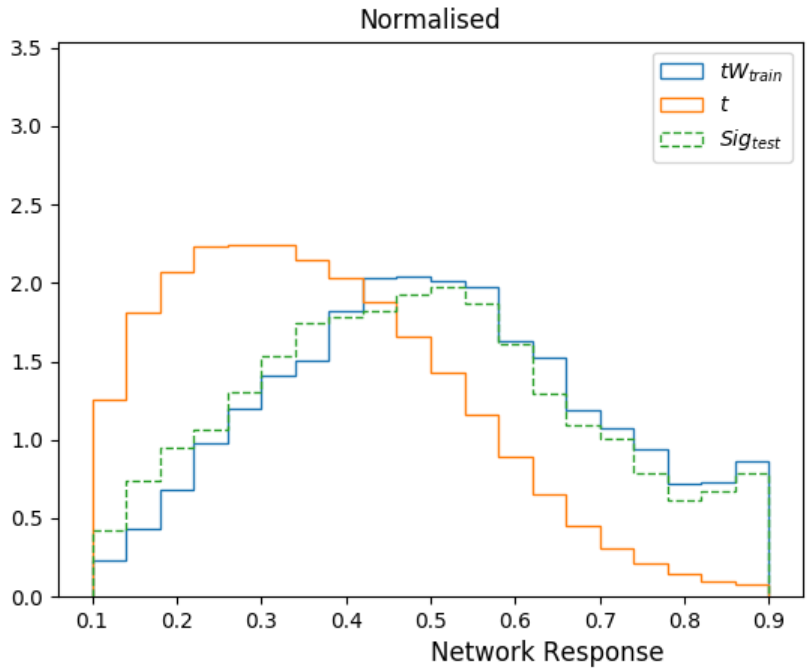
To train the network and to test its performance all the parts need to be in place. Therefor for all hyperparameters had to be initialized with values assumed to be around a good setup. For all testruns one hyperparameter is tested while all the other stay in place. If one parameter gets optimsied it keeps the value for the other testruns. Some of the hyperparameters are correlated which makes it hard to find a true optimum. In addition to the testruns sometimes assumptions were made affecting the range of a testrun or the choice of a final parameter.

6.1 The input variables

got bdt variables
compared to simple kinematic

6.2 The network architecture

The architecture of the neural network is formed by its nodes and layers. The choice of the architecture is nontrivial and as a lot of aspect's of machine learning not an exact science. However, one can make some assumptions about the appropriate architecture. First of all the complexity of the model should about match the complexity of the task assigned. Although it usually is not trivial to find an estimator for a task's complexity and even less to match it to a certain architecture a test series often lads to a good estimate. Both the depth and the overall



size of the model play a role. The depth defines how often the input is processed during the network. The number of nodes is the number of features that can be kept during each step of processing.

In general an architecture that is too deep and wide will pick up too many features too fast and overtrains before it gets to a good minimum. This can be seen in an early divergence between the training loss and the validation loss. An architecture too simple is not able to pick up the features of the task at all and is not learning at all. The loss stays constant or changes very slowly.

In this work a testseries was performed, training a network for a wide range combinations of nodes and layers. For the sake of simplicity the number of nodes per layer was kept constant during each training. Two variables were then plotted against the size of the architecture. First the overall smallest loss the model achieved during the training was plotted. The other variable was the minimal difference between the training and the validation loss. To keep it simple the complexity of the architecture was defined as the product of nodes and layers. These are certainly not the most sophisticated indicators for the model's complexity and its performance. Nonetheless the plots show that region of good performance exists which was chosen for this analysis.

