ETP-KA/2018-20

Domain Adaptation Studies in Deep Neural Networks for Heavy-Flavor Jet Identification Algorithms with the CMS Experiment

David Walter

Department of Physics
Karlsruhe Institute of Technology (KIT)

MASTER THESIS

Referee: Dr. Matthias Ulrich Mozer
Co-Referee: Prof. Dr. Günter Quast
Institut für Experimentelle Teilchenphysik

September 03, 2018

Akzeptiert vom ersten Referenten der Masterarbeit.
Karlsruhe, den 03. September 2018

. . . . . .
(**Dr. Matthias Ulrich Mozer**)

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, den 03. September 2018

. . . . . .
(**David Walter**)

Introduction

In the last decades, the existence of several elementary particles was proven at particle accelerator experiments. In many analyses, their couplings and interactions were determined and particle properties like their masses were measured. These measurements were made possible by efficient algorithms for the reconstruction and identification of the particles created in particle collisions. Since the discovery of the Higgs boson [1, 2], the standard model (SM) forms a self-consistent theory, describing all known elementary particles. In the search for new physics, measured data is accurately analyzed and compared with SM predictions. An improvement of the reconstruction and identification algorithms can lead to more precise results that can reveal small deviations from the SM.

Quarks and gluons are elementary particles, which interact via the strong interaction and occur in most particle collisions. The identification of heavy-flavor quarks, namely the charm, bottom and top quarks, is important as they are produced in many rare processes. The top quark is the heaviest known elementary particle and therefore of particular importance. As it decays in almost every case into a bottom quark and a W boson, the identification of the bottom quarks is crucial to reconstruct top quarks. Bottom quark-antiquark pairs are furthermore one of the most frequent decay products of the Higgs boson, which is the only scalar particle in the SM and under detailed investigation [3, 4]. The identification of charm quarks is for example needed in the search for new supersymmetric particles [5]. Apart from this, the decay of the Higgs boson into a charm quark-antiquark pair can be used to study the coupling of the Higgs boson to fermions of the second generation [6]. Furthermore, the bottom quark identification performance benefits from a discrimination between bottom and charm quarks.

Gluons and all quarks, except for the top quark, form particles bound by the strong force before they decay. They cannot be detected directly, but create a particle shower, which makes up a so-called jet. Particles with bottom, charm and light-flavor (up, down and strange) quarks have different lifetimes. Bottom quarks have a much higher mass than charm, light-flavor quarks or gluons and travel only short distances of some millimeters within modern particle accelerator experiments. The decay of particles with bottom quarks into particles without bottom quarks releases on average more energy, distributed to a higher number of daughter particles compared to decays of particles with light-flavor quarks and gluons only. Furthermore, with an exact reconstruction

of the particle tracks, the resulting different decay lengths can be exploited to identify the particle that caused the jet. Particles with charm quarks have a decay length that is shorter than these from bottom quarks and their mass is in between of these from particles with bottom or light flavor quarks. Therefore a separation of jets originating from charm quarks from these with bottom quarks or light-flavor quarks and gluons is especially challenging.

Particle physics benefits from the rapid improvements in the field of machine learning in the present years. Powerful algorithms like deep neural networks (DNNs) are well suited for analyses with millions of events. The tools are especially helpful in the jet identification by performing a multivariate classification. Based on the differences in the jet properties, DNNs can learn to distinguish jets by their initial quark flavor in a process called training. The training is performed on jets obtained from simulation. However, in the comparison between simulated and measured jets, differences occur which lead to increased uncertainties in the jet recognition for jets from measured data. The concept of domain adaptation from the field of transfer learning provides promising methods to mitigate this problem.

This thesis explores the application of domain adaptation methods in an empirical study of jet identification algorithms that are based on DNNs. Two different domain adaptation methods are applied for events from the CMS experiment, simulated or recorded in the data taking period of 2016. The domain adaptation is performed on events, selected similar to recently popular analyses: top quark-antiquark pair processes with an electron-muon pair or a muon and four jets in the final state. It is shown that the domain adaptation acts as a regularizer between jets from recorded data and simulation and leads to an improvement in the data-to-simulation agreement of the jets. Furthermore, the performance of the jet identification algorithms with the use of domain adaptation is calculated for events from recorded data and simulation and compared to the performance of the standard algorithms.

