

# Hyperparameter Optimisation of an Adversarial Neural Network in the $tW$ channel at 13 TeV with ATLAS

Christian Kirfel

Masterarbeit in Physik  
angefertigt im Physikalischen Institut

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

April 2019

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. Ian C. Brock
2. Gutachter: Prof. Dr. Klaus Desch

# Danksagung

---

Ich möchte mich bei meinem Vater bedanken, dass er seine Faszination für die Wissenschaft mit mir geteilt hat. Ich bedanke mich bei Ian C. Brock für die Möglichkeit, diese Arbeit zu verfassen, für seine Ermutigung zu neuen Projekten und für die Fähigkeit, seiner Arbeitsgruppe ein Gefühl von Zusammenhalt zu geben

Meiner Familie und meinen Freunden danke ich, dass sie die Stütze für mich waren, ohne die ich diese Arbeit nie hätten schreiben können. Für all die Hilfe, für ermutigende Worte, für die Brettspiel-Abende und Kaffeepausen und nicht zuletzt für all die angeregten Diskussionen kann ich allen anderen Mitgliedern der Arbeitsgruppe nicht genug danken.





# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Theoretical basics</b>	<b>5</b>
2.1	The Standard Model of Particle Physics . . . . .	5
2.1.1	Force carrier particles . . . . .	5
2.1.2	Matter particles . . . . .	7
2.2	Top-quark physics . . . . .	7
2.2.1	Top-quark production . . . . .	7
2.2.2	Top-quark decay . . . . .	8
2.2.3	The $tW$ channel . . . . .	8
2.3	Kinematics of particle colliders . . . . .	9
<b>3</b>	<b>The LHC and the ATLAS detector</b>	<b>13</b>
3.1	Large Hadron Collider . . . . .	13
3.2	The ATLAS detector . . . . .	14
3.2.1	The ATLAS coordinate system . . . . .	15
3.2.2	The Inner Detector . . . . .	17
3.2.3	The ATLAS calorimeter system . . . . .	17
3.2.4	The Muon Spectrometer . . . . .	18
3.2.5	Particle detection in the ATLAS detector . . . . .	18
3.3	$tW$ event selection . . . . .	20
3.4	Monte Carlo simulations . . . . .	21
3.4.1	ATLAS simulation . . . . .	22
3.4.2	Systematic uncertainties in Monte Carlo simulations . . . . .	22
<b>4</b>	<b>Machine Learning</b>	<b>25</b>
4.1	The concept of machine learning . . . . .	25
4.2	Neural Networks . . . . .	26
4.2.1	The input layer . . . . .	26
4.2.2	Decision making process . . . . .	27
4.2.3	Optimisers - Choosing the next step . . . . .	29
4.2.4	Adaptive optimisers . . . . .	31
4.3	Regularisation and Optimisation . . . . .	32
4.3.1	Dropout . . . . .	32
4.3.2	Batch normalization . . . . .	33

4.4	Receiver operating characteristic curve . . . . .	34
4.5	Adversarial Neural Networks . . . . .	34
4.5.1	The adversarial neural network . . . . .	34
<b>5</b>	<b>Hyperparameter optimisation of a classifying neural network</b>	<b>39</b>
5.1	Technical details . . . . .	40
5.2	Final setup of the network . . . . .	40
5.3	The input variables . . . . .	40
5.4	The network architecture . . . . .	42
5.5	Setup of the optimisation . . . . .	44
5.5.1	Choice of the optimiser . . . . .	45
5.5.2	Tuning the optimiser . . . . .	45
5.6	Regularisation . . . . .	47
5.6.1	Dropout . . . . .	48
5.6.2	Batch normalisation . . . . .	49
<b>6</b>	<b>Adversarial Neural Network</b>	<b>51</b>
6.1	Approach I: classical neural network . . . . .	52
6.2	Approach II: hidden layer input . . . . .	53
6.3	Approach III: compressed hidden layer input . . . . .	55
6.4	Summary . . . . .	55
<b>7</b>	<b>Conclusions</b>	<b>57</b>
	<b>Bibliography</b>	<b>59</b>
<b>A</b>	<b>Monte Carlo simulations</b>	<b>63</b>
<b>B</b>	<b>Pre-training losses</b>	<b>65</b>
	<b>List of Figures</b>	<b>67</b>
	<b>List of Tables</b>	<b>69</b>

## Introduction

---

Good tests kill flawed theories; we  
remain alive to guess again.

---

Karl Popper

One of my most influencing childhood experiences is Lego<sup>®</sup> [1]. Lego is a system of interlocking plastic bricks allowing to build a nearly unlimited variety of structures ranging from medieval castles to models of particle detectors [2]. Although the set of differently shaped bricks available to the player has greatly increased over the years, I still see Lego's appeal and charm in the originally simple concept of a limited set of elemental pieces, enabling an inspired user to construct almost anything he or she could think of.

This fascination for an elegant concept of elementary pieces that form everything has stayed with me through my life and education. I found it again in the Standard Model of Particle Physics (SMPP) [3]. The SMPP summarizes the set of elementary particles currently known and their interactions. They represent all the tools necessary to explain our world at least to a great extent. Carried by this inspiration I found my way into high energy collider physics and eventually, with this thesis, had the great opportunity of working with the ATLAS collaboration and taking part in an experiment extending the frontiers of modern physics.

When it comes to particle physics research, a conventional approach is to collide common particles at energies high enough to create new particles. Additionally, in order to filter collisions for event of interest, one needs to have a high rate of events to generate a sufficient sample size. It suggests itself to apply machine learning techniques on the task of filtering events at collider experiments. In this work, an adversarial neural network [4] is introduced. It adds the promise of minimising the sensitivity of a selection model for systematic uncertainties to the network's classification task.

At first, an introduction to the state of the art for particle physics is given. The SMPP is introduced and further information on physics incorporating top-quarks is presented thereby motivating the difficulties in separating the Standard Model process from its background and reducing its systematic uncertainties. Secondly the Large Hadron Collider and the ATLAS detector are introduced to explain how events are generated and reconstructed in the detector and how simulations for the experiment are generated. Then the concept of machine learning

is described with a focus on neural networks and adversarial neural networks including their hyper-parameters. Finally, the setup and training of a classifying neural network is described and then used as basis for the setup of an adversarial neural network. The results using an adversarial neural network are discussed based on the promises it might offer for an analysis highly dependent on systematic uncertainties.

# Theoretical basics

---

In this chapter the Standard Model of Particle Physics, or Standard Model, with its elementary particles and interactions is introduced. Furthermore, the analysis in this thesis is motivated and the underlying problems are outlined. For this reason a brief introduction to the theory of particle production and decay is given.

## 2.1 The Standard Model of Particle Physics

Originally, created in an attempt to unify the electromagnetic, weak, and strong force under one theory, the Standard Model of particle physics represents the status quo in this field summarizing the elementary particles and their interactions. The model is a gauge quantum field theory and its internal 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. Figure 2.1 shows the Standard Model particles and their central properties which will be introduced in this chapter starting with the interactions and their mediators followed by a summary of the particles and lastly a section on the top-quark in addition to its properties directly relevant for the research in this thesis.

Although failing to answer open questions like the origin of dark matter or neutrino oscillations, the Standard Model has proven to be a powerful model being very successful in providing experimental predictions for decades.

### 2.1.1 Force carrier particles

There are three interactions represented in the Standard Model: the electromagnetic interaction, the weak interaction, and the strong interaction. Each is represented by an integer-spin particle, called *mediator boson*. Gravity is usually not included in this model as it is negligible at this scale.

The electromagnetic force is described by the theory of quantum electrodynamics. Its mediator is the photon, a massless particle that couples to the electric charge. The weak force, most commonly known as the interaction responsible for the  $\beta$ -decay, couples to the weak charge; which is an intrinsic property of all fermions. Its mediators are the  $Z$ - and the two

### Standard Model of Elementary Particles

		three generations of matter (fermions)			interactions / force carriers (bosons)	
		I	II	III		
mass		$\approx 2.2 \text{ MeV}/c^2$	$\approx 1.275 \text{ GeV}/c^2$	$\approx 173.0 \text{ GeV}/c^2$	0	$\approx 125.18 \text{ GeV}/c^2$
charge		$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	0	0
spin		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	0
		<b>u</b> up	<b>c</b> charm	<b>t</b> top	<b>g</b> gluon	<b>H</b> higgs
	<b>QUARKS</b>	$\approx 4.7 \text{ MeV}/c^2$	$\approx 95 \text{ MeV}/c^2$	$\approx 4.18 \text{ GeV}/c^2$	0	
		$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0	
		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	
		<b>d</b> down	<b>s</b> strange	<b>b</b> bottom	<b><math>\gamma</math></b> photon	
	<b>LEPTONS</b>	$\approx 0.511 \text{ MeV}/c^2$	$\approx 105.66 \text{ MeV}/c^2$	$\approx 1.7768 \text{ GeV}/c^2$	0	$\approx 91.1876 \text{ GeV}/c^2$
		-1	-1	-1	1	
		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$		
		<b>e</b> electron	<b><math>\mu</math></b> muon	<b><math>\tau</math></b> tau	<b>Z</b> Z boson	
		$< 2.0 \text{ eV}/c^2$	$< 0.19 \text{ MeV}/c^2$	$< 18.2 \text{ MeV}/c^2$	$\pm 1$	$\approx 80.379 \text{ GeV}/c^2$
		0	0	0	1	
		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$		
		<b><math>\nu_e</math></b> electron neutrino	<b><math>\nu_\mu</math></b> muon neutrino	<b><math>\nu_\tau</math></b> tau neutrino	<b>W</b> W boson	
						<b>GAUGE BOSONS</b> <b>VECTOR BOSONS</b>
						<b>SCALAR BOSONS</b>

Figure 2.1: Summary table of the Standard Model particles and their properties. The matter particles, quarks in purple and leptons in green, and the force carriers in red are shown. In addition, the Higgs, in yellow, is included as the latest addition to the model. The sketch [5] was updated to PDG 2018 data [6]

opposite-charged  $W$ -Bosons. The electromagnetic and the weak force can be unified as the electroweak interaction forming a  $SU(2) \times U(1)$  symmetry. One aspect of the weak force is that it allows for quark-flavour change via an interaction with a  $W$ -boson. The probability of this flavour change is represented by the Cabibbo-Kobayashi-Maskawa matrix, CKM matrix, which is described in detail in the PDG [6].

The strong force is responsible for the binding of quarks in the nucleon, the protons and neutrons. It is mediated by the 8 differently flavoured gluons. It only couples to particles with colour charge and is represented by the  $SU(3)$  symmetry term in the Standard Model.

As already stated the gravitational force is not included in the Standard Model as its coupling strength at the scale is only of the order  $10 \times 10^{-37}$ . Although it is not a mediator, the Higgs boson was included in the Standard Model after its discovery in 2012. It is the particle associated to the mechanism which gives all other particles mass.

### 2.1.2 Matter particles

The non-mediator particles in the model are called *fermions*, which are half-integer spin. They can be broadly classified into three generations of leptons and quarks

The first generation of particles forms the most common form of matter. The electron, the electron-neutrino, and the up- and down-quark, which are the main constituents of the proton and neutron. In the higher generations the defining quantum numbers stay the same while the mass of the particles increases. The quarks that share the properties of the up-quark are called up-type quarks, and they are the charm- and top-quarks. In contrast, there are down-type quarks: the strange- and bottom-quarks. Bound quark states are called hadrons. Moreover, there is an antiparticle for each particle with all quantum numbers reversed.

The neutrinos interact only via the weak interaction. There is one neutrino for each lepton-family. The non-neutrino leptons are the electron,  $e^-$ , the muon,  $\mu^-$ , and the tauon,  $\tau^-$ , in order of increasing mass. They have an intrinsic electric charge which allows them to interact electromagnetically. Furthermore, the six quark flavours are up and down, strange and charm, bottom and top. The quarks are the only particles interacting with all three forces. They carry not only a weak and an electric charge but also colour charge enabling them to interact with gluons.

## 2.2 Top-quark physics

The most essential aspects of particle interactions, underlying the events of interest of this work, are the production and decay processes of the top-quark. This section introduces the main properties of the top-quark. The top-quark, being the third generation up-type quark, is special because its mass exceeds the masses of the other quarks by orders of magnitude. Its mass is about  $173 \text{ GeV}/c^2$  which is higher than the masses of the weak mediators. It has a lifetime of  $\sim 5 \times 10^{-25} \text{ s}$  which is smaller than the typical hadronisation time, meaning that the top-quark forms no bound states with other quarks and instead decays. These essential properties of the heaviest quark lead to some interesting aspects and motivate one to research its properties.

To describe the production and decay of the top-quark Feynman-diagrams will be used. These diagrams are used to present the process and the underlying math. The time-axis is defined as the positive x-axis. Fermions are depicted by a solid line with an arrow; for particles it points with time and for antiparticles against it. Gluons are represented by curly the other bosons by wavy lines. Alternatively the mediators of the weak force are sometimes depicted using dashed lines. In addition diagrams may be labeled with an order. The lowest order diagram is the *leading order* process, LO. To achieve a higher precision corrections are applied. One then speaks of *next-to-leading order*, NLO, where the next-to can added iteratively for even higher order corrections.

### 2.2.1 Top-quark production

Most commonly top-quarks are created via the strong force in a top- anti-top-quark final state. Both gluon-gluon fusion and quark-antiquark-annihilation are possible production processes.

Figure 2.2 shows the processes at LO. Additionally, top-quarks can be produced as single

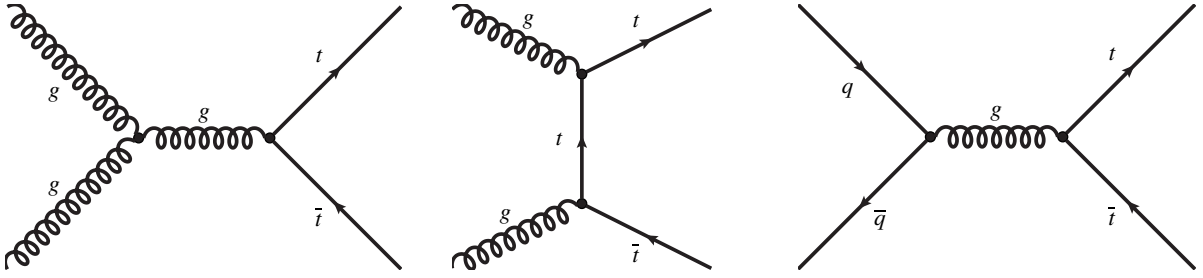


Figure 2.2:  $t\bar{t}$  pair production feynman diagrams at LO.

top-quarks via the electroweak interaction. The dominant process is the production through the interaction of a bottom-quark and a  $W$ -boson shown in figure 2.3(a). Also possible but the least common is the  $s$ -channel involving a virtual  $W$ -boson displayed in figure 2.3(b). Finally the channel of interest for this work is the  $tW$ -channel diagrammed in figure 2.3(c). The probability for a production to occur is denoted using the *cross-section* of the process.

## 2.2.2 Top-quark decay

The top-quark almost exclusively decays into a  $W$ -boson and a bottom-quark. The  $W$  can subsequently decay leptonically, in a lepton-neutrino pair, or hadronically, in a quark-antiquark pair. The channel of interest for this work is the  $tW$ -channel which is diagrammed in figure 2.3(c). The mode in which both  $W$ -bosons decay leptonically is investigated. The analysis is described in greater detail in the following subsection.

## 2.2.3 The $tW$ channel

Finally the channel of interest for this work is the  $tW$ -channel diagrammed in figure 2.3(c). Its final state only differs from the  $t\bar{t}$ -final state by one missing bottom-quark. This makes the  $t\bar{t}$  channel the dominant background for the  $tW$ -signal. The separation gets especially complicated because the cross-section of  $tW$  is about 10 smaller than the  $t\bar{t}$  cross-section. The cross-sections for the channels are listed below:

$$\sigma_{tW} \sim 71.7 \text{ pb} \quad (2.1)$$

$$\sigma_{t\bar{t}} \sim 832 \text{ pb}. \quad (2.2)$$

Figure 2.4 shows the final state of the  $tW$  decay with both  $W$ -bosons decaying leptonically at LO. For that reason, the channel is named dilepton channel. At NLO a gluon splitting can result into a further bottom-quark in the final state. Figure 2.5 shows the  $t\bar{t}$  final state in comparison to the NLO final state of the  $tW$ -channel. In the final state these channels are not distinguishable, i.e., they interfere. These diagrams will be referred to as doubly resonant, in contrast to the singly resonant diagrams such as diagrammed in figure 2.2. Given that the  $t\bar{t}$ -cross-section is 10 orders larger than the  $tW$ -cross-section this gives rise to an NLO correction exceeding the



actual LO cross-section. This results in the  $tW$ -channel not being well-defined at NLO. To allow treating  $tW$  as a separate process a workaround has to be used. There are two possible schemes for handling the interference in the calculation of the cross-section:

**Diagram Removal (DR)** removes all diagrams containing a second top-quark propagator that can be on-shell. It is used for the production of the nominal sample in this work.

**Diagram Subtraction (DS)** only the  $t\bar{t}$  contribution is canceled when the top-quark is on-shell. This scheme is used for the systematics sample.

For more information on the schemes and their motivation see section 3.4.2,

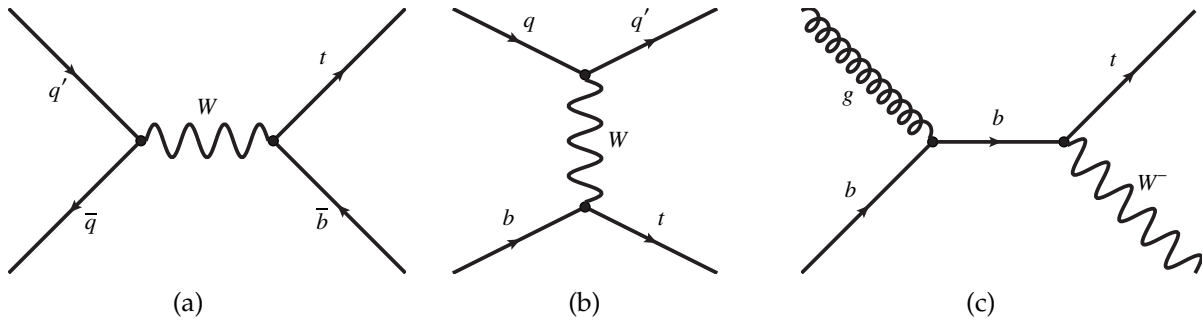


Figure 2.3: Single-top-production diagrams: s-channel (b), t-channel (a),  $tW$ -channel (c)

## 2.3 Kinematics of particle colliders

There are a few kinematic variables and experimental properties worth discussing because they are characteristic for collider experiments which will be briefly introduced in this section.