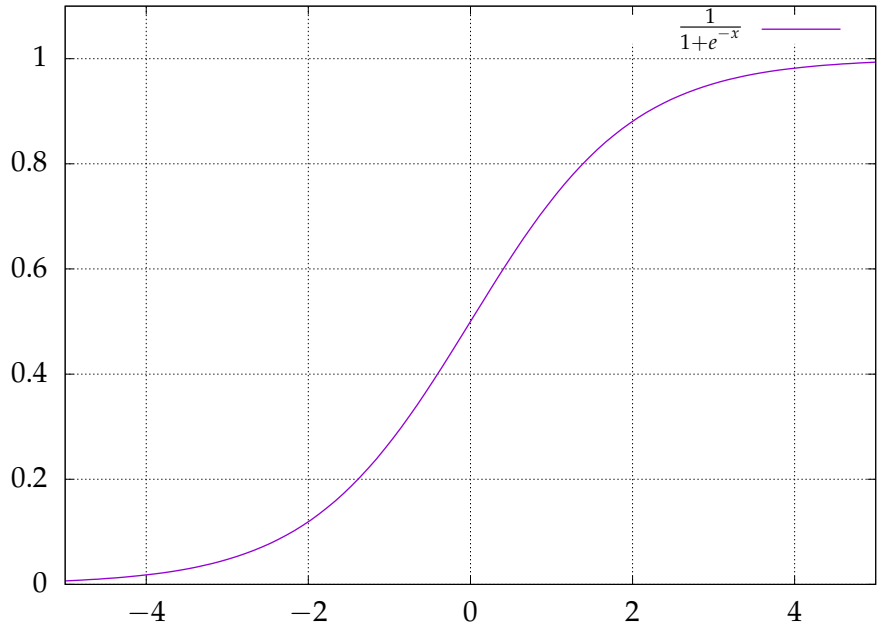
The good region lays between about x and y layers and x and y nodes each

Furthermore the activation function for the hidden layer is elu the activation function

6.3 Setup of the optimisation

For the

6.4 Regularisation


$\frac{1}{1+e^{-x}}$
1
0.8
0.6
0.4
0.2
0
−4
−2
0
2
4

CHAPTER 7

Adversarial Neural Network

In this chapter the setup and training of the Adversarial Neural Network is described. The Network did not achieve the desired results in its initial configuration. For this reason different configurations for the implementation of the network structure are presented in addition to the hyperparamter investigations. The first part of the chapter focuses on the initial run of the ANN using the base network presented in chapter **??**. The results are investigated and the hyperparameters are adapted using an initial setup for the second network. The second network is investigated trying out a number of significantly differetn setups.

Furthermore the problems with this setup are discussed and used to motivate a different approach adapting the network input which is then also investigated. Lasttly the effectiveness of the two models is discussed presenting possible future steps as the training and setup were strongly limited by the time constraint of this work.

7.1 Setup of the first network

The adversarial setup uses the classifier trained and tunes in chapter **??** as a basis. For the classical setup this output is used as input for the adversarial network. To tune the hyperparameters of the second network the classifier was left unchanged.

7.2 Setup of the second network

CHAPTER 8

Conclusions

CHAPTER 9

Machine Learning Tools

This chapter introduces the tools used for the implementation of the neural network. The main tool is the Keras python library. Keras is toolset for deep learning networks developed by google that is able to run on a variety of backends. It summarizes the vector calculations needed for a neural network in modules making it perfect for fast and easy usage.
The backend used in this work is tensorflow.

9.1 Keras

Keras is an application programming interface. It is written in python and able to run on Tensorflow, CNTK or Theano. It was developed by google and summarizes the necessary calculations for running a deep neural network training in fast and easy modules.
[23]

9.2 Tensorflow

[27]

Bibliography

[1] Wikipedia contributors, *Elementary particles included in the Standard Model*, [Online; accessed 03-March-2019], 2006, URL: https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg.

[2] M. Tanabashi et al., *Review of Particle Physics*, Phys. Rev. D **98** (3 2018) 030001, URL: https://link.aps.org/doi/10.1103/PhysRevD.98.030001.

[3] M. Thomson, *Modern Particle Physics*, 6th ed., Cambridge University Press, 2016.

[4] D. Griffiths, *Introduction to Elementary Particle Physics*, 2nd ed., Wiley-VCH, 2008.

[5] G. Aad et al., *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003.

[6] L. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008) S08001.

[7] S. Chatrchyan et al., *The CMS Experiment at the CERN LHC*, JINST **3** (2008) S08004.