Contents

1 The Standard Model

The standard model of particle physics (SM) describes all known elementary particles and three of the four known fundamental forces. These forces are the electromagnetic interaction, the weak interaction and the strong interaction. Gravity is not included yet, but is negligible on a microscopic scale. The SM predicted many particles that were found later in experiments, for example the top quark in 1995 [7], the tau neutrino in 2000 [8] or the Higgs boson in 2012 [1, 2]. It also provides precise predictions about the properties of elementary particles like the Landé g-factor [9]. Despite the great success of the SM, hints exist that the particles and their interactions are described by a more fundamental mechanism. For example, many free parameters, like the masses of particles, cannot be predicted by the SM and have to be determined by measurements. Furthermore, dark matter and dark energy are not described by the SM. Therefore, to find evidences for new physics, it is important to measure the particle properties predicted by the SM with best possible precision.

This chapter gives a brief summary of the SM with focus on the electroweak interaction and quantum chromodynamics. A more detailed description of the SM can be found in references [10–13]. Natural units with $\hbar = c = 1$ are used to simplify the formulas quoted in this chapter.

1.1 Overview

The SM is a quantum field theory that describes the particles as fields and that includes quantum mechanics as well as special relativity. It is a gauge theory in which underlying interactions are described by the $\mathrm{U}(1)_\mathrm{Y} \otimes \mathrm{SU}(2)_\mathrm{L} \otimes \mathrm{SU}(3)_\mathrm{C}$ symmetry group.

Special relativity requires a Lorentz invariant description of the SM. The spin of a particle defines its representation of the Lorentz group and has a direct influence on its behavior under Lorentz transformation. With respect to the spin quantum number, the particles in the SM can be divided into two groups, fermions with half integer spin and bosons with integer spin. Fermions are described as spinor fields and are the constituents of all matter, for example atoms. They exist in three generations, where only the first generation of particles is stable and does not decay over time. Fermions can be further distinguished into quarks and leptons. Quarks are the constituents of protons and neutrons, they interact through all forces and can be distinguished by their elec-

Table 1.1: The fermions of the standard model. All quarks and leptons of the SM and their electroweak charges are listed. The fermions occur with negative chirality (left handed) in weak isospin doublets or with positive chirality (right handed) in weak isospin singlets. Under the assumtion that the neutrino mass is zero, there is no right handed neutrino in the SM. For each fermion, an antifermion exists with opposite charge and chirality.

Fermions	Generation 1	Generation 2	Generation 3	Weak hypercharge Y_W	3rd comp. of isospin I_3	Electric charge Q
Quarks	$\begin{pmatrix} u \\ d \end{pmatrix}_L$	$\begin{pmatrix} c \\ s \end{pmatrix}_L$	$\begin{pmatrix} t \\ b \end{pmatrix}_L$	$+1/3$	$+1/2$	$+2/3$
				$+1/3$	$-1/2$	$-1/3$
	u_R	c_R	t_R	$+4/3$	0	$+2/3$
	d_R	s_R	b_R	$-2/3$	0	$-1/3$
Leptons	$\begin{pmatrix} \nu_e \\ e \end{pmatrix}_L$	$\begin{pmatrix} \nu_\mu \\ \mu \end{pmatrix}_L$	$\begin{pmatrix} \nu_\tau \\ \tau \end{pmatrix}_L$	-1	$+1/2$	0
				-1	$-1/2$	-1
	e_R	μ_R	τ_R	-2	0	-1

tric charge. Quarks with an electric charge of $+2/3\,e$ are called up-type quarks, while down-type quarks carry an electric charge of $-1/3\,e$. The charged leptons, the electron, the muon and the tau lepton can interact via the electromagnetic and weak force. The neutral leptons, the neutrinos, can only interact via the weak force. A summary of the fermions is given in table 1.1. The second group of the SM consists of bosons. Bosons with spin 1, described by vector fields, are the mediator particles for the three fundamental forces of the SM: the photon (γ) for the electromagnetic interaction, the $W^\pm$ and Z bosons for the weak interaction and the gluons (g) for the strong interaction. The Higgs boson is the only particle of the SM with spin 0 and is described by a scalar field. It interacts with all massive particles. A summary of the bosons is shown in table 1.2.