One of the most important attributes of a collider experiment is its centre-of-mass energy,  $\sqrt{s}$ . The centre-of-mass energy is a Lorentz invariant value which is valuable in particle physics because it denotes the available energy for particle production in an experiment. It is defined as:

$$\sqrt{s} = \sqrt{\left(\sum_i E_i\right)^2 - \left(\sum_i \vec{p}_i\right)^2} \quad (2.3)$$

where  $E_i$  and  $\vec{p}_i$  are the energies and momenta of the initial or final state particles. In the case of the Large Hadron Collider, which is simulated for the Monte Carlo samples used in this thesis and introduced in chapter 3, two proton beams of equal energy are brought to collision resulting in an energy formula of

$$\sqrt{s} = 2E_{beam} \quad (2.4)$$

assuming there are two beams with energies much higher than the particles' masses. Therefore one can neglect them in the calculation.

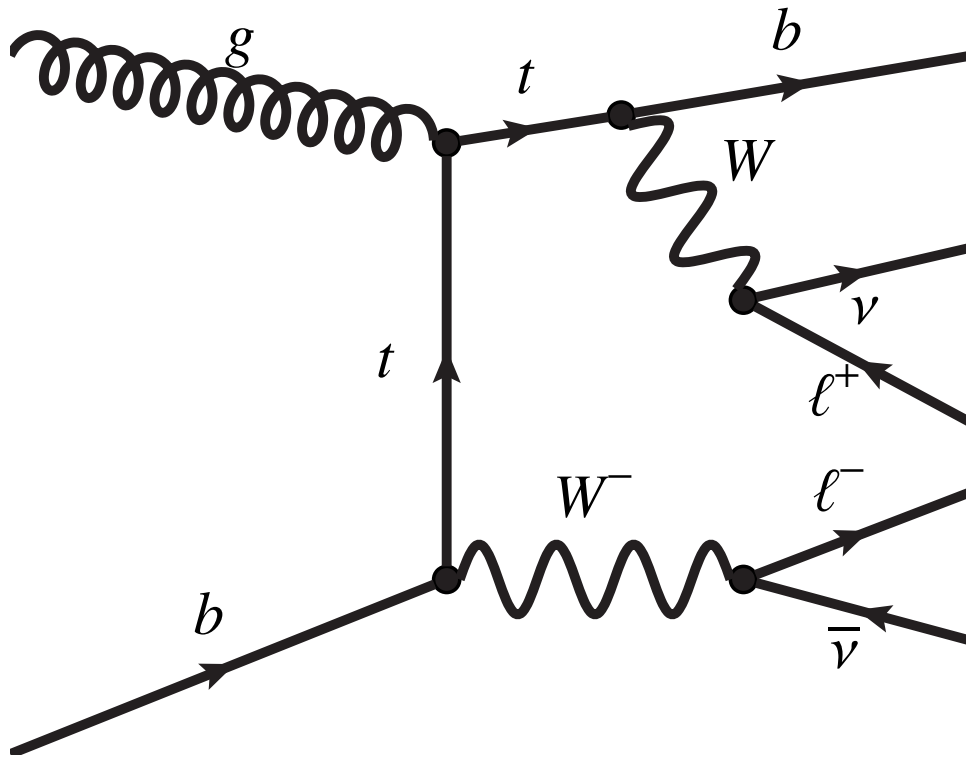


Figure 2.4: Final state of a  $tW$  decay. Both  $W$ -bosons decay leptonically, i.e., in a lepton and the respective neutrino.

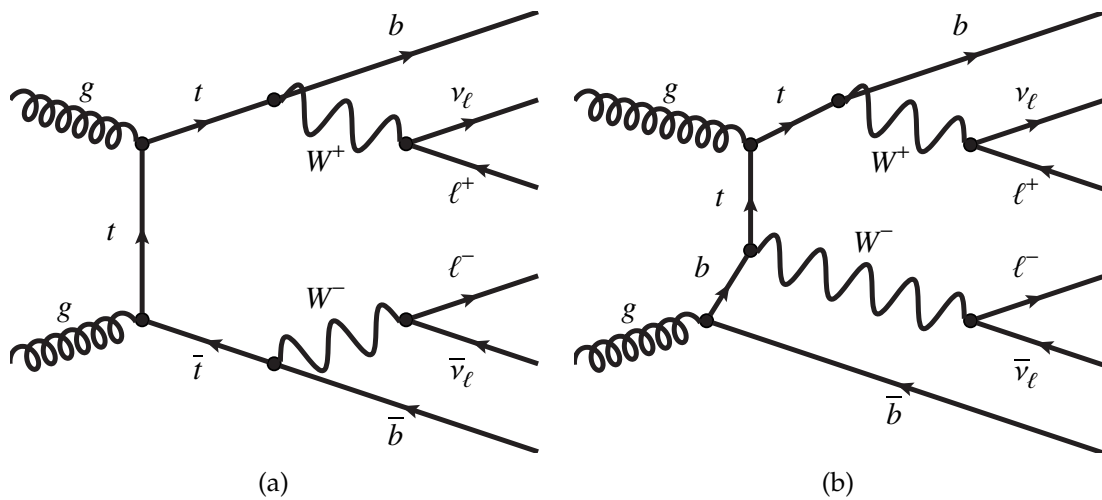


Figure 2.5: Comparison of the final state of a  $t\bar{t}$  event 2.5(a) and a NLO  $tW$  event 2.5(b). Both  $W$ -bosons decay leptonically and the final states are identical.

The second property is the instantaneous luminosity of an experiment which defines how many interactions an experiment can produce per area and time. For the colliding gaussian beams of the Large Hadron Collider it can be defined as:

$$L = f \frac{n_1 n_2}{4\pi\sigma_x\sigma_y} \quad (2.5)$$

where  $f$  is the frequency of the beams,  $n_i$  is the particle number per bunch of protons and the  $\sigma_i$  values stand for the horizontal and vertical beam size respectively. The integrated luminosity is then used to estimate the total number of interactions for a certain period of measurement:

$$\mathcal{L} = \int L(t)dt \quad (2.6)$$

Multiplied with a cross-section the corresponding number of interactions to be expected,  $N$ , can be defined as  $N = \sigma\mathcal{L}$ . For more information about the Standard Model and interactions in particle detectors, see [3, 7].



---

## 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 or more precisely high luminosity, to achieve statistically significant results. Secondly the energy provided has to be sufficiently high to overcome the large binding energy of hadrons.

This work was carried out using simulations based on the ATLAS [8] detector at the Large Hadron Collider (LHC) [9] which offers both a record breaking center-of-mass energy and luminosity. This chapter summarises both machines and provides the knowledge necessary to understand the simulations used. First of all, a summary of the machines' parts and their technical details is given, followed by a description of how the detector detects particles and how their properties are measured. A brief introduction into how detector events are simulated is presented.

### 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 benefiting multiple particle physics experiments.

The LHC is a circular particle collider with a circumference of 26.7 km designed to accelerate and collide two counter-rotating proton beams. The protons are accelerated in bunches of up to  $10^{11}$  protons, at energies up to 6.5 TeV. This way, the record breaking luminosity of  $10^{34} \text{ cm}^{-2} \text{ s}^{-1}$  and center-of-mass energy of up to 13 TeV is achieved. 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 one can refer to the LHC design report [9]. The four main interaction points, at which the beams are brought to collision, contain the main experiments of the LHC. Two of them are general purpose detectors, namely ATLAS [8] and CMS [10]. That means they are designed to cover a wide range of final states rather than focusing on a single analysis. The third is the LHCb [11] which focuses on bottom-quark physics. Lastly, ALICE [12] is 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.

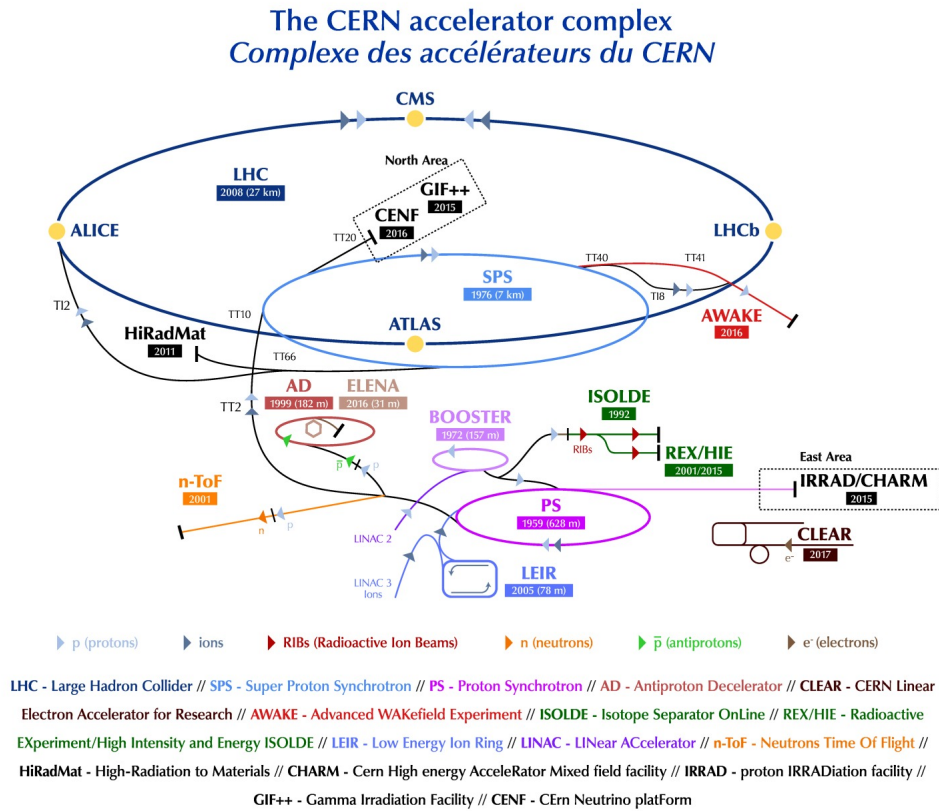


Figure 3.1: Sketch of the LHC accelerator complex showing the acceleration systems and the main storage ring with its experiments [13].

### 3.2 The ATLAS detector

The ATLAS detector is a general purpose detector meaning it aims at covering a maximum number of final states. This allows for many topics of research within the realm of particle physics.

ATLAS, “A Toroidal LHC Apparatus” has the distinguishing structure of a general purpose detector. Its innermost part is formed by tracking detectors directly surrounding the interaction point, followed by calorimeters, and a final layer for muon tracking. All the components are visualized in figure 3.3 including two humans to give an impression of the detector’s size.

The innermost tracking detectors are summarized under the name Inner Detector (ID). The ID consists of two silicon detectors namely the Pixel Detector and the Semi Conductor Tracker as well as a straw detector named Transition Radiation Tracker. It allows for precise measurement of not only charged particles’ position, and thus *vertex* information, but also for their charge and momentum. A vertex is the origin of an interaction and decay determined by the intersection of tracks. It is, however, not able to detect neutral particles.

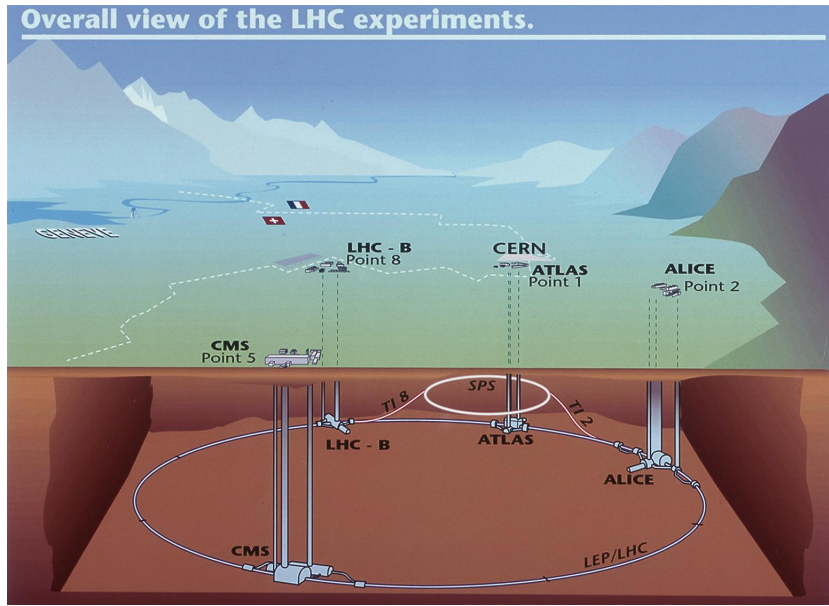


Figure 3.2: Sketch of the LHC ring, the position of the experiments, and the surrounding countryside. The four big LHC experiments are indicated (ATLAS, CMS, LHC-B and ALICE) along with their injection lines (Point 1, 2, 4, 8) [14].

The calorimeter system is divided into two components, being the electromagnetic calorimeter and the hadronic calorimeter. It allows to measure the energy deposited by particles in the detector material.

The Muon Spectrometer is the last tracking detector which identifies particles crossing it as muons, as all other charged particles are usually stopped in the calorimeter system.

In the following sections the detector's components are outlined in more detail [15] to then summarize how particles can be detected and distinguished. At first, an introduction to the coordinate system is given. The reconstruction of objects from the detector response is explained and an event selection for  $tW$  candidates introduced.

### 3.2.1 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,  $\theta$ , is the angle between the  $z$ -axis and the  $x$ - $y$ -plane and the azimuthal angle,  $\phi$ , 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  $\phi$ , the pseudo-rapidity,  $\eta$ , and the transverse momentum,  $p_T$ . The pseudo-rapidity replaces the polar angle and is defined as

$$\eta = \frac{1}{2} \ln \left[ \tan \left( \frac{\theta}{2} \right) \right] \approx \operatorname{arctanh} \frac{p_z}{E}. \quad (3.1)$$

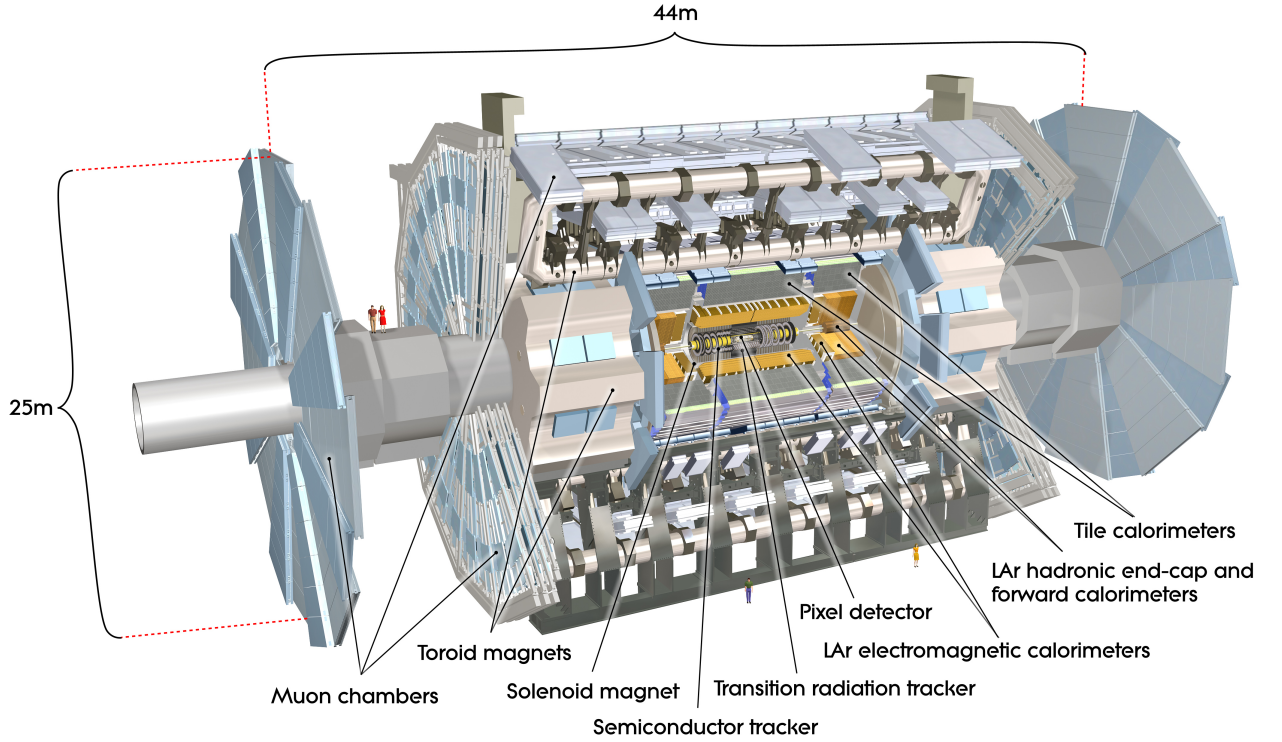


Figure 3.3: Sketch of the ATLAS detector and all its components including two average humans for scale [16].

The transverse momentum is defined as

$$p_T = \sqrt{p_x^2 + p_y^2}, \quad (3.2)$$

where  $p_x$  and  $p_y$  are the momenta along the corresponding axes. The angular variables are defined within

$$\eta \in (-\infty, \infty), \phi \in [-\pi, \pi). \quad (3.3)$$

Lastly the angular separation of two objects can be measured using their  $R$ -value. The angular difference,  $\Delta R$ , is defined as:

$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}. \quad (3.4)$$

The shape of the ATLAS detector makes the cylindrical system the obvious choice. This comes with the additional benefit of a well defined transverse plane where the sum of all vectors in the final state should be zero because the initial state has no transverse momentum.