[8] A. A. Alves Jr. et al., *The LHCb Detector at the LHC*, JINST **3** (2008) S08005.

[9] K. Aamodt et al., *The ALICE experiment at the CERN LHC*, JINST **3** (2008) S08002.

[10] E. Mobs, *The CERN accelerator complex - August 2018. Complexe des accélérateurs du CERN - Août 2018*, (2018), General Photo, URL: http://cds.cern.ch/record/2636343.

[11] J.-L. Caron, "Overall view of LHC experiments.. Vue d'ensemble des experiences du LHC.", AC Collection. Legacy of AC. Pictures from 1992 to 2002., 1998, URL: https://cds.cern.ch/record/841555.

[12] N. Wermes and H. Kolanoski, *Teilchendetektoren Grundlagen und Anwendungen*, 1st ed., Springer Verlag, 2016.

[13] J. Pequenao, "Computer generated image of the whole ATLAS detector", 2008, URL: http://cds.cern.ch/record/1095924.

[14] W. R. Leo, *Teqhniques for Nuclear and Particle Physics Experiments*, 2nd ed., Springer Verlag, 1994.

[15] Y. Takubo, *The Pixel Detector of the ATLAS experiment for the Run2 at the Large Hadron Collider*, JINST **10** (2015) C02001, arXiv: 1411.5338 [physics.ins-det].

[16] *Electron efficiency measurements with the ATLAS detector using the 2015 LHC proton-proton collision data*, tech. rep. ATLAS-CONF-2016-024, CERN, 2016, URL: https://cds.cern.ch/record/2157687.

[17] G. Aad et al.,
Topological cell clustering in the ATLAS calorimeters and its performance in LHC Run 1,
Eur. Phys. J. **C77** (2017) 490, arXiv: 1603.02934 [hep-ex].

[18] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*,
JHEP **04** (2008) 063, arXiv: 0802.1189 [hep-ph].

[19] G. Aad et al., *Muon reconstruction performance of the ATLAS detector in proton–proton collision data at $\sqrt{s}$ =13 TeV*, Eur. Phys. J. **C76** (2016) 292, arXiv: 1603.05598 [hep-ex].

[20] ATLAS Collaboration, *Performance of b-Jet Identification in the ATLAS Experiment. Performance of b-Jet Identification in the ATLAS Experiment*, JINST **11** (2015) P04008. 127 p, 111 pages plus author list + cover page (128 pages total), 55 figures, 17 tables, submitted to JINST, All figures including auxiliary figures are available at http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/PERF-2012-04/, URL: http://cds.cern.ch/record/2110203.

[21] *Optimisation of the ATLAS b-tagging performance for the 2016 LHC Run*,
tech. rep. ATL-PHYS-PUB-2016-012, CERN, 2016,
URL: https://cds.cern.ch/record/2160731.

[22] M. Aaboud et al., *Performance of missing transverse momentum reconstruction with the ATLAS detector using proton-proton collisions at $\sqrt{s}$ = 13 TeV*, Eur. Phys. J. **C78** (2018) 903, arXiv: 1802.08168 [hep-ex].

[23] F. Chollet et al., *Keras*, https://keras.io, 2015 (cit. on pp. 21, 22, 24, 25, 39).

[24] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*,
arXiv e-prints, arXiv:1412.6980 (2014) arXiv:1412.6980, arXiv: 1412.6980 [cs.LG]
(cit. on p. 25).

[25] I. J. Goodfellow et al., *Generative Adversarial Networks*,
arXiv e-prints, arXiv:1406.2661 (2014) arXiv:1406.2661, arXiv: 1406.2661 [stat.ML]
(cit. on p. 27).

[26] G. Louppe, M. Kagan and K. Cranmer, *Learning to Pivot with Adversarial Networks*, (2016),
arXiv: 1611.01046 [stat.ME] (cit. on pp. 27, 30).

[27] G. B. Team, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*,
Software available from tensorflow.org, 2015, URL: http://tensorflow.org/
(cit. on p. 39).

APPENDIX A

Useful information

In the appendix you usually include extra information that should be documented in your thesis, but not interrupt the flow.

List of Figures

List of Tables