1.2 Electroweak Interaction

The electromagnetic interaction and the weak interaction are described in a unified electroweak theory by the $U(1)_Y \otimes SU(2)_L$ group. By demanding the theory to be locally invariant under $U(1)_Y$ transformations, there must exist a massless vector boson (B), and a conserved quantity, the weak hypercharge Y_W. To obtain a local $SU(2)_L$ symmetry, three additional massless vector bosons (W_0, W_1 and W_2) are needed; the conserved quantity of this symmetry is the third component of the weak isospin I_3. The electric charge

$$Q = \frac{Y_W}{2} + I_3 \quad , \tag{1.1}$$

Table 1.2: Summary of the bosons of the SM. The vector bosons with spin 1, the photon, the $W^{\pm}$ and Z bosons and the gluons are the mediator particles of the three fundamental forces that are described by the SM. The Higgs boson with spin 0 is not mediating a force, but it couples to the mass of the particles, masses are taken from [14]

Boson	Force	Couples to	Mass (GeV)
photon (γ)	electromagnetic	electric charge	0
$W^{\pm}$	weak	weak charge	80.379
Z			91.188
8 gluons (g)	strong	color charge	0
Higgs (H)	-	mass	125.18

is related with these conserved quantities and is hence conserved. The Higgs boson originates from a spontaneous symmetry breaking of the local $U(1)_Y \otimes SU(2)_L$ symmetry and leads to a non-diagonal mass matrix for the four vector bosons. This mass matrix can be diagonalized to find the mass eigenstates

$$\begin{aligned} \gamma &= \cos(\theta_W)B + \sin(\theta_W)W_0 \quad , \\ Z &= -\sin(\theta_W)B + \cos(\theta_W)W_0 \quad , \\ W^{\pm} &= \frac{1}{\sqrt{2}}(W_1 \mp iW_2) \quad , \end{aligned} \tag{1.2}$$

with the Weinberg angle θ_W. In these mass eigenstates, the photon remains massless while the Z, W^+ and W^- bosons receive a mass. The Z boson and the photon carry no electric charge, while the $W^{\pm}$ bosons carry an electric charge of $\pm 1\,e$. The large mass of the Z and $W^{\pm}$ bosons is responsible for the short range of the weak interaction, as they have a lifetime of less than 10^{-24} s [14]. The effective strength can be estimated with the Fermi coupling constant of $G_F \approx 10^{-5}\,\mathrm{GeV}^{-2}$ of the weak interaction. The electromagnetic coupling constant has a value in the order of 10^{-2}.

With respect to $SU(2)_L$ transformations, the quarks and leptons of one generation are grouped into weak isospin doublets, consisting of two left-handed spinor fields with $I_3 = \pm 1/2$. Weak isospin singlets with $I_3 = 0$ are represented by right-handed spinor fields, which cannot couple to charged currents. By interacting through $W^{\pm}$ bosons, flavor[1] changing charged currents are possible. They describe transitions of fermions within a weak isospin doublet, for example $u \to d + W^+$. For quarks, this current is

[1] The flavor indicates the type of a particle. For example, there are six quark flavors (u, d, s, c, b, t) and six lepton flavors (e, μ, τ , ν_e, ν_μ, ν_τ).