### 3.2.2 The Inner Detector

Tracking detectors are used to measure a charged particle's trajectory, momentum and charge. Two types of such trackers 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 parts of the ID it holds true that they are surrounded by a magnetic field and cover a pseudorapidity range of  $|\eta| < 2.5$ . The magnetic field results in curved trajectories enabling an estimate of momentum and charge [17].

Pixel detectors are based on the effect of ionisation of semiconductor material when a particle traverses it. The induced charge is picked up by the detector's pixels providing information regarding the particle's position. 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 bottom-quarks, giving the layer its name [18].

The SCT, a silicon microstrip detector, is the second silicon-based tracker immediately following 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 a structure composed of straw tube detectors. Each straw tube is an individual drift chamber with a strong potential difference due to negatively charged walls and a negatively charged wire at the center. The tubes are filled with a gas mixture (Xe or Ar) causing traversing charged particles to ionise the gas. The resulting ions are accelerated to the walls. The drifting particles cause a cascade effect that is then read as a hit related to the time the cascade particles drifted within the tube. Between the tubes, material is inserted resulting in transition radiation. This radiation has a cross section much higher for electrons than for other particles over a wide range of energy thus adding possible electron identification 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 consecutive hits within the ID gives an estimate on the particle's trajectory and can thereby also provide information on the vertex the particle originates from. This vertex information is useful for an experiment with a luminosity as high as that of the ATLAS experiment because events interfere and information from so called *pileup* events can affect the event's information. Pileup events or objects originates from additional proton-proton collisions. If the secondary collision happens at the same time as the original hard scattering the pileup is called *in-time* and the events become superimposed. Additionally, the reaction time of the calorimeters can lead to objects originating from different bunch-crossings to interfere with the event, leading to *out-of-time pileup*. As pileup originates from different events it can be separated from the event of interest by separating the vertices.

### 3.2.3 The ATLAS calorimeter system

The ATLAS calorimeter system is divided into three main parts. The electromagnetic calorimeter (ECAL), comprising a barrel and two end-caps, and the hadron calorimeter (HCAL), built by a tile calorimeter, consisting of 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 traversing particle's energy by exploiting the formation of *particle showers* [15]. 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 ionisation. The resulting charge or photons can be measured as an estimate of the initial energy.

Electromagnetic calorimeters exploit the energy loss of electromagnetically interacting particles in matter. Mainly photons and electrons lose their energy based on pair production and Bremsstrahlung respectively. The energy loss initialises a cascade of particle decays called an *electromagnetic shower* [15]. 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 material. In the absorber the showers are induced to subsequently be detected in the detection layers. The barrel part covers a range of  $|\eta| < 1.475$  and the end-caps consisting of two coaxial wheels cover  $1.375 < |\eta| < 3.2$

As the ECAL uses electromagnetic showers, the hadron calorimeter depends on *hadronic shower* evolution. Hadronic showers are initialized due to ionisation or strong interaction with the material's nuclei. If the resulting particles still interact with the material, a shower ensues. 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 calorimeter (HEC) is liquid argon-based and covers  $1.4 < |\eta| < 3.1$ . Additionally, in the a high-density forward calorimeter covers  $3.1 < |\eta| < 4.9$  Due to the larger size of hadronic showers the HCAL occupies more detector space than the ECAL.

### 3.2.4 The Muon Spectrometer

The second tracking component of ATLAS is the muon spectrometer (MS) 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 depositing their complete energy, and to collect trigger information and information on trajectory and momentum. In order to accomplish these 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  $\phi = 300^\circ$  and  $\phi = 270^\circ$ .

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.

### 3.2.5 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 tells how consistently a track holds hits in every layer. A track offers momentum and electric 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 even to a corresponding track.

In the following, the higher order objects reconstructed from this basic information and 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 and 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 lower the probability to label non-electron objects as electrons [19].

**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 energy deposited in the calorimeters [20]. In addition, there is further information that can be associated to jets: i.e., 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 anti- $k_t$  algorithm was used [21].

**Muon** reconstruction uses MS hits matched with ID tracks. The choice can be further specified by applying an identification cut based on MS/ID agreement and the integrity of MS hit response. As for electrons isolation can be required [22].

**$b$ -jets** are jets originating from the decay of a bottom-quark and therefore a strong discriminant for events containing a top-quark 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 for example clear secondary vertices, vertex alignment of a primary, a secondary  $b$ -vertex and a tertiary  $c$ -vertex, the decay length, and the characteristic energy scale [23, 24].

**Missing transverse momentum** ,  $p_T^{\text{miss}}$ , arises from momentum imbalance in the transverse plane. Momentum in the transverse plane should be preserved due to it being perpendic-

ular to the beam axis and imbalance is an indicator for neutrinos escaping the detector without depositing their energy in the calorimeter system. It is calculated using two contributions: signals from fully reconstructed and calibrated particles, and information from reconstructed charged particle tracks [25].

In addition, the ATLAS trigger system is a crucial component to analysis. 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 is taken into account. Given the incredible luminosity of the LHC, such a preselection is important to minimize the data actually processed by individual analyses and selection schemes. The ATLAS trigger system consists of three triggers: 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 completely hardware-based and its decision making process mostly rests upon 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 software-based and takes the L1 events as input. The L2-trigger defines regions of interest (ROI) as regions in the angular plane where the objects for L1 were detected, and applies a further cut to these objects. 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 applies a first, broad selection.

### 3.3 $tW$ event selection

The process that this work focuses on is the  $tW$ -channel in the dilepton decay mode, and its most relevant background process: top-quark pair-production,  $t\bar{t}$ . Other backgrounds for the  $tW$  channel are reduced by applying an event selection:

- A single electron or muon trigger
- Electrons: tightly identified, isolated,  $E_T > 26$  GeV
- Muons: tight isolation,  $p_T > 26$  GeV
- A pair of leptons with opposite electric charge
- Leading lepton  $p_T > 27$  GeV
- Veto for a third lepton  $p_T > 20$  GeV
- A lepton must match the trigger
- At least one jet with:  $p_T > 25$  GeV,  $|\eta| < 2.5$ , tagged at 77 % working point

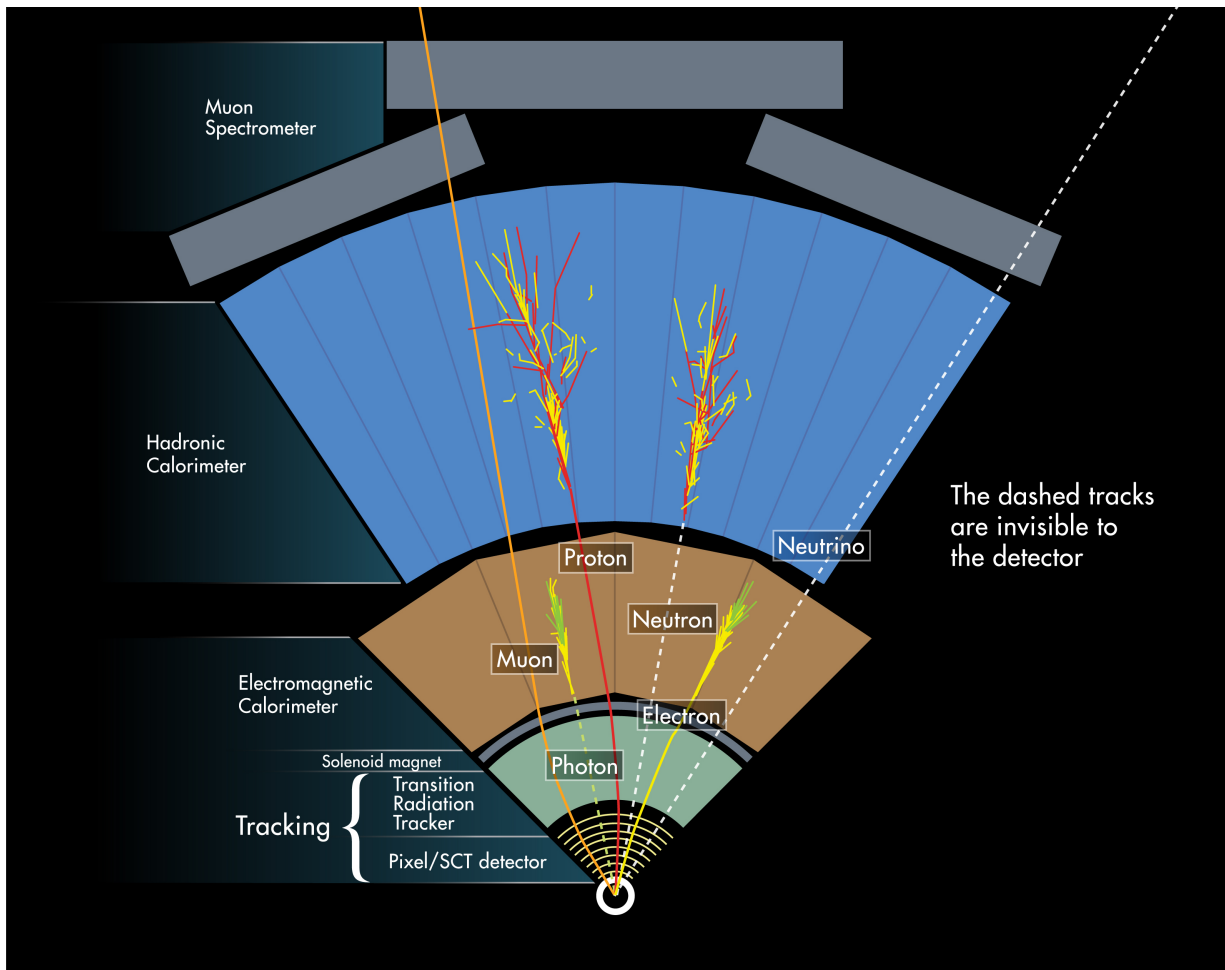


Figure 3.4: Scheme of the ATLAS-detector showing examples of typical particle detections. [16].

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 simulations

A Monte Carlo simulation is a computer based stochastic calculation of a process that in principle could be deterministic. However, the problem and the amount of statistics suggest a stochastic approach.

Monte Carlo, in the context of the ATLAS experiment, is the simulation of physics processes used to calibrate the performance of the detector and to compare model and measurement. As for for this work, the classifying tools and event selection rules are tested and tuned based on simulations to achieve a better understanding of what efficiencies are to be expected in a data analysis. This provides the truth labels to an event that is needed for supervised learning, see section 4, and cannot be provided by data.

A simulation has to be based on quantities that can not only be calculated from theoretical models but also measured during the experiment, because the models should reflect reality in terms of physics processes and detector effects. In collider physics, predictions are mainly based on the cross-section of an interaction stating how high the probability for certain interaction is. In addition, the decay width of a particle is needed to describe how particles generated in an interaction behave in the detector system.

A proton-proton collision is modeled by describing the protons as a sea of partons, i.e., the gluons and quarks their momentum is carried by, and then calculating the cross-sections for these partons to interact. The technique of seeing the collision as an interaction of two individual partons while decoupling these interactions from the parton interactions in the individual protons is called factorization. For a parton carrying a fraction  $x$  of the proton's momentum for a center-of-mass energy,  $\hat{s}$ , the cross-section,  $\sigma(p_1 p_2 \rightarrow X)$ , to create a final state  $X$  can be described as:

$$x_i = \frac{p_{parton,i}}{p_{proton}}, \quad (3.5)$$

$$\hat{s} = x_1 x_2 s, \quad (3.6)$$

$$\sigma(p_1 p_2 \rightarrow X) = \sum_{i,j=q,\bar{q},g} \int dx_i dx_j f_{i/p_1}(x_i, \mu^2) f_{j/p_2}(x_j, \mu^2) \cdot \hat{\sigma}_{ij}(ij \rightarrow X; \hat{s}, \mu^2). \quad (3.7)$$

There are two quantities in equation 3.7 not yet explained. The parton density function (PDF),  $f_{i/p_m}(x_i, \mu^2)$  denotes the probability of a parton with a certain momentum fraction,  $x_i$ , in the proton  $k$  takes part in the hard scattering. The PDF is independent of the collision and has to be gained from experiments rather than through calculations.

The second quantity  $\mu$  is the so called factorization scale. It describes to which degree the interactions of the partons in the proton can be neglected. It is an arbitrary scale determining the precision of the simulation to some degree.

### 3.4.1 ATLAS simulation

The MC simulation for the ATLAS detector is generated in two steps [26]. In the first step the actual collision is simulated, defining the particles in the final state. The underlying algorithm is called Monte Carlo event generator. Secondly, the response to these particles is simulated by the detector simulation allowing objects to be reconstructed from the event similar to actual data gaining a comparison on every level of reconstruction.

Furthermore, a set of different options and generators can be chosen for the simulation. Examples would be how a parton shower is simulated or to which degree multi-parton interaction is taken into account.

### 3.4.2 Systematic uncertainties in Monte Carlo simulations

Simulations, as described earlier, have to be based on certain assumptions about the input parameters. This is especially true in those cases for which the theory does not offer a precise result. This results in systematic uncertainties arising from the simulations, which should be

taken into account when performing an analysis. In those cases different Monte Carlo samples are produced, each representing a different assumption about the initial parameters; showing first of all how the data should behave with these scaled parameters and allowing to test whether the systematic or the original sample fits better to the data. These samples of Monte Carlo will just be referred to as systematics in this work.

Usually the simulations use the leading order diagrams. When next-to-leading order diagrams become relevant, the simulation sometimes has to be adapted resulting in different samples for different orders. This work focuses on the  $tW$  and  $t\bar{t}$  events interfering at NLO as described in section 2.2.3. The total NLO amplitude is the sum of the of singly resonant diagrams and all doubly resonant diagrams:

$$\mathcal{A} = \mathcal{A}_{tW} + \mathcal{A}_{t\bar{t}}. \quad (3.8)$$

Calculating the cross-section requires the square of this amplitude:

$$|\mathcal{A}|^2 = |\mathcal{A}_{tW}|^2 + 2\mathcal{R}\{\mathcal{A}_{tW}\mathcal{A}_{t\bar{t}}^*\} + |\mathcal{A}_{t\bar{t}}|^2, \quad (3.9)$$

$$\equiv \mathcal{S} + \mathcal{I} + \mathcal{D}. \quad (3.10)$$

The Diagram Removal approach is used as a nominal sample, denoted as DR, is used. It fully removes the  $\mathcal{A}_{t\bar{t}}$  amplitude and thus both the pure  $t\bar{t}$  contribution as well as the interference term vanish. The amplitude then becomes:

$$|\mathcal{A}_{DR}|^2 = \mathcal{S}. \quad (3.11)$$

Alternatively the Diagram Subtraction or DS can be used, which is in this work represented by the systematics sample. It introduces a gauge invariant term that cancels the  $t\bar{t}$  contribution and in a simplified way, can be written as

$$|\mathcal{A}_{DS}|^2 = \mathcal{S} + \mathcal{I} + \mathcal{D} - \tilde{\mathcal{D}}, \quad (3.12)$$

$$\approx \mathcal{S} + \mathcal{I}. \quad (3.13)$$

The nominal samples were produced using a full ATLAS detector simulation implemented in GEANT4 and the systematic samples were mostly simulated using ATLF2 [27, 28]. The systematic samples used in this thesis were produced using GEANT4 as a DS sample requires a full simulation.

The events were generated by POWHEGPYTHIA 8 [29].





# Machine Learning

---

## 4.1 The concept of machine learning

Over the last decades, computers have become indispensable tools of science; handling large amounts of data, completing tedious calculations, and controlling sophisticated experiments. For the most part, these machines were assigned discrete tasks and they followed step-by-step commands, designed beforehand by human users, and had expected outcomes. For particle physics in particular, computers have been used to select and process data from large samples, allowing the processing of these data at a speed 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. In contrast, machine learning enables 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. Thereby the machines would be enabled 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. One prerequisite for 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 needs to understand the system, to evaluate a decision step, and finally generate new decision steps. Understanding a system means to be aware of all its features and possibilities. Humans have their senses to easily break down observations into useful features and concepts that can then be processed for finding a decision. A computer has no such senses, 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

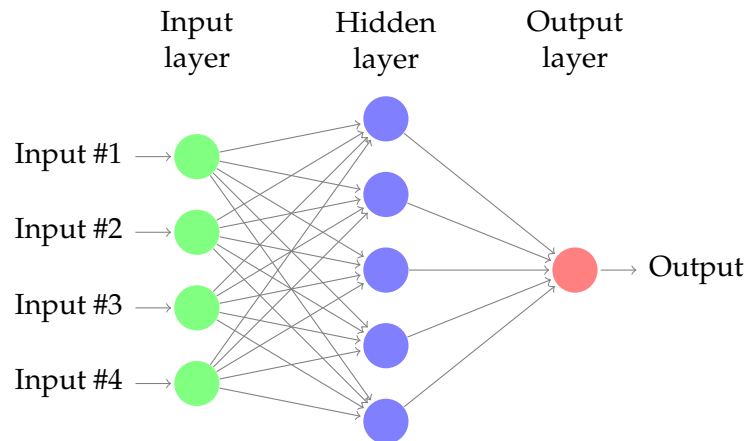


Figure 4.1: Sketch of the typical neural network structure. In this example only one hidden layer is included and the output is a single node [30].

metric is introduced that measures the quality of the model, called the *cost* or *loss function*. This function allows for iterative improvement 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 incompletely investigated topic.

A very commonly used machine learning technique is the *artificial neural network* which, on its own forms, an extended field that builds the base of this work. The essential concepts of machine learning will be explained in the context of neural networks.

## 4.2 Neural Networks

The artificial neural network, or just neural network, is one of the most commonly applied approaches to machine learning. Its structure and naming is inspired by the neurons forming the human nervous system.

Instead of living neurons, a neural network consists of numerous very simple processors, called nodes. These nodes are usually structured into several layers as presented in figure 4.1. In addition, there are several ways to structure and connect these nodes, frequently matching a certain problem. In this explanation, only the most commonly way is explained. In that case each node of a middle layer is getting input from each node in its predecessor and is outputting to each node of the following layer. This obviously is only true in one direction for the extreme layers, input and output. A simple view of communication between nodes is shown in figure 4.2 describing the step between a node and the previous layer. This is called a feed-forward neural network, the math of which will be described in detail later.

### 4.2.1 The input layer

To understand a task and draw reasonable conclusions, the underlying system has to be understood at first, which means its features need to be determined and summarised. The

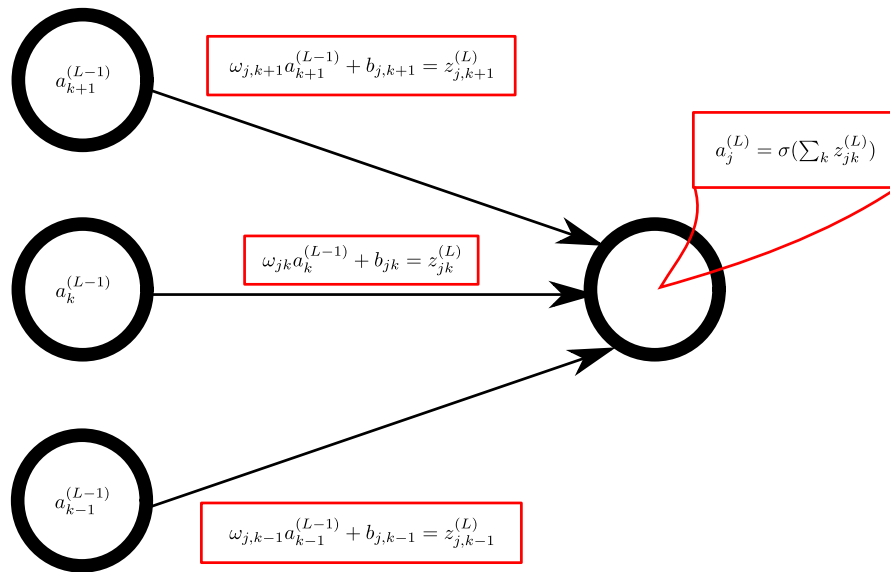


Figure 4.2: Network propagation from layer  $(L - 1)$  to layer  $L$ . The linear functions and the nomenclature of the parameters are indicated.

human brain is capable of investigating unknown systems and learning the features that are the most unique or interesting ones. For that, the nervous system uses 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 usually is the task that requires 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 reconstructed objects in an event.

The input to a neural network is given to the input layer of nodes and subsequently processed through each following layer. For diverse tasks, different layers might deal with various parts of the information. However, in this work the linear way of giving all information used to an input layer and then processing it is used.

#### 4.2.2 Decision making process

Computers representing nothing more than very powerful calculators, excel at performing high numbers of clearly defined calculations. This requires a precisely elaborated task containing no uncertainty. This is completely different for the human nervous system, which relies on a certain uncertainty when processing information through a net of neuron cells. In this net, 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 a neural network, the neurons and their fuzzy interactions are represented by the nodes. Like neurons, each node can use input from many other nodes to create a new output signal. Thereby the sets of input information can be linked to each other in numerous ways. Combined with a weighting system, this allows one to create complex models and match a variety of problems.

$$z_j^L = \sum_{k=0}^N \omega_{jk}^L a_k^{L-1} + b_k \quad (4.1)$$

The input of every node is the weighted output of all previous nodes, as shown in equation (4.1).  $z_j^L$  is the input to the  $j$ -th node in the  $L$ -th layer.  $\omega_{jk}^L$  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.

The weight allows a network to predict which variables are correlated or allow for better decision rules when combined. In addition, the weights can just define the strongest variables and 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. This is called the *activation-function* of a node. A common choice is the sigmoid function as presented in equation (4.2) [31].

$$a_j^L = \sigma(z_j^L) = \frac{1}{1 + e^{-z_j^L}} \quad (4.2)$$

The sigmoid function has an output between 0 and 1, which is frequently desired 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 4.1.

For this thesis, mainly the exponential linear unit, or *elu*, and the rectified linear unit, or *relu*, were tested. *Elu* is suitable for converging the cost to zero relatively fast and provides the possibility of negative output while *relu* allows for the same benefit as sigmoid but requires 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 model in each step. Following the process of cost minimisation, the solution of the initial problem can be approximated. 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 labeled data. Each event in the training set has is assigned a label representing what process the network is looking at. This is referred to as *truth label* or just *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 known label. 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 of loss function. Equation (4.3) shows the underlying function where  $p$  is the estimated probability for the prediction  $\hat{y}$  and  $y$  is the truth label.

$$C = -(y \log p + (1 - y) \log(1 - p)). \quad (4.3)$$

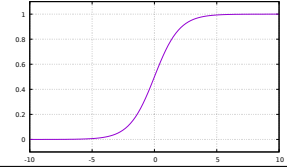
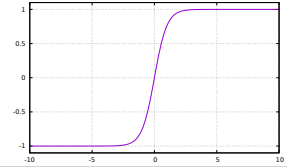
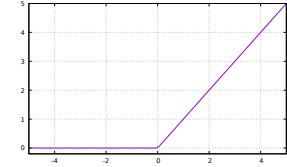
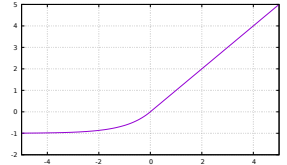
Name	Function	Plot
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	
Hyperbolic tangent	$f(x) = \frac{2}{1+e^{-2x}} - 1$	
Rectified Linear unit, RELU	$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	
Exponential Linear Unit, ELU	$f(x) = \begin{cases} \alpha(e^x - 1) & \text{if }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

Table 4.1: Selection of activation functions taken from the Keras documentation [31].

The loss is the most important indicator for the training quality. This training quality must not be mixed up with the overall quality of generated model. Only after a test on a different sample or real data the model can be fully evaluated.

### 4.2.3 Optimisers - 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. These training steps are referred to as *epochs*. One could certainly do this randomly until the network finds a very low costed decision rule if infinite computational power was 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, there are three variables that have impact on the loss, the weight, the bias, and the activation. 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 preferred 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.
3. The gradient of the loss is calculated as the partial derivative of all network parameters using the following equation:

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

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 are different choices of optimisers, which try to accommodate different problems as well as some important parameters to 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,  $m$ , stands for the amount of data processed to evaluate the next step.  $f$  is the network for a current configuration or model,  $\theta$ , and the prediction  $\hat{y}$ .  $\theta$  summarises all the parameters optimised by the network during the training. The truth label is  $y$ . Using these definitions the gradient becomes:

$$g = \frac{1}{m} \nabla_{\theta} \sum_j L(f(\hat{y}^j; \theta), y^j). \quad (4.4)$$

The configuration  $\theta$  is then updated using the gradient and a multiplicative constant,  $\eta$ , called the *learning rate*, which determines the step size for each update following:

$$\theta' = \theta - \eta g. \quad (4.5)$$

Optimisation processes like these 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 only learning rate as its hyperparameter. A good learning rate should be small enough to avoid oscillations around minima but high enough to approach a minimum efficiently. A good estimate is given by the Robbins Monro condition:

$$\sum_k \eta_k = \infty, \quad (4.6)$$

$$\sum_k \eta_k^2 < \infty. \quad (4.7)$$

As the choice of learning rate will not be perfect for every part of the problem's topology, *momentum*,  $\nu$ , can be introduced as a second parameter to the optimiser [31]. The effect desired is twofold; momentum should increase the learning rate in the direction of the minimum

and lower it when approaching said minimum. Momentum scales each step by how aligned previous steps were, meaning it will allow avoiding local minima or moving slowly along a slope. This is accomplished by enlarging steps at the beginning of the training but diminishing them at the end close to the minimum. It promises to speed up the training with less risk of large oscillations, which is an effect of high learning rates. Momentum also takes a single scaling hyperparameter  $\alpha$  and is updated each step in the following way:

$$v' = \alpha v - \eta \frac{1}{m} \nabla_{\theta} \sum_j L(f(\hat{y}^j; \theta), y^j), \quad (4.8)$$

$$\theta' = \theta + v'. \quad (4.9)$$

Alternatively one can use *Nesterov momentum* [31], which is a more advanced adoption of momentum as it updates the step a further time after applying the gradient:

$$v' = \alpha v - \eta \frac{1}{m} \nabla_{\theta} \sum_j L(f(\hat{y}^j; \theta + \alpha \times v), y^j), \quad (4.10)$$

$$\theta' = \theta + v'. \quad (4.11)$$

Finally, it can be helpful to decrease the learning rate of the network stepwise while approaching a minimum to avoid oscillations or even missing the minimum completely. This can be accomplished by the hyperparameter of learning rate *decay*. It simply decreases the learning rate in each iteration,  $t$ , by a small hyperparameter,  $\phi$ , following the assumption that smaller steps are sufficiently close to the minimum [31]. The learning rate is then defined as:

$$\eta' = \frac{\eta}{1 + \phi t}. \quad (4.12)$$

#### 4.2.4 Adaptive optimisers

In addition to the merely 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. From all of the adaptive optimisers [31] *Adam* is probably the most popular [32]. *Adam* updates both its learning rate and momentum over the course of the training based on an exponentially decaying average of past gradients and 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  $\beta$  resulting in this gradient definitions:

$$\hat{g}^2 = \frac{\sum g^2}{1 - \beta_1^t}, \quad (4.13)$$

$$\hat{g} = \frac{\sum g}{1 - \beta_2^t}. \quad (4.14)$$

The model's parameters are then updated according to:

$$\theta' = \theta + \frac{\eta}{\sqrt{\hat{g}^2 + \epsilon}} \hat{g}. \quad (4.15)$$

*Adam* is often considered an excellent algorithm as it contains many corrections to hyperparameters during the training, and thereby allows for easier optimisation. However, it also needs more computational power.

### 4.3 Regularisation and Optimisation

Fluctuations and noise in the training sample can be a big problem for a model trained on the sample. A neural network might pick noise and random fluctuations up as features of its decision rule which 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 fraction of the sample used by the training. The most extreme scenario is that the network is wide 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 and thus becomes a mask of the sample.

A possible way to solve this issue is stopping the training early or finding an elaborate estimator to stop on. This way, noisy features would not yet be part of the discriminant. This might in turn also lead to a suboptimal result of the overall training as one cannot be sure that the correct features always get included 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 following subsection. Additionally, a *batch normalisation* can have an effect of regularisation and is therefore introduced in this section as well.

#### 4.3.1 Dropout

Dropout can be described as an additional layer which attempts to hinder the network from relying on less dominant features. In essence a dropout layer removes different nodes in each iteration. This forces the network to build models that are not based on strong correlations between nodes, making the weights less interdependent. In short, it means training several neural networks depending on which nodes are turned on during a training epoch which it keeps the training in motion for a high number of epochs. Figure 4.3 sketches the process.

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



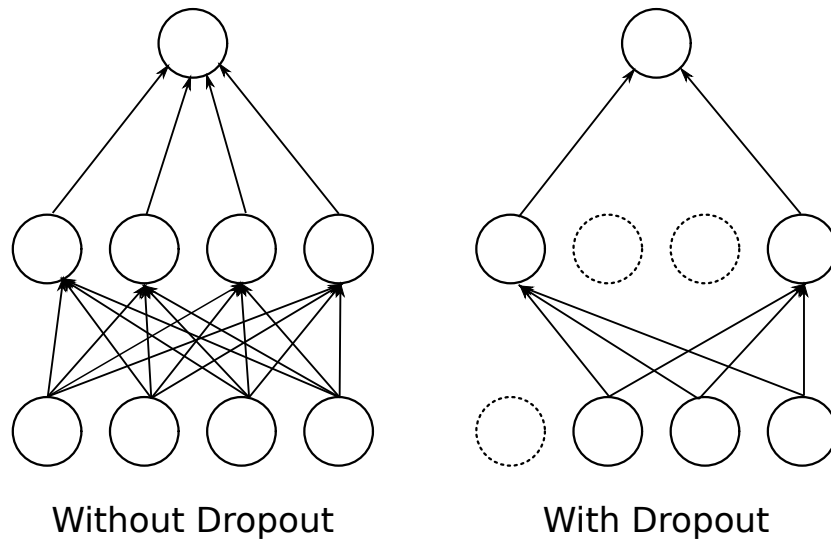


Figure 4.3: Sketch of network before and after the inclusion of dropout. On the left hand side dropout is not applied and all nodes are connected. On the right hand side the dashed circles are nodes excluded by dropout and therefore not connected to the other nodes.

### 4.3.2 Batch normalization

In supervised learning, the training result is strongly dependent on the set of data the network is trained on. This means that the performance might change dramatically when the test data is very different. one can imagine a classifier distinguishing between images of cars and apples. If the training set contains predominantly green cars, the colour green might become a strong indicator for the classification “car”. In general, the colour is not be a defining property of a car 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 general output and connections between nodes in a neural network is not necessarily limited, allowing for certain connections to be extremely dominant and overshadowing other features. This is to be avoided 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  $\mu_B$  and the mini-batch standard deviation  $\sigma_B^2$ . The normalised output,  $x_{i,norm}$ , then becomes:

$$\mu_B = \frac{1}{m} \sum_i x_i, \quad (4.16)$$

$$\sigma_B^2 = \frac{1}{m} \sum_i (x_i - \mu_B)^2, \quad (4.17)$$

$$x_{i,norm} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}. \quad (4.18)$$

## 4.4 Receiver operating characteristic curve

The receiver operating characteristic (ROC) curve is a graphical plot for a binary classifier. The y-axis shows the true positive rate (TPR); i.e., the percentage of correctly assigned labels of value  $\hat{y}$ . The x-axis shows the false positive rate (FPR) and represents the percentage of wrongly assigned  $\hat{y}$  labels. The ROC curve can be used to evaluate the quality of a model generated by a classifier. A diagonal curve means the model is as good as a random guess. Curves below the diagonal are worse than a random guess and can indicate that something is going wrong in the training process. A good measure is the *Area Under the Curve* (AUC). The AUC indicates the overall quality of a model.

## 4.5 Adversarial Neural Networks

The main part of this work is the examination and training of an adversarial neural network. An adversarial neural network consists of a classifying network 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 [33] are presented. For more information about an approach directly tested on physics, one can refer to "Learning to Pivot with Adversarial Networks" [4].

### 4.5.1 The adversarial neural network

Neural networks have been very successful in classification tasks but not so much for generative tasks. This was the original problem that gave birth to the idea of a generative adversarial network. Unfortunately generative networks often produce 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 as this adversary is able to accomplish this task, the first network fails at its generative task. Training the two networks against each other disincentivises 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 real data. If the classifier has these strong dependencies on systematic uncertainties it might lead to a high covariance 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 strongly the model depends 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 render 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. The classifier is referred to as *Net1* and the adversary as *Net2* and the problem becomes:

$$\min_{Net1} \max_{Net2} V(Net1, Net2) = \mathbb{E}_{x \sim \rho_{data}} [\log Net1(x)] + \mathbb{E}_{z \sim \rho_{sys}} [\log(1 - Net2(z))]. \quad (4.19)$$

$V(Net1, Net2)$  is the combined value function for the two adversary networks. The first network is trained to be an optimal classifier, called classifier, represented by  $\log Net1(x)$  while the second network, called the adversary, 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. A combined loss function is used as the value function. It is just the difference between the two separate loss functions with a hyper-parameter,  $\lambda$ , to control the impact of the adversary as shown in equation (4.20):

$$\mathcal{L} = L_{net1} - \lambda L_{net2}. \quad (4.20)$$

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

For this thesis, the adversarial network is set up 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 4.4 shows a sketch of the setup. The hyperparameter  $\lambda$  is set to tune the impact of the second network on the model. A high  $\lambda$  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 6, 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 4.5 shows an example for these loss functions taken from the paper "Learning to Pivot with Adversarial Networks" [4].

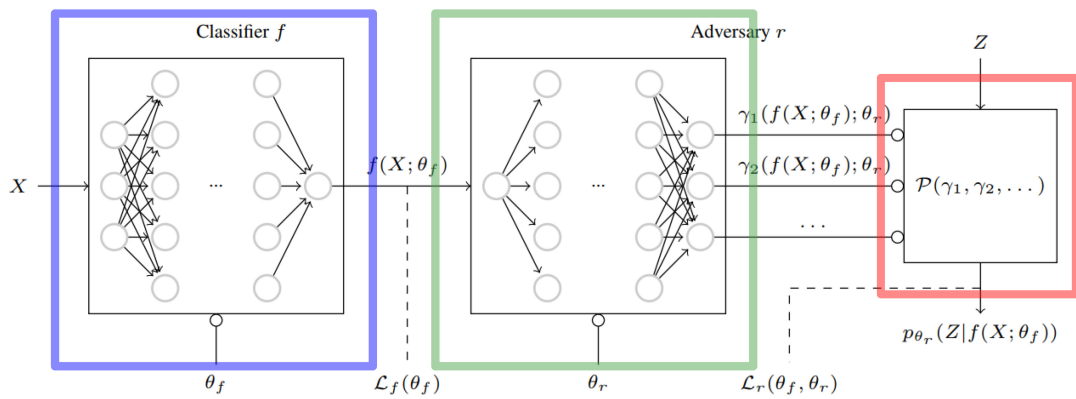


Figure 4.4: Sketch of the adversarial network setup. The classifier is shown in blue. Its output is fed into the adversary, indicated by the green square. The dependency on the nuisance parameter and the combination is shown by the red square. This way the two losses  $L_f$  and  $L_r$  are generated. Furthermore the training of the classifier immediately affects both output models. Sketch based on [4].