given by

$$\begin{aligned} J^{\mu}_{\mathrm{CC}} &= \begin{pmatrix} \bar{\mathrm{u}} & \bar{\mathrm{c}} & \bar{\mathrm{t}} \end{pmatrix} \gamma^{\mu} \frac{1-\gamma^5}{2} \begin{pmatrix} \mathrm{d}' \\ \mathrm{s}' \\ \mathrm{b}' \end{pmatrix} \\ &= \bar{\mathrm{u}}\, \gamma^{\mu} \frac{1-\gamma^5}{2}\, \mathrm{d}' + \bar{\mathrm{c}}\, \gamma^{\mu} \frac{1-\gamma^5}{2}\, \mathrm{s}' + \bar{\mathrm{t}}\, \gamma^{\mu} \frac{1-\gamma^5}{2}\, \mathrm{b}' \quad , \end{aligned} \tag{1.3}$$

with the Dirac matrices γ^{μ} and γ^5 and $\bar{q} = q^{\dagger}\gamma_0$ denoting the Dirac adjunct spinor of q. The quarks described in table 1.1, are in their electroweak eigenstate and are distinguished from the observable mass eigenstates. Here, q' denotes a quark in its electroweak eigenstate and q in its mass eigenstate. The mass eigenstates of the quarks are therefore a mixture of their electroweak eigenstates and vice versa:

$$\begin{pmatrix} \mathrm{d}' \\ \mathrm{s}' \\ \mathrm{b}' \end{pmatrix} = V \begin{pmatrix} \mathrm{d} \\ \mathrm{s} \\ \mathrm{b} \end{pmatrix} = \begin{pmatrix} V_{\mathrm{ud}} & V_{\mathrm{us}} & V_{\mathrm{ub}} \\ V_{\mathrm{cd}} & V_{\mathrm{cs}} & V_{\mathrm{cb}} \\ V_{\mathrm{td}} & V_{\mathrm{ts}} & V_{\mathrm{tb}} \end{pmatrix} \begin{pmatrix} \mathrm{d} \\ \mathrm{s} \\ \mathrm{b} \end{pmatrix} \quad . \tag{1.4}$$

The 3×3 unitary matrix V is the Cabibbo-Kobayashi-Maskawa matrix (CKM-matrix) that allows transitions of quarks between two generations. The probability of these transitions is low due to small off-diagonal values of V [14]:

$$V = \begin{pmatrix} 0.97420 \pm 0.00021 & 0.2243 \pm 0.0005 & 0.00394 \pm 0.00036 \\ 0.218 \pm 0.004 & 0.997 \pm 0.017 & 0.0422 \pm 0.0008 \\ 0.0081 \pm 0.0005 & 0.0394 \pm 0.0023 & 1.019 \pm 0.025 \end{pmatrix} \quad . \tag{1.5}$$

The matrix V has four free parameters that cannot be predicted by the SM and have to be determined by measurements. By inserting equation 1.4 into equation 1.3, one can, for instance, obtain the current for a transition of a b quark to a c quark . The relevant part is given by

$$J^{\mu}_{\mathrm{CC,cb}} = \bar{\mathrm{c}}\, \gamma^{\mu} \frac{1-\gamma^5}{2}\, V_{\mathrm{cb}}\, \mathrm{b} \quad , \tag{1.6}$$

which is suppressed by the factor of $V_{\mathrm{cb}} = 0.0405$.

1.3 Quantum Chromodynamics

The strong interaction is responsible for the interaction between quarks and gluons and is described by quantum chromodynamics (QCD) with the underlying summetry of the $\mathrm{SU}(3)_{\mathrm{C}}$ group. Eight gluons are needed to obtain a $\mathrm{SU}(3)_{\mathrm{C}}$ invariant description. The charge of this symmetry is called color, of which three different types exist: red,

green and blue. For each color, there is also a corresponding anticolor. As this symmetry is not broken, the gluons remain massless and indistinguishable. They carry one color and one anticolor charge, this allows interactions of gluons among themselves. Quarks on the other hand carry one color charge while antiquarks are charged with an anticolor.