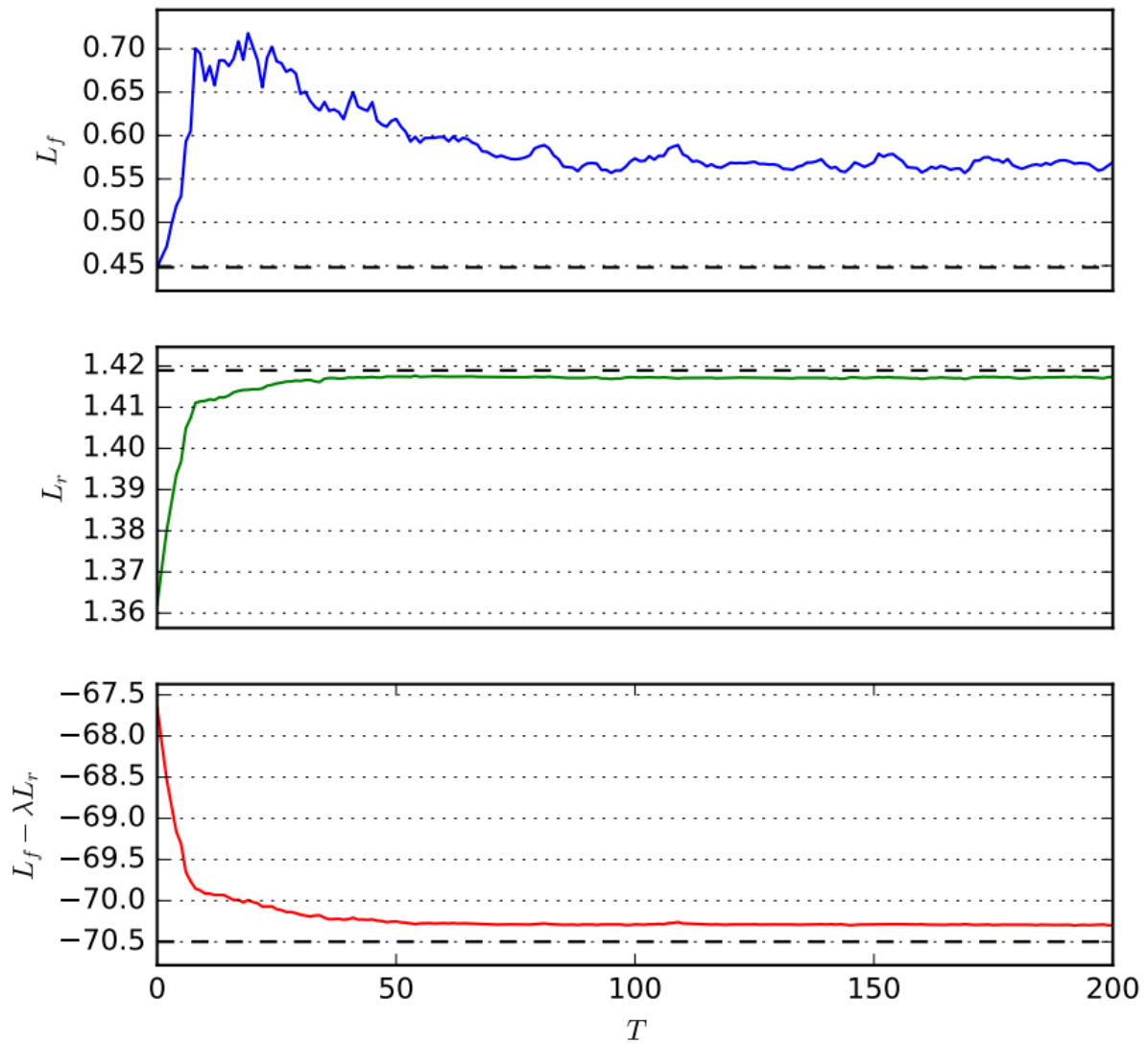


Figure 4.5: Three losses of an adversarial network for  $\lambda = 10$  taken from a toy example [4]. From top to bottom the classifier, adversary and combined loss is presented.  $T$  is the number of training iterations.



---

## Hyperparameter optimisation of a classifying neural network

---

The adversarial neural network is based on a common classifying neural network, which is trained on signal/background separation. The output of that classifier is then used as input for the adversary to create a negative feedback for the classifier. Before the adversarial, second network is added, the classifying network is optimised on its own to ensure 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 that is less sensitive to systematic uncertainties.

This chapter describes the hyper-parameter optimisation of the first network starting with the motivation of the input information. The second section explains the choice of the architecture followed by the step-wise setup of the optimiser. Lastly regularisation of the network is tested and described. The overall aim of the description is to provide some understanding on the hyper-parameters available and their correlations. The impact of the hyper-parameters on the training results is presented not only to introduce their function but also to motivate possible solutions later on.

To train the network and test its performance, all its parts need to be in place. Therefore, all hyper-parameters had to be initialized with values assumed to represent a reliable setup. Initially, this was accomplished by starting with a very simple network using a minimal set of hyper-parameters which were then developed to a more and more optimised network. The results shown in this work are going to be based on the final choice of hyper-parameters, where only one parameter is then varied at a time to explain the impact of a particular tuning. Sometimes, this leads to the effects of a hyper-parameter to be less powerful. The already optimised model is relatively stable, and, if no significant impact is visible, the plots are not shown. For that reason, section 5.2 will already introduce the final network structure, which is then explained and motivated step by step.

The structure was achieved within reason and the computational power constraints. There certainly are hyper-parameters that deserve more attention, and in addition to that, there are alternative setups for the whole network that were not tested. A further investigation of network optimisation is, without a doubt, very promising.

## 5.1 Technical details

The artificial neural networks in this thesis were created using the Keras python library [31]. Keras is an application programming interface written in python and able to run on Tensorflow [34], CNTK [35] or Theano [36]. It was developed by google and summarises the necessary calculations for running a deep neural network training in fast and easy modules. The backend is the package responsible for the underlying vector calculations needed for the network setup and training. In this work, the Tensorflow package was used as a backend.

## 5.2 Final setup of the network

This section describes the fully optimised classifier which is then, parameter-wise, varied during this chapter to motivate the final network structure. The hyperparameters are listed and the loss curve, the ROC curve, and the final separation are shown in figure 5.1. They are the main tools a training performance is evaluated on in this thesis. For the sake of completeness, the agreement between nominal and systematic response is also shown as it will be a standard inclusion for the plots during the adversarial training.

- Input: 14 variables motivated by a BDT variable scan [37].
- Hidden layers: 6 ELU layers  $\times$  128 nodes each
- Output layer: 1 SIGMOID node
- Optimisation: SGD, learning rate = 0.06, momentum = 0.3, no nesterov, no decay
- Duration: 600 epochs

## 5.3 The input variables

Two sets of input variables were tested for the classifier. The first one is a set of simple kinematic variables. This is tested to exploit a neural network's ability to deduce all further information from the complete basis of a system. The second set of variables uses more complex variables based on the most significant variables for a boosted decision tree usage on a similar problem. The variable sets used were: Figure 5.2 shows a comparison of the separation and the loss curves for both variable sets. It is visible that for the set of simple variables the network runs into overtraining after about 200 epochs visible in plot 5.2(d) because  $Loss_{train}$  and  $Loss_{test}$  start to diverge. This also results in a bad separation with a strong disagreement between the two samples. One way overtraining could be avoided using simple kinematic variables would be to stop earlier; in this case after about 200 epochs. There is an argument for testing a simple set of variables for a shorter training duration. However, this optimisation process would be a project of its own because all the other parameter would have to be optimised accordingly. Furthermore, in this work one has to keep in mind that the network is supposed to be used in adversarial setup later. This demands for a longer training period and a large number of



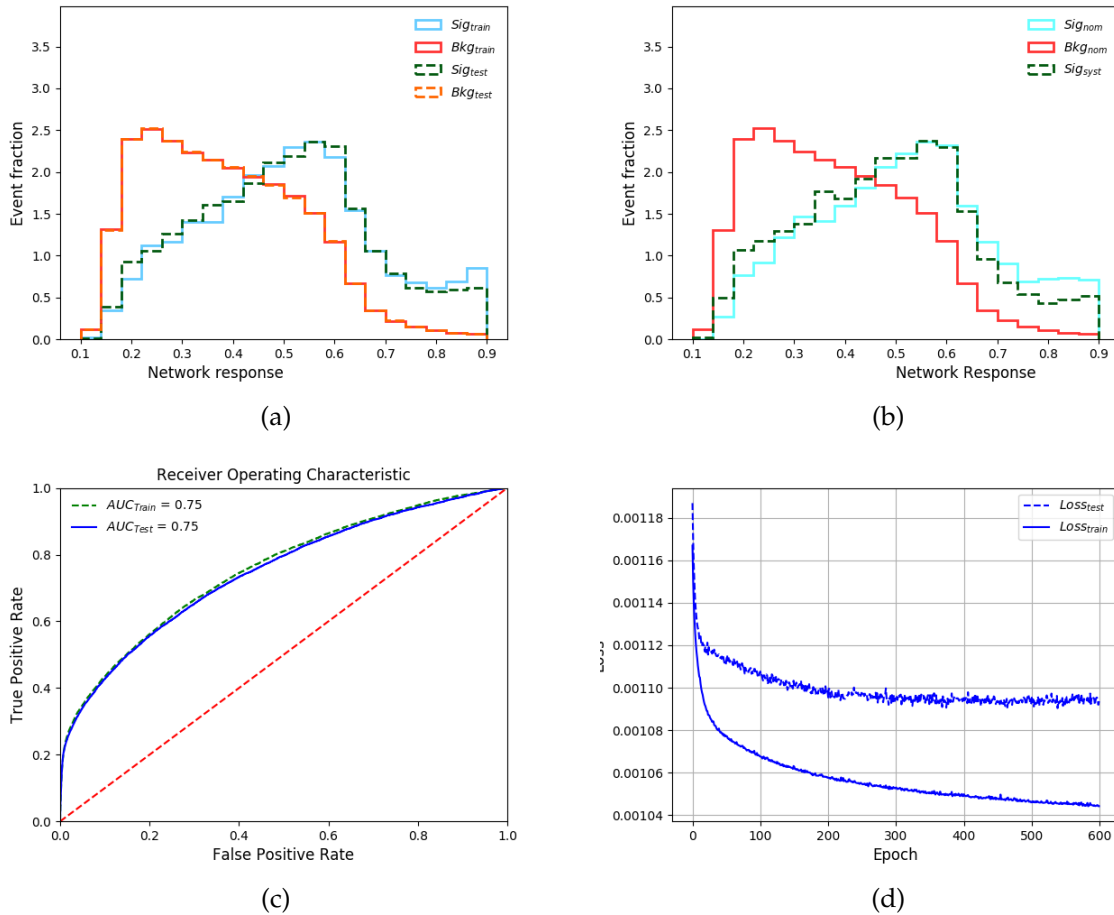


Figure 5.1: Figure (a) shows the response of the classifier. The solid lines represent the training sample and the dashed lines the test sample. Figure (b) shows the difference for response in systematic and nominal samples. The solid lines represent the nominal samples and the dashed line the systematic sample. Figure (c) shows the ROC curve. The dashed line is the training curve and the solid line is the test curve. Figure (d) diagrams the losses with dashed for test and solid for training.

Variable
$p_{\text{T}}^{\text{miss}}$
$ \eta_{\text{jet1}} $
$ \eta_{\text{jet1}} $
$ \eta_{\text{lep1}} $
$ \eta_{\text{lep2}} $
$p_{\text{Tjet1}}$
$p_{\text{Tjet2}}$
$p_{\text{Tlep1}}$
$p_{\text{Tlep2}}$
$\phi_{\text{jet1}}$
$\phi_{\text{jet2}}$
$\phi_{\text{lep1}}$
$\phi_{\text{lep2}}$

Table 5.1: Simple kinematic variables

Variable
$m_{\text{lep1}} + m_{\text{jet2}}$
$p_{\text{Tlep1}} + p_{\text{Tlep2}} + p_{\text{T}}^{\text{miss}}$
$p_{\text{Tjet1}} + p_{\text{Tjet2}}$
$m_{\text{lep1}} + m_{\text{jet1}}$
$p_{\text{Tlep1}} - p_{\text{Tjet1}}$
$R_{\text{lep1}} - R_{\text{jet2}}$
$R_{\text{lep1,lep2}} - R_{\text{jet2}}$
$m_{\text{lep2}} + m_{\text{jet1}}$
$p_{\text{Tjet2}}$
$R_{\text{lep1}} - R_{\text{jet1}}$
$R_{\text{lep1}} - R_{\text{jet1}}$
$R_{\text{lep2}} - R_{\text{jet2}}$
$\text{Centrality}_{\text{lep2}} + \text{Centrality}_{\text{jet2}}$
$R_{\text{lep2}} - R_{\text{jet1}}$

Table 5.2: Complex variables

features, with a strong separating potential, to be available. Otherwise the probability of finding a classifier, less sensitive to a systematic uncertainty, decreases and the behaviour of the losses has to supervised very carefully.

Showing an overall good performance, the set of complex variables, as suggested by the boosted decision tree variable scan, performed in the analysis “Measurement of the cross-section for the production of a  $W$  boson in association with a top quark at 13 TeV” was chosen [37]. Please note that the set of variables in the paper referred to does not always agree because the set used in this work was just inspired by it.

## 5.4 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 aspects 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 leads to a good estimate. Another possible starting point is the amount of variables necessary to fully describe a system resulting in the minimum variable number necessary to input in the network. This also gives a first estimate on how large the architecture should at least be. Both the depth and the overall size of the model play a role. A simplified way of explaining these two properties is by saying that the depth defines how often the input is processed while 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 would learn training set related features

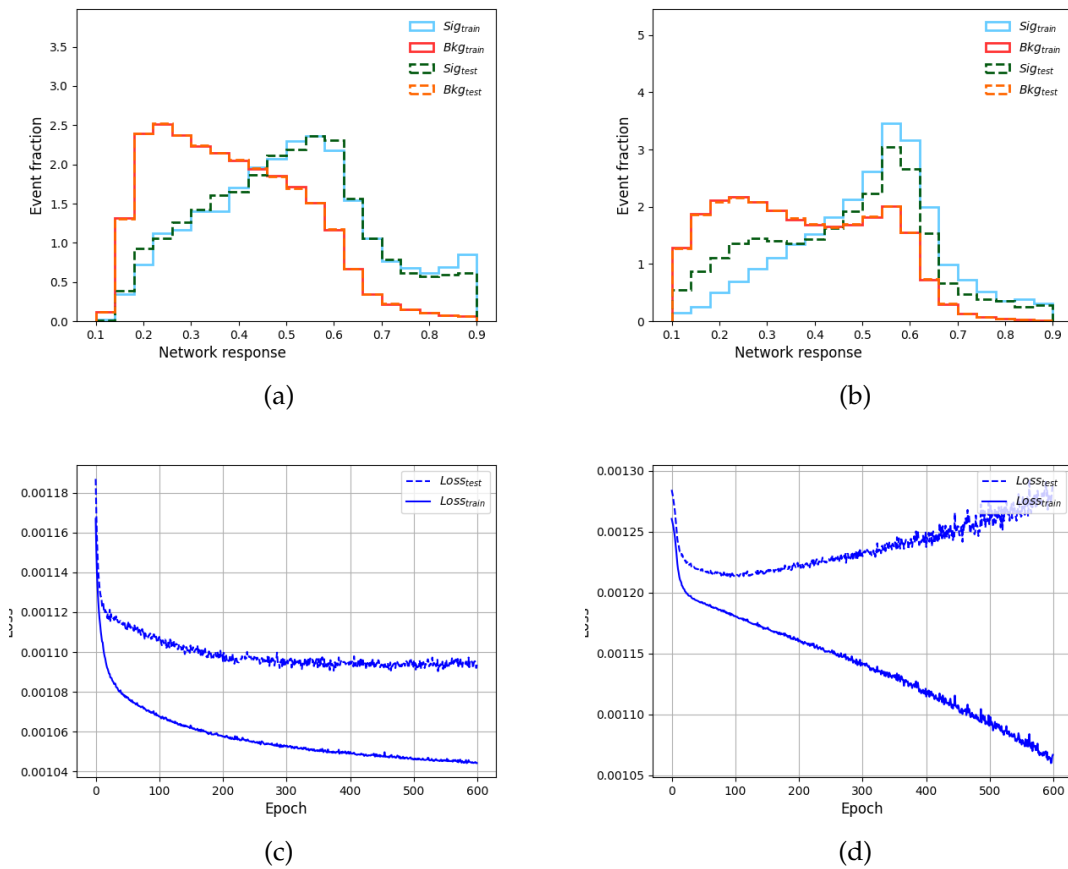


Figure 5.2: Figure (a) and figure (b) show the separation for the complex and simple variables respectively. The losses for the variable sets are depicted in figure (c) and figure (d).

too fast, meaning it overtrains before it finds a good minimum. This can be seen in an early divergence between the training loss and the test loss. An architecture too simple is not able to pick up the features of the task at all resulting in no learning. In other words, the loss stays constant or changes very slowly.

In this work, a test-series was performed. The network was trained for a wide range of combinations of nodes and layers; nodes  $\in [8, 512]$  and layers  $\in [1, 10]$ . 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 test 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. However, the plots do allow for educated guesses for a good choice of architecture.

Figure 5.3(a) shows that more complex architectures also achieve smaller loss values. This, unfortunately is not a clear estimator for overall performance since heavily overtrained networks will achieve low loss values as well. Therefore, a second estimator was taken into account.

Figure 5.3(b) shows the minimal difference between  $Loss_{train}$  and  $Loss_{test}$  achieved during the whole training process. There are two regions showing the minimum difference. The first region includes simple architectures. This, however, is a result of a slow learning process. In this case, the respective overall loss is not minimal and the network's separation power is very low. The second region, with a complexity of 500 to 800, was used as the region of choice; before heavy overtraining occurs. As a result, a more complex architecture in this region, comprising 6 layers with 128 nodes each, was chosen. Although this choice seems fairly arbitrary at this point, there is arguments for a complex architecture as it can always be regulated using dropout if need be. Additional prove that further increasing the architecture leads to overtraining can be seen in the loss for a complex architecture in figure 5.4. Here the overtraining becomes visible in the difference between the two curves and their overall odd behaviour. Here the two loss functions are not only diverging but the test loss also shows non-linear behaviour around 400 epochs. This is usually not expected because once a feature only relevant for the training set has been included it stays problematic.