The nature of the strong force leads to a potential for a quark-antiquark pair of

$$V(r) = -a \cdot \frac{\alpha_s(r)}{r} + b \cdot r \quad , \tag{1.7}$$

with the positive constants a and b, the radius between the two particles r and the strong couplings constant $\alpha_s(r)$, which is dependent on the radius and is in the order of unity[2]. For small radii, the first term is dominant and allows bound states. The lower the distance between two particles, the lower is the strong interaction between them, this effect is called asymptotic freedom. On the other hand, the potential increases linearly with the distance between the particles. If a quark in a bound state is pulled apart from the other quark(s) in this bound state, the energy of the system is steadily increasing. When the energy is high enough, a quark-antiquark pair is produced in order to form two color neutral objects with the separated particles. This effect is called confinement and is the reason that no color-charged isolated object exists. A color-neutral object that consists of quarks is called hadron.

Hadrons can be grouped into mesons and baryons. Mesons are quark-antiquark pairs, whereas baryons consist of three quarks. Besides the top quark, all quarks can be bound in hadrons, therefore a huge number of different hadrons exists. Hadrons containing b quark(s) are denoted as b hadrons. Hadrons without b quark(s) but with at least one c quark are called c hadrons. Hadrons are bound by the strong force and decay through the strong, weak or electromagnetic interaction. Therefore, the lifetime of hadrons [14] varies from 10^{-8} s for charged pions ($u\bar{d}$, $\bar{u}d$), 10^{-12} s for B mesons ($u\bar{b}$, $\bar{u}b$, $\bar{b}d$, $b\bar{d}$) to values up to 10^{-24} s for hadrons in excited states like the ρ meson ($u\bar{d}$, $\bar{u}d$). The only stable hadron is the proton (uud), as the neutron (udd) is only stable in atomic nuclei. The top quark is the only quark that cannot occur in a bound state because its lifetime of about $5 \cdot 10^{-25}$ s [15] is roughly 20 times smaller than the hadronization time.

[2]For instance, at the energy of the mass of the Z boson, $M_Z = 91.188\,\text{GeV}$, $\alpha_s(M_Z^2) = 0.1181 \pm 0.0011$ [14]

2 The Compact Muon Solenoid Experiment at the Large Hadron Collider

Particle accelerator experiments have a history of great success in the investigation of physical laws. With these machines, it was possible to prove the existence of different elementary particles of the standard model, for instance, the top quark [7]. Particle accelerators make use of the electric charge of particles to accelerate them to velocities close to the speed of light. In a circular collider, particles are moving in circles in opposite directions and are forced to head-on collisions. These collisions release a high amount of energy and heavy particles are produced that are highly unstable and decay after a short time. Detectors are used to measure the properties of these particles directly or indirectly.

This chapter gives a brief overview of the collider and detector system that has produced and measured the events used in this work.

2.1 The Large Hadron Collider

The Large Hadron Collider (LHC) [16, 17] is a circular collider located at the European Organization for Nuclear Research (CERN) in the region of Geneva. With a circumference of 27 km, it is the world's largest and most powerful particle accelerator and lies in a tunnel about 100 m beneath the earth. The LHC is used to accelerate protons, xenon ions and lead ions. These particles undergo several steps of production and pre-accelecation before they are injected into the LHC. An illustration of the system is shown in figure 2.1.

The LHC consists of two beam pipes, allowing the acceleration of protons in opposite directions. In this setup, for head-on collisions with equal beam energies, the center-of-mass energy $\sqrt{s}$ is twice the beam energy E:

$$\sqrt{s} = 2 \cdot E \quad . \tag{2.1}$$

In 2010, the first collisions with a beam energy of 3.5 TeV started. In 2012, the center-of-mass energy was increased to $\sqrt{s} = 8$ TeV. From 2013 to 2015, the LHC was shut down for maintenance. After this, Run 2 started in 2015 with $\sqrt{s} = 13$ TeV. After a shutdown