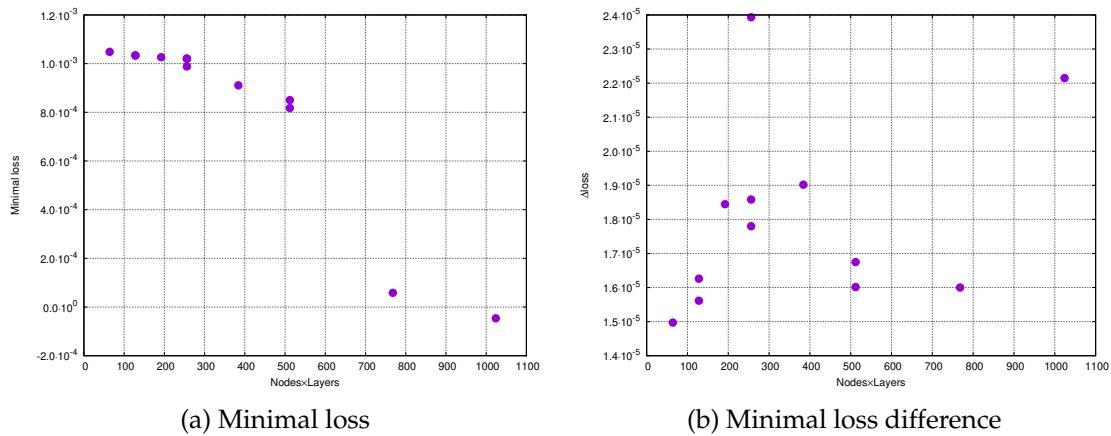


Figure 5.3: Figure (a) shows the minimal overall loss achieved during the training for the range of architectures. Figure (b) shows the minimal difference between  $Loss_{train}$  and  $Loss_{test}$  achieved.

In addition to the choice of the architecture, the activation function for the layers has to be set. Two different activation functions are used in the neural network. The main function connects the nodes in the hidden layers while the last one converts the output to a value between 0 and 1. For the output the sigmoid function is a natural choice because we look at two possible labels for the outcome: signal and background. Therefore, no other activations were tested. For the hidden layers ELU and RELU were tested. Figure 5.5 shows a comparison of the ROC curve and the loss development for both activation-functions. For RELU, a strong disagreement between training- and test-sample occurs and ELU became the activation function of choice. Further activation-functions were not tested due to time constraints.

## 5.5 Setup of the optimisation

For the most part of the optimisation, SGD was used and its tuning is shown in detail in subsection 5.5.2. Adam was also tested but not fully tuned, due to constraints on time and

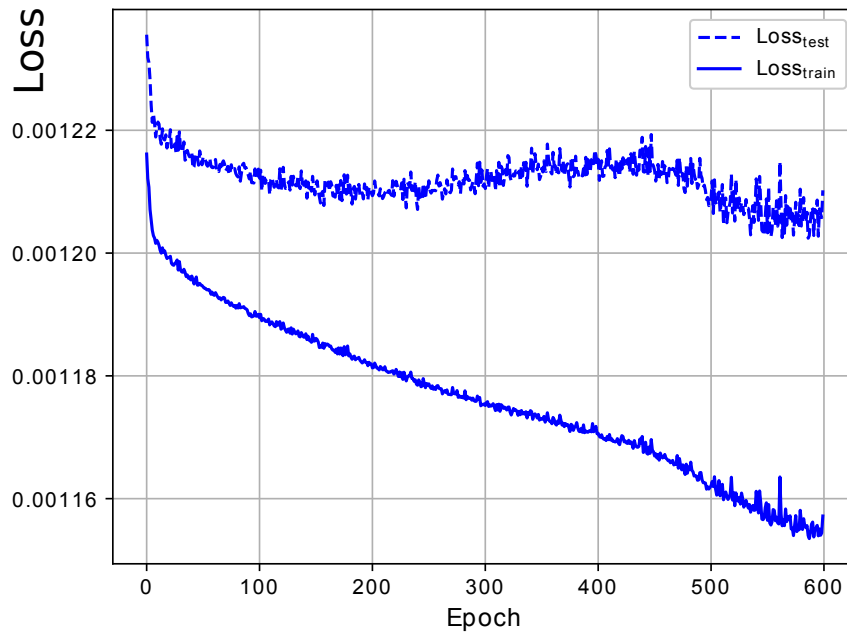


Figure 5.4: Loss function for the classifier for 8 layers and 512 nodes each. A large disagreement between the two curves is visible which is a sign for overtraining.

computational resources. In subsection 5.5.1, a brief overview of the performance using Adam is given.

### 5.5.1 Choice of the optimiser

A classic gradient-based optimiser, SGD, is compared to the adaptive optimiser, Adam. Figure 5.6 compares the loss development for both optimisers. It is obvious that Adam behaves strangely and it was therefore discarded. It is possible that Adam could outperform SGD for a better tuning of its hyper-parameters and it certainly is worth testing out if the computational power is provided.

### 5.5.2 Tuning the optimiser

As described in section 4.2.3, an optimiser has several hyper-parameters of its own. In this section, the learning rate, the momentum with the option of Nesterov, and the decay parameter will be probed. For the learning rate, a range of values between a small learning rate of 0.001 and high learning rates up to 0.5 were tested to investigate the effect on the network's ability to approach a minimum. The choice was based on how high the oscillations of the losses are and how efficiently the training converges. High learning rates were discarded for an increased probability of overtraining and a lot of unwanted oscillations while small learning rates made the training unnecessarily slow and inefficient. Figure 5.7 shows a comparison

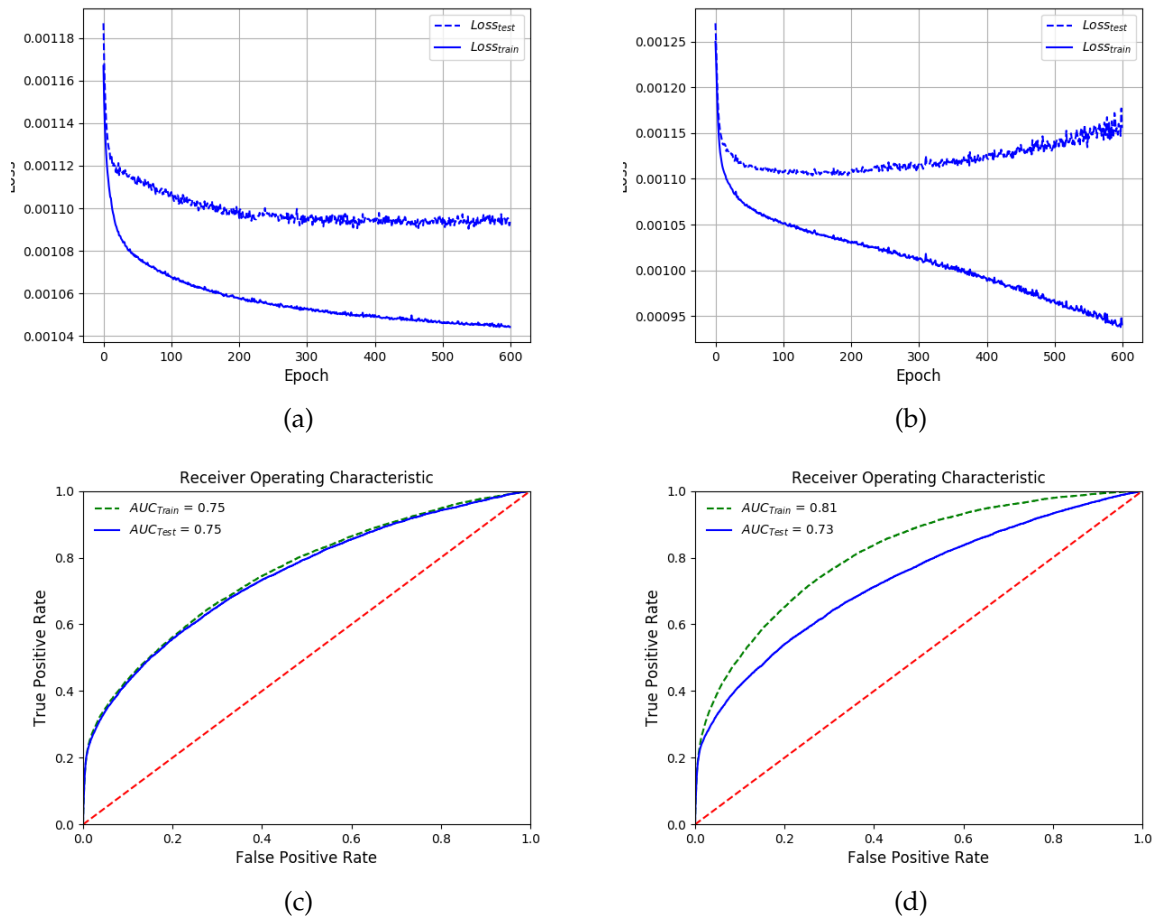


Figure 5.5: Figure (a) and figure (b) show the losses for ELU and RELU respectively. The ROC curves for the different activation-functions are diagrammed in figure (c) and figure (d).

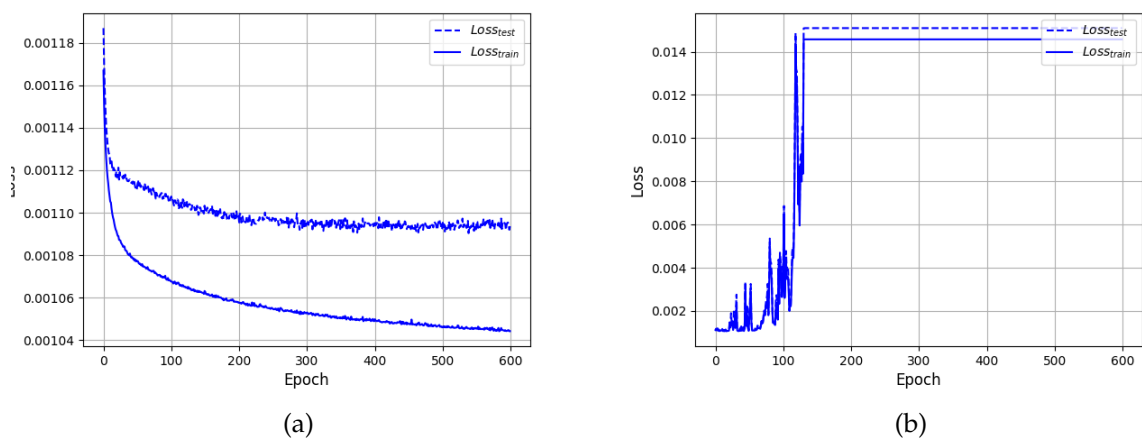


Figure 5.6: Loss curves for the chosen network architecture using SGD (a) and Adam (b).

of the network losses for a small learning rate of 0.001 in subfigure 5.7(a) and a relatively high learning rate of 0.2 in subfigure 5.7(b). For the small learning rate there are no signs of overtraining and no oscillations visible. However, the training converges slowly. For a high learning rate the  $Loss_{train}$  and  $Loss_{test}$  show larger divergence and each curve shows more oscillations individually. In the end, an intermediate learning rate of 0.06 is chosen. In this

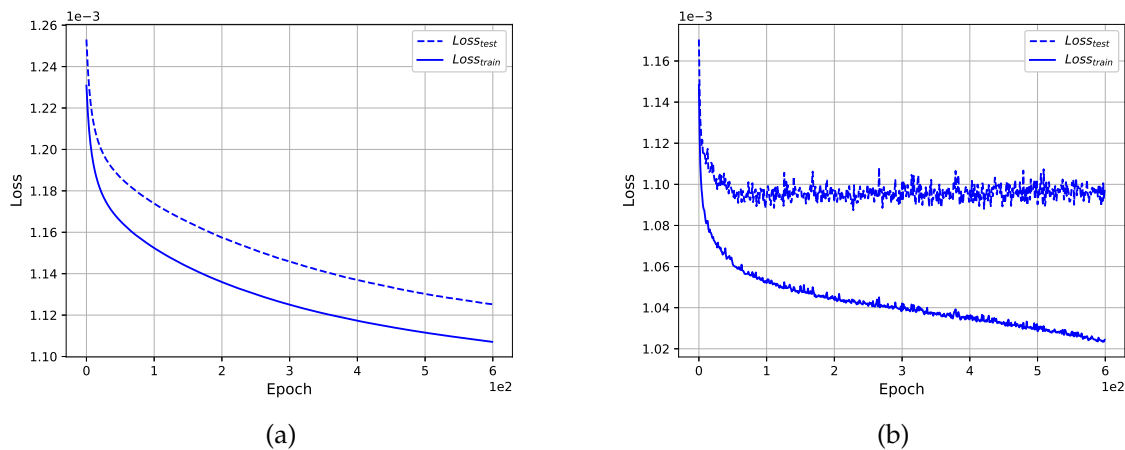


Figure 5.7: Loss behaviour for learning rates of different scales. Figure (a) shows losses for 0.001 and figure (b) for 0.2.

setup, dropout is used. That makes it a bit more difficult to see the clear signs of overtraining as it can be avoided for many setups. For a plot that shows this, see section 5.6.1.

Like the learning rate, the momentum was scanned over a wide range of values. Again, the risk of overtraining and the amount of oscillations was evaluated against the overall training efficiency. This resulted in a final value of 0.3. In addition, the test of using Nesterov momentum did not lead to any significant effects. It was therefore not utilized to avoid using unnecessary computational power.

The decay parameter was tested to see its effect on the loss behaviour. Figure 5.8 shows the standard loss behaviour compared to losses for a decay value of  $1 \times 10^{-6}$ . When decay is used, the losses in subfigure 5.8(b) become slightly more stable and plateau a bit earlier and at a greater value. Overall, no significant improvement is visible. For that reason, decay is not included in the final setup of the classifier. It was kept as an option for the later combination of the classifier and the adversary where it might help keeping the model close to the optimum while slowly updating to a model less sensitive to systematics.

## 5.6 Regularisation

The two regularisation tools tested were dropout-layers and batch-normalisation layers. The main purpose of dropout is regularising a network while batch-normalisation focuses on keeping all outputs on the same order which indirectly also helps the regularisation of the network. Dropout is tested for a range of values defining the percentage of nodes turned off per iteration step. Batch-normalisation is just tested in the default setup to investigate its effects.

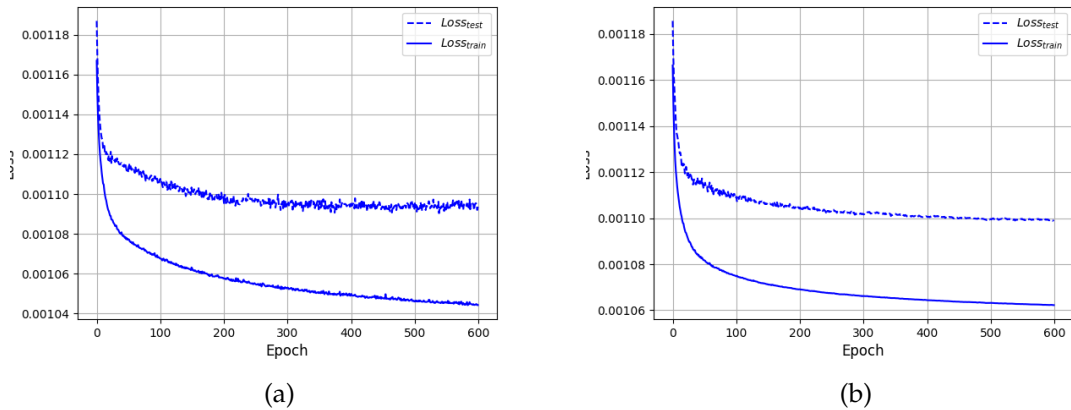


Figure 5.8: Comparison of usual loss behaviour and loss behaviour including a decay parameter. Figure (a) shows the losses without decay. Figure (b) shows the losses for a decay parameter of  $1 \times 10^{-6}$

### 5.6.1 Dropout

A dropout layer is added to each hidden layer of the network. Figure 5.9 shows the impact of dropout on the behaviour of the losses. For only 1 percent dropout, compared to the default 10 percent, overtraining occurs relatively early. For a very high dropout of 80 percent the losses behave strangely. The curve becomes unnaturally linear in some regions and converges less efficiently. In conclusion, dropout is a valuable addition to the network. It makes changing the architecture and the optimiser more forgiving and is a simple way to adjust a network without rearranging all hyper-parameters.

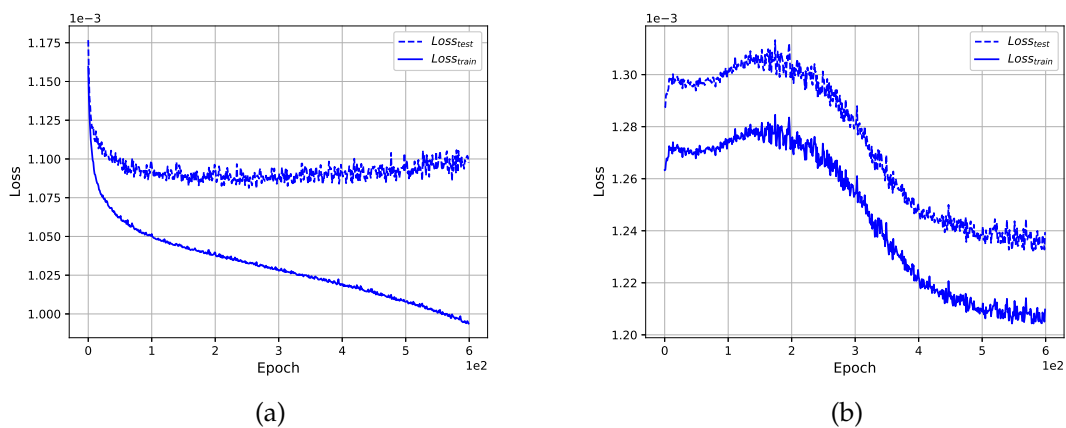


Figure 5.9: Comparison of usual loss behaviour and loss behaviour including high dropout. Figure (a) shows the losses for a dropout of 0.01. Figure (b) shows the losses for a dropout of 0.8



### 5.6.2 Batch normalisation

Batch normalisation was added to every deep layer and it had no significant effects on the output. Either the model does not develop any overly dominant features or dropout already does a sufficient job of normalisation. The extra layers were discarded for the classifying network to avoid an increase in computational time.



# 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 hyper-parameter investigations. In total three main approaches to the adversarial neural network were tested. The first one uses the understanding earned from training the classifier and the original approach introduced in the paper “Learning to pivot with Adversarial Networks” [4]. The second approach uses the last hidden layer of the classifier as input for the adversary instead of using the output of the classifier. Lastly, the information of the hidden layer is compressed to a smaller layer before being transferred to the adversary.

The first section of the chapter focuses on the initial run of the ANN using the base network presented in chapter 5. The results are investigated and the hyper-parameters are adapted using an initial setup for the second network. The problems with this setup are explained and possible reasons are listed. The second section describes the approach of using hidden layers as input. The results are compared to the initial approach and conclusions are drawn. The third approach deals with further adapting the information from the hidden layer and is described in the third section. Finally, the approaches are compared and possible further steps are listed.

## Prerequisites

The technical details for the setup are the same as for the classifying network as introduced in section 5.1. The adversarial setup uses the classifier’s general setup and architecture trained and tuned in chapter 5 as a basis. For the classical setup, this output is used as input for the adversarial network.

In general, both the classifier and the adversary are pre-trained. That means they are trained on their tasks without using a combined loss-function to generate better starting conditions for the adversarial training iterations. In addition, observing the losses of the networks during the pre-training indicates whether the networks influence each other.

## 6.1 Approach I: classical neural network

As already stated, the initial setup uses the classifier as trained in chapter 5 and an adversary inspired by this structure. Optimisation and activation remained the same while the architecture was simplified to factor the one-dimensional input into the system. Figure 6.1 shows the results. The network response 6.1(b) has almost no signs of a separation. Furthermore, the systematic distribution seems to be more pushed to the background distribution which is completely opposite to the original goal. It hints that modifications to the setup should be made. While the loss for both networks decreases, no minimum that hinders the adversary from learning is found. The expected increase in adversarial loss does not occur. The early spike in the losses comes from the pretraining losses.

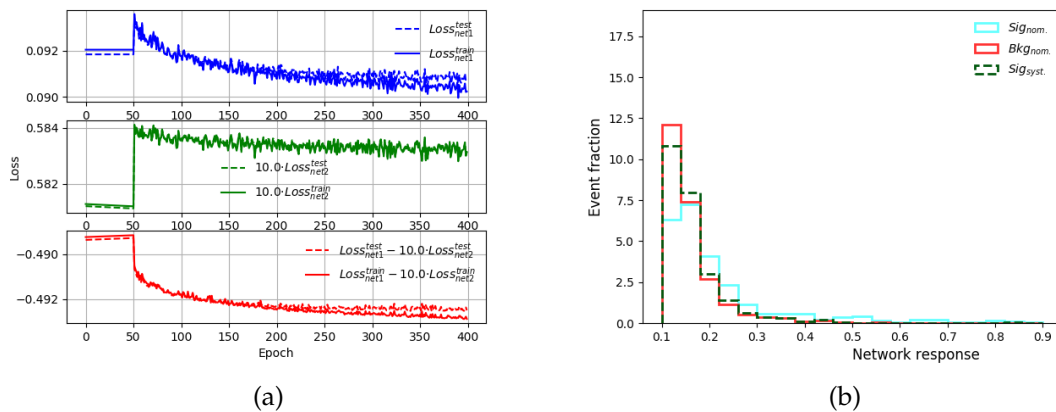


Figure 6.1: Results for an adversarial neural network using the classic approach and hyper-parameters adopted from the classifier. Figure (a) shows the losses; top to bottom: classifier, adversary, and combined loss. Figure (b) shows the separation for the nominal and the systematic sample; solid blue for nominal, dashed green for systematic, and red for the background.

It becomes clear that the optimised hyper-parameters for the classifier are not suitable for the task of training an adversarial neural network. This is an important insight because it means that the adversary is not something one can just add to a model to improve its sensitivity to a nuisance parameter. Instead, it is an integral aspect of a whole training procedure of its own. The next step taken during the analysis, presented here, is to try to investigate further what the performance of the combined network is strongly correlated with and how to understand what the network suffers from.

A hyper-parameter scan was performed in order to achieve a better performance for the adversarial neural network setup. The results for a very low learning rate and a non-existing momentum are shown in figure 6.2. With a slow optimisation the training becomes more stable and both networks can learn. This is assumed to be due that in an adversarial training, not just a minimum is sought. Instead, the classifier is first moved close to a minimum and then supposed to stay close to that model of decent classification, while also slowly adapting to features less sensitive to the systematic uncertainty. At some point, the adversarial loss stops decreasing. However, it does not increase again and in the response a difference between

nominal and systematic sample is visible

As even with these updated hyper-parameters no model is found that is sufficiently independent on systematics, another change to the network has been elaborated. On their own both networks improve their performance but in combination the classifier does not find a model that significantly downgrades the performance of the adversary. The worst case scenario is that this is due to no such model existing. It is very well possible that for this particular task there is no training that is independent of systematics. A simple change would be the set of input parameters. However, this would cause a whole new training problem and a subsequent hyper-parameter optimisation. For that reason, the input of the adversary was looked into. At this point, the adversary is just fed the sigmoid-output of the single, final node of the classifier. As mentioned earlier the amount of variables and the complexity of the architecture should be scaled with the task. Possibly the single output is sufficient to see a systematic dependency but not to improve the model using this insight. Furthermore, given the separation between  $tW$  and  $t\bar{t}$  is small and the systematic to be minimised is their interference term, the classifier network's output may not be enough for both networks to find an insensitive minimum. Approach II is based on this thought.

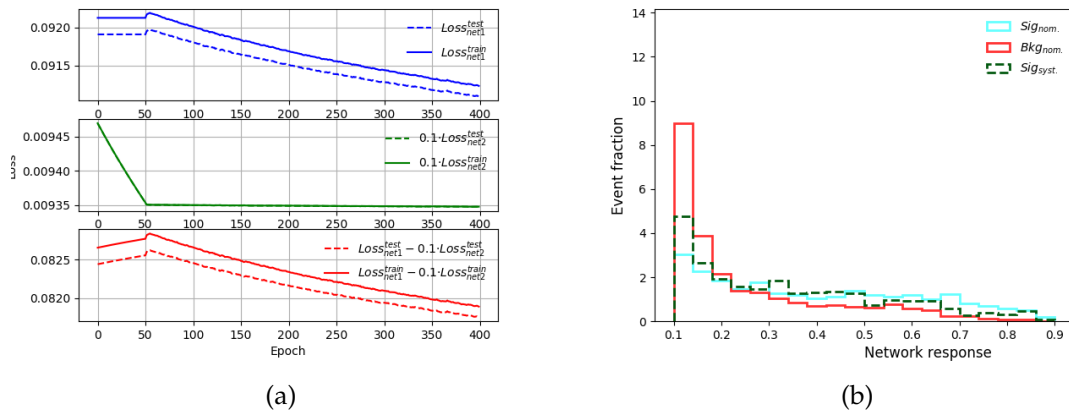


Figure 6.2: Results for an adversarial neural network using low learning rate and no momentum for the classifier. Figure (a) shows the losses; top to bottom: classifier, adversary, and combined loss. Figure (b) shows the separation for the nominal and the systematic sample; solid blue for nominal, dashed green for systematic, and red for the background.

## 6.2 Approach II: hidden layer input

The amount of information given to the adversary is the output of one single node. This does not justify the usage of complex architecture and it allows no insight into the deeper dependencies of the classification model. Alternatively one can use the last hidden layer of the classifier which can be described as the final model the sigmoid-decision is based on. This is not fully correct as all dependencies of the model created in the last step are excluded this way. Nevertheless, it is a viable approximation worth testing. This approach simply inputs the last hidden layer into the adversary without changing any other hyper-parameters. A

hyper-parameter scan is then performed to test the behaviour of the network. Using this setup

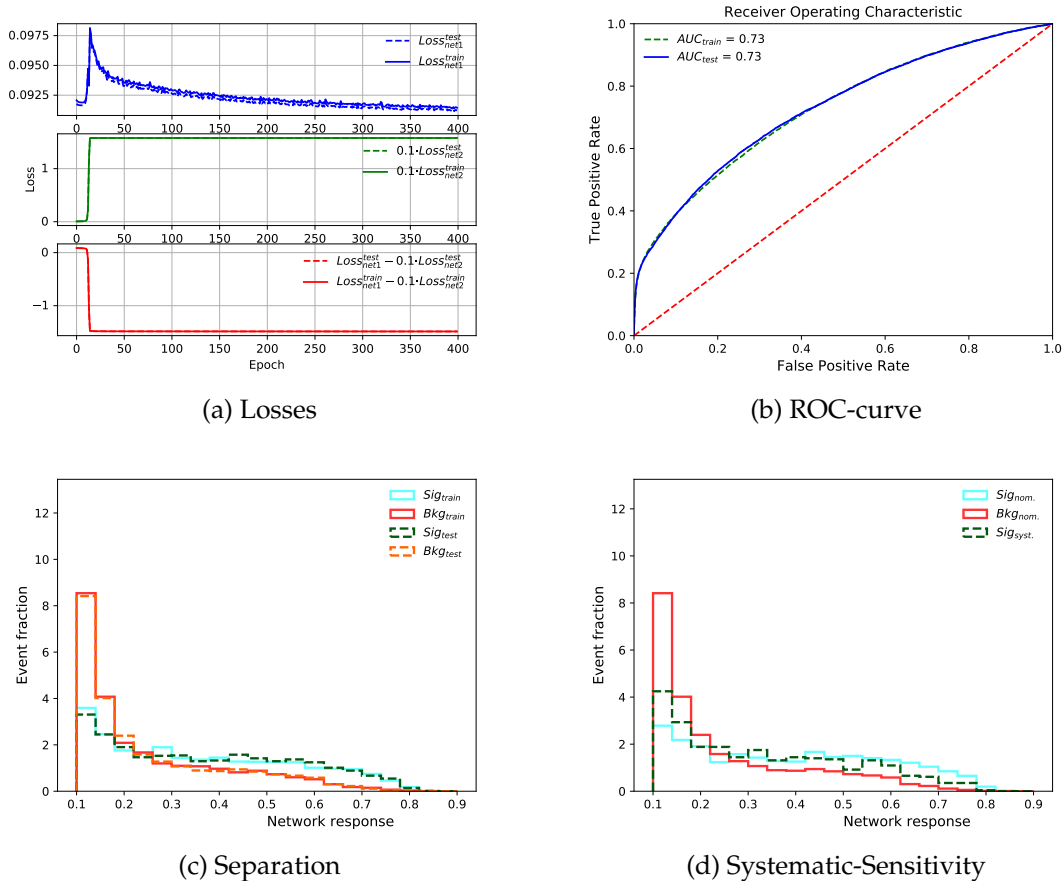


Figure 6.3: Performance plots for the second approach. Both a slightly better separation and a lower sensitivity are visible.

the losses behave as desired. Figure 6.3 summarises the results. At one point the adversary fails to gain any information while the classifier keeps on learning and the combined loss decreases. The rise of the adversary’s loss looks very steep but this is only a problem of the scale. In hindsight the model is very unstable and for many hyper-parameter setups it just stops working properly. Furthermore, the sensitivity to systematic samples does visibly improve. The improvements are small but the overall shape of the nominal and systematics distribution have become more alike and systematic distribution is less stretched to the background. A further interesting behaviour was achieved for a relatively high learning rate of 0.2. In this case the loss of the adversary increases later. The curve and separation is shown in figure 6.4. After the usual 400 training epochs the model seems to still be changing. There is a possibility that a better model for a higher epoch count could be achieved using this setup. It also shows that the optimisation of the hyperparameters has significantly different effects depending on which approach is chosen.

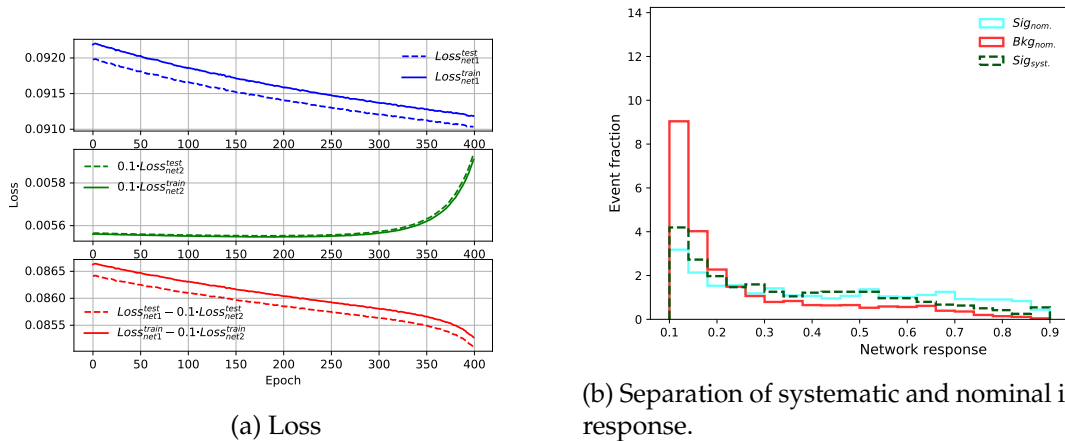


Figure 6.4: Behaviour of approach II for a learning rate of 0.2 showing a loss behaviour that could motivate a deeper analysis.

### 6.3 Approach III: compressed hidden layer input

After using only a single input node in the classical approach and 128 nodes in the second approach, an intermediate number of inputs is tested. For approach III, a second to last layer with only 16 nodes is added to the classifier and then fed into the adversary. This way the input dimension is closer to the dimension of the classifier. In addition, it promises to control the instability of the second approach.

Parallel to the other approaches, a hyper-parameter scan was performed. Figure 6.5 shows the result for an effective setup. The setup is about as unstable as that of approach II. Loss curves for different learning rates are shown in figure 6.6. The learning rate of 0.06 as originally used for the classifier performed badly for the classical, adversarial approach. For approach III it behaves different and in the end of the desired behaviour is visible. This is a feature that deserves more research. Furthermore figure 6.6(b) diagrams the odd behaviour for the usually very good learning rate of 0.001.

### 6.4 Summary

In conclusion, the approach for an adversarial neural network to reduce a model's sensitivity on systematic uncertainties, as suggested in the paper "Learning to pivot with Adversarial Networks" [4], has not been successful at achieving the promised results for a  $tW$ - $t\bar{t}$ -separation. It was, however, possible to show the desired behaviour of the two adversarial networks. It has been shown that the input to the adversary is essential and a part of the analysis that would deserve more research. Furthermore, the adversarial neural network has proven to be a structure of its own rather than just two arbitrary classifiers combined. They have to be built around an unhurried optimisation process allowing both networks to slowly learn from different areas of the system's topology.

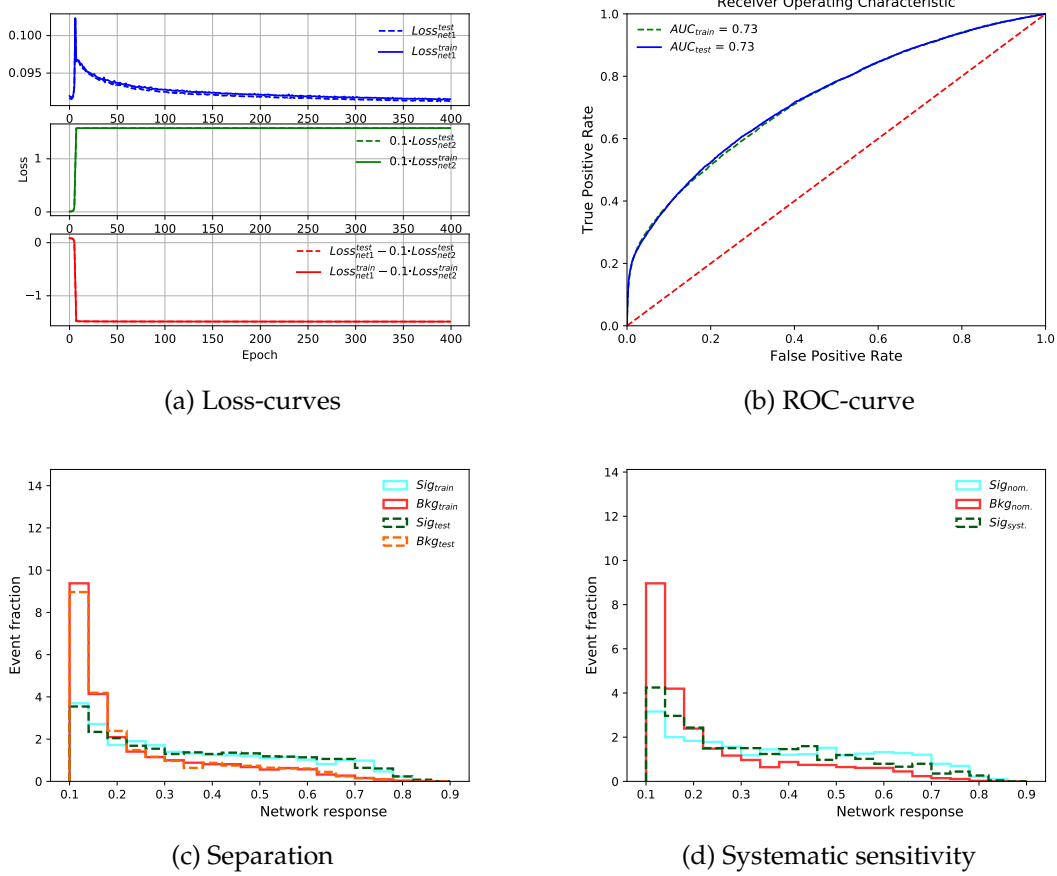


Figure 6.5: Performance of approach III. The plots show no significant improvements. The losses behave as expected.

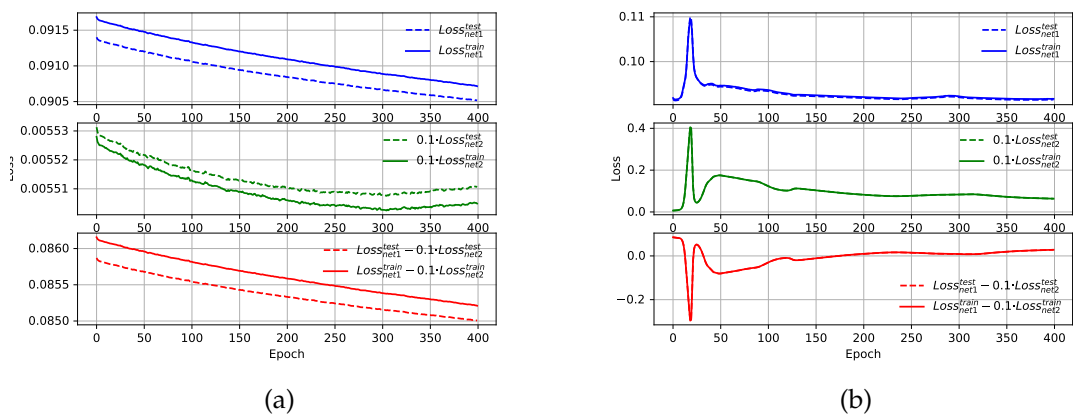


Figure 6.6: Miscellaneous loss plots for approach III. Figure 6.6(a) shows the losses for a learning rate of 0.06 and figure 6.6(b) for 0.001.



---

## Conclusions

---

The work presented in this thesis focussed on two issues of neural networks. The first one concerns the behaviour of neural networks and adversarial neural networks for classification tasks in particle physics. Here, the influence of the hyper-parameters on network performance for different setups was tested in order to understand how a classifier should be set up and how it can be upgraded to an adversarial neural network.

The second one addresses the possibilities offered by adversarial neural networks to limit the influence of systematic uncertainties on a classification model. In particular, the technique of an adversarial network was tested on the  $tW$ - $t\bar{t}$  separation and the systematic uncertainty arising from the different simulation approaches of DR and DS for the interference term at NLO.

A fundamental knowledge of neural networks and a set of hyper-parameters were discussed. It was shown that a good set of parameters is not only dependent on the topology of the problem but also massively depends on the setup of the network structure. A good classifier does not generally also form a good basis for an adversarial network structure. Instead, an adversarial network requires a low learning rate and above all a slow optimisation because more features of the problem need to be taken into account. This is due to the sensitivity to a certain systematic being added to the primary task of classification. The initial optimum, established for the classifier alone, that allowed for straightforward and fast training is not the desired model anymore.

Moreover, the adversarial neural network tested in this work is relatively unstable and the outcome changes significantly when hyper-parameters are slightly adjusted. This effect was not observed for the classifying network and resulted in a laborious optimisation process, which made the constraint of computational power even more significant. The most important feature of the adversarial network, investigated with the three approaches presented in chapter 6, is the dimension of input it is provided with. There is a strong argument that only a single node, as classically suggested, does not justify a complex adversarial network architecture. In addition to that, the general information lost when looking only at the final node should not be neglected during the setup. The performances achieved were strongly dependent on the shape of input.

Lastly, a significant improvement on the sensitivity of the classifier for the systematic sample could not be achieved. Although the classifier was able to generate a model that rendered its adversary unable to learn, no significant differences were visible in the distribution. The

shapes changed but, at the stage of this work, no sophisticated argument can be made for an improvement. In conclusion, switching from the single-node-input to a more complex input for the adversary seems to be the more promising approach. Combined with a different set of variables and a detailed hyper-parameter scan one might be able to achieve a reduction of sensitivity while keeping the quality of the overall classification constant. For similar problems it would be interesting to test out the consequences of only providing the data actually affected by the systematic uncertainties to the adversary. In this work the systematics only affected the  $tW$ -channel and the  $t\bar{t}$  was still considered nominal. This could have a strong bias effect on the result.

Neural networks in general are a promising technique for researchers in particle physics. The variety of highly different structures and hyper-parameters of the networks make them a potentially powerful tool for different scopes of research. An improvement in the sensitivity to systematics has been achieved for different researches using adversarial network approaches. Although a small effect on the sensitivity was achieved in this work, the sensitivity was not reduced as far as hoped for. Perhaps the topology of the problem and the shape of the systematics mean that the task has no model that is insensitive. However, the strategies presented in this work, pave the way for further research focusing on the approach and the appliance of machine learning techniques for data analysis in particle physics.

# Bibliography

---

- [1] *The Lego Group*, <https://www.lego.com/de-de>, Accessed: 2019-03-22 (cit. on p. 1).
- [2] Sascha Mehlhase, <https://build-your-own-particle-detector.org/models/atlas-lego-model-mini/?lang=de>, Accessed: 2019-03-23 (cit. on p. 1).
- [3] M. Thomson, *Modern Particle Physics*, 6th ed., Cambridge University Press, 2016, ISBN: 9781107034266 (cit. on pp. 1, 11).
- [4] G. Louppe, M. Kagan and K. Cranmer, *Learning to Pivot with Adversarial Networks*, (2016), arXiv: 1611.01046 [stat.ME] (cit. on pp. 1, 34–37, 51, 55).
- [5] 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](https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg) (cit. on p. 6).
- [6] 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> (cit. on p. 6).
- [7] D. Griffiths, *Introduction to Elementary Particle Physics*, 2nd ed., Wiley-VCH, 2008, ISBN: 9783527406012 (cit. on p. 11).
- [8] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003 (cit. on p. 13).
- [9] L. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008) S08001 (cit. on p. 13).
- [10] S. Chatrchyan et al., *The CMS Experiment at the CERN LHC*, JINST **3** (2008) S08004 (cit. on p. 13).
- [11] A. A. Alves Jr. et al., *The LHCb Detector at the LHC*, JINST **3** (2008) S08005 (cit. on p. 13).
- [12] K. Aamodt et al., *The ALICE experiment at the CERN LHC*, JINST **3** (2008) S08002 (cit. on p. 13).
- [13] E. Mobs, *The CERN accelerator complex - August 2018.*, (2018), General Photo, URL: <http://cds.cern.ch/record/2636343> (cit. on p. 14).
- [14] J.-L. Caron, “Overall view of LHC experiments.”, AC Collection. Legacy of AC. Pictures from 1992 to 2002., 1998, URL: <https://cds.cern.ch/record/841555> (cit. on p. 15).
- [15] N. Wermes and H. Kolanoski, *Teilchendetektoren Grundlagen und Anwendungen*, 1st ed., Springer Verlag, 2016, ISBN: 9783662453490 (cit. on pp. 15, 18).

- [16] J. Pequeno, "Computer generated image of the whole ATLAS detector", 2008, URL: <http://cds.cern.ch/record/1095924> (cit. on pp. 16, 21).
- [17] W. R. Leo, *Techniques for Nuclear and Particle Physics Experiments*, 2nd ed., Springer Verlag, 1994, ISBN: 3540572805 (cit. on p. 17).
- [18] ATLAS Collaboration, "The Pixel Detector of the ATLAS experiment for the Run2 at the Large Hadron Collider", *Proceedings, 10th International Conference on Position Sensitive Detectors (PSD10): Surrey, UK, September 7-12, 2014*, vol. 10, 02, 2015 C02001, arXiv: 1411.5338 [physics.ins-det] (cit. on p. 17).
- [19] ATLAS Collaboration, *Electron efficiency measurements with the ATLAS detector using the 2015 LHC proton-proton collision data*, ATLAS-CONF-2016-024, CERN, 2016, URL: <https://cds.cern.ch/record/2157687> (cit. on p. 19).
- [20] ATLAS Collaboration, *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] (cit. on p. 19).
- [21] M. Cacciari, G. P. Salam and G. Soyez, *The anti- $k_t$  jet clustering algorithm*, JHEP **04** (2008) 063, arXiv: 0802.1189 [hep-ph] (cit. on p. 19).
- [22] ATLAS Collaboration, *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] (cit. on p. 19).
- [23] 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, URL: <http://cds.cern.ch/record/2110203> (cit. on p. 19).
- [24] ATLAS Collaboration, *Optimisation of the ATLAS b-tagging performance for the 2016 LHC Run*, ATL-PHYS-PUB-2016-012, CERN, 2016, URL: <https://cds.cern.ch/record/2160731> (cit. on p. 19).
- [25] ATLAS Collaboration, *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] (cit. on p. 20).
- [26] ATLAS Collaboration, *The ATLAS Simulation Infrastructure*, Eur. Phys. J. **C70** (2010) 823, arXiv: 1005.4568 [physics.ins-det] (cit. on p. 22).
- [27] S. Agostinelli et al., *GEANT4: A Simulation toolkit*, Nucl. Instrum. Meth. **A506** (2003) 250 (cit. on p. 23).
- [28] ATLAS Collaboration, *The ATLAS calorimeter simulation FastCaloSim*, Journal of Physics: Conference Series **331** (2011) 032053, URL: <https://doi.org/10.1088%2F1742-6596%2F331%2F3%2F032053> (cit. on p. 23).
- [29] ATLAS Collaboration, *Summary of ATLAS Pythia 8 tunes*, ATL-PHYS-PUB-2012-003, CERN, 2012, URL: <https://cds.cern.ch/record/1474107> (cit. on p. 23).

- 
- [30] Kjell Magne Fauske,  
URL: <http://www.texample.net/tikz/examples/neural-network/>  
(cit. on p. 26).
- [31] F. Chollet et al., *Keras*, <https://keras.io>, 2015 (cit. on pp. 28–31, 40).
- [32] 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. 31).
- [33] 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. 34).
- [34] G. B. Team, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*,  
Software available from [tensorflow.org](http://tensorflow.org), 2015, URL: <http://tensorflow.org/>  
(cit. on p. 40).
- [35] F. Seide and A. Agarwal, “CNTK: Microsoft’s Open-Source Deep-Learning Toolkit”,  
*Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and  
Data Mining*, KDD ’16, ACM, 2016 2135, ISBN: 978-1-4503-4232-2,  
URL: <http://doi.acm.org/10.1145/2939672.2945397> (cit. on p. 40).
- [36] Theano Development Team,  
*Theano: A Python framework for fast computation of mathematical expressions*,  
arXiv e-prints **abs/1605.02688** (2016), URL: <http://arxiv.org/abs/1605.02688>  
(cit. on p. 40).
- [37] ATLAS Collaboration, “Measurement of the cross-section for the production of a  
W-boson in association with a top quark at 13 TeV”, ATL-COM-PHYS-2019-222, 2019,  
URL: <https://cds.cern.ch/record/2667560> (cit. on pp. 40, 42).



---

## Monte Carlo simulations

---

The following section provides the technical details for the Monte Carlo simulations used in this work. All of the sets are distributed by the SingleTop group at ATLAS. The sets used are v23 sets from Release 21. They were filtered to only the 2 j2b region.

DSID	$\sigma[pb]$	$k$ -factor	Tags
Single top-quark, $tW$ -channel, DR			
410648	4.0	0.94	e6615_e5984_s3126_r9364_r9315_p3409
410649	3.99	0.95	e6615_e5984_a875_r9364_r9315_p3526
Single top-quark, $tW$ -channel, DS			
410656	3.89	0.97	e6615_e5984_s3126_r10201_r10210_p3409
410657	3.97	0.95	e6615_e5984_s3126_r10201_r10210_p3409
Top-quark pair			
410657	76.95	1.14	e6348_e5984_a875_r10201_r10210_p3554

Table A.1: MC samples used in this thesis.





## Pre-training losses

This chapter shows two exemplary loss curves during the pre-training. They have not been shown in the main part. However, one can observe the dependency of the adversary on the classifier. Secondly it can be seen that the adversary does not affect the classifier.

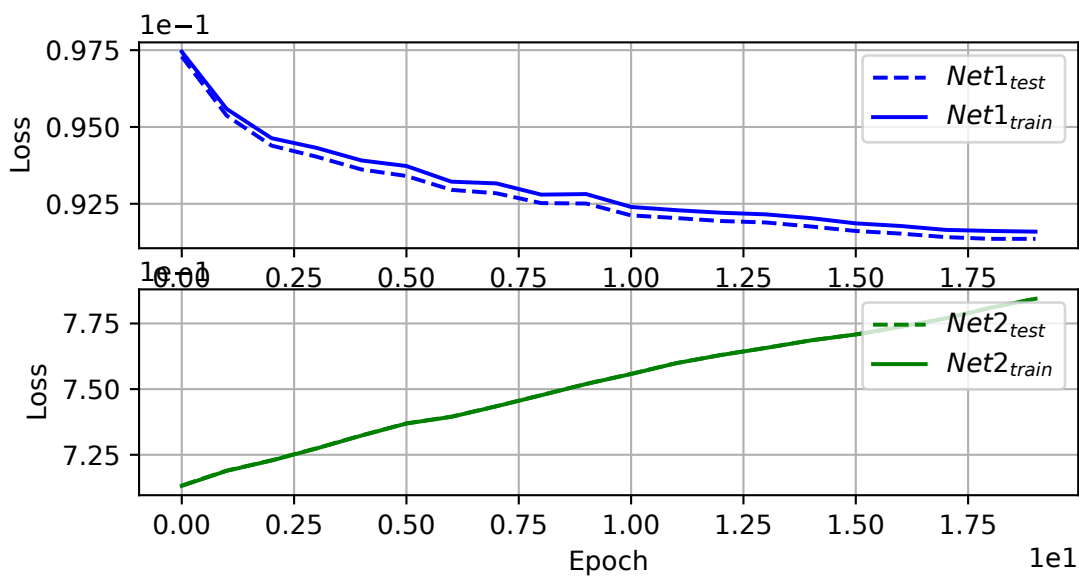


Figure B.1: Pre-training of the classifier. It is visible that even a normal training worsens the adversary's performance. It also indicates that the adversary really depends on the classifier.

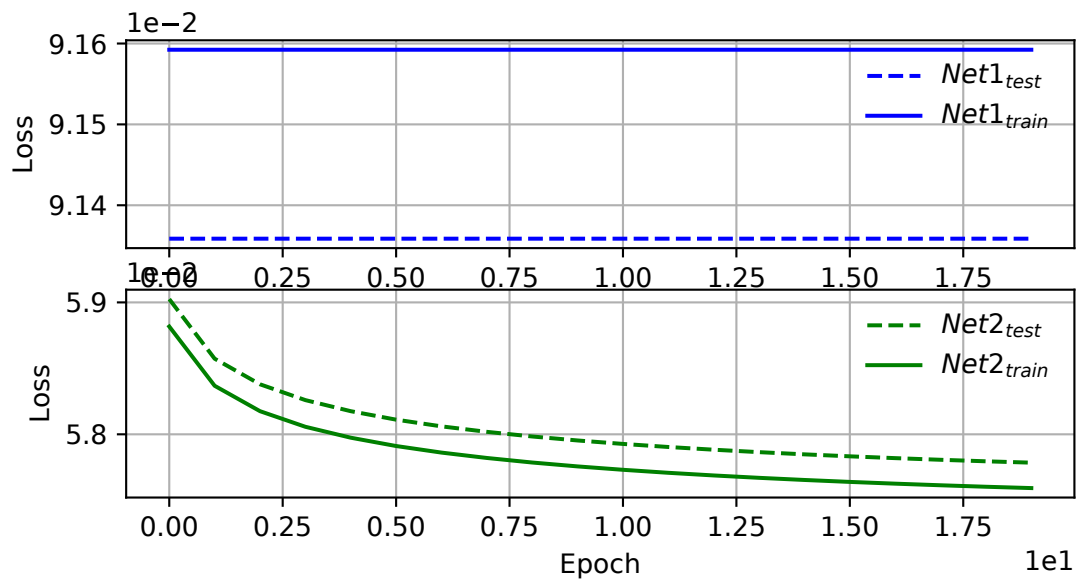


Figure B.2: Pre-training of the adversary. It is visible, that the adversary does not affect the classifier directly.

# List of Figures

---

2.1	Standard Model of Particle Physics . . . . .	6
2.2	$t\bar{t}$ pair production feynman diagrams at LO . . . . .	8
2.3	Single-top-production diagrams . . . . .	9
2.4	Final state of a $tW$ decay . . . . .	10
2.5	Comparison of the final state of a $t\bar{t}$ and $tW$ event . . . . .	10
3.1	Sketch of the LHC accelerator complex . . . . .	14
3.2	Sketch of the LHC ring. . . . .	15
3.3	Sketch of the ATLAS detector . . . . .	16
3.4	Scheme of the ATLAS detector's detection procedure . . . . .	21
4.1	Sketch of a typical neural network structure . . . . .	26
4.2	Network parameter nomenclature . . . . .	27
4.3	Dropout Sketch . . . . .	33
4.4	Adversarial setup sketched . . . . .	36
4.5	Exemplary loss of an adversarial network structure . . . . .	37
5.1	Network performance of the classifier . . . . .	41
5.2	Performance of the classifier using different variables . . . . .	43
5.3	Network performance's dependency on the architecture . . . . .	44
5.4	Classifier losses for a complex architecture . . . . .	45
5.5	Classifier performance for different activation functions . . . . .	46
5.6	Classifier losses for different optimisers . . . . .	46
5.7	Classifier loss for different learning rates . . . . .	47
5.8	Performance of the classifier with decay . . . . .	48
5.9	Performance of the classifier for different dropout percentages . . . . .	48
6.1	ANN results for a standard setup . . . . .	52
6.2	ANN results for optimised parameters . . . . .	53
6.3	ANN results approach II . . . . .	54
6.4	ANN approach II, high learning rate . . . . .	55
6.5	ANN results approach III . . . . .	56
6.6	Misc plots approach III . . . . .	56
B.1	Classifier pretraining . . . . .	65
B.2	Adversary pretraining . . . . .	66



# List of Tables

---

4.1	Selection of activation function . . . . .	29
5.1	Simple kinematic variables . . . . .	42
5.2	Complex variables . . . . .	42
A.1	DSID list . . . . .	63