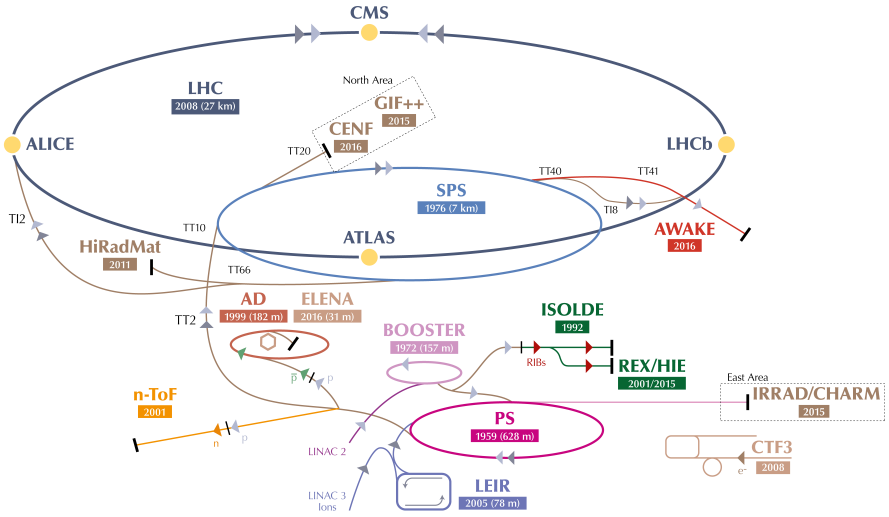


Figure 2.1: The CERN accelerator complex. The figure shows several accelerator facilities at CERN. The protons that end up in the LHC are obtained from hydrogen atoms and are first accelerated with the LINAC2 accelerator to 50 MeV. Next, they are injected into the Booster accelerator and are accelerated to 1.4 GeV. It follows the acceleration in the Proton Synchrotron (PS) to 25 GeV and in the Super Proton Synchrotron (SPS) to 450 GeV. Finally, the protons are injected into the LHC in both directions. From there they reach a final energy of 6.5 TeV within 20 minutes. In addition, the four main detectors at the LHC are shown [18].

in 2018, the LHC is planned to run at $\sqrt{s} = 14\,\text{TeV}$ in 2021 for Run 3.

The protons in the beam are packed in up to 2808 bunches, each bunch consists of about $1.2 \cdot 10^{11}$ protons. For each beam, eight superconducting cavities, cooled down to 4.5 K, generate an electromagnetic potential with a frequency of 400 MHz to accelerate and tighten the bunches. Each of the cavities is delivering an accelerating field of 5 MV/m. To avoid collisions with gas molecules during operation, the beam pipes have a vacuum pressure of 10^{-13} atm. Dipole magnets with a magnetic field strength of up to 7.74 T each, force the particle beam to stay on the circular path. Quadrupole magnets and magnets of higher order are used to focus the beam. To keep the magnets in superconducting state, superfluid helium is used to cool them down to 1.9 K.

Besides the center-of-mass energy, the luminosity is another important quantity for particle accelerators. The luminosity L determines the event rate and therefore, a high luminosity is desired:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \sigma L \quad , \tag{2.2}$$

where N is the number of events and σ is the cross section, given by the physical

laws described in chapter 1. The luminosity depends on the properties of the particle accelerator:

$$L = \frac{nN_1N_2f}{A} \quad . \tag{2.3}$$

Here, n is the total number of bunches in the collider, N_1 and N_2 are the number of particles in the colliding bunches, f is the circulating frequency for one single bunch and A is the cross-sectional area. As the luminosity is inversely proportional to the area, the beam must be focused as much as possible to reach a high luminosity. The total number of events can be derived by integrating equation 2.1 over time to

$$N = \sigma \int L\mathrm{d}t = \sigma\mathcal{L} \quad , \tag{2.4}$$

with the integrated luminosity $\mathcal{L}$. In 2016, a peak luminosity of $1.5 \cdot 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ was reached with an integrated luminosity of $41\,\mathrm{fb}^{-1}$ ($1\,\mathrm{b} = 10^{-24}\,\mathrm{cm}^2$) for the total year [19]. After Run 3 at the LHC, a high luminosity upgrade is planned in which the luminosity will be increased to $5 \cdot 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$.

There are four interaction points at the LHC. At each point, one detector is located. One is for the ALICE experiment that studies collisions of lead ions. Another one is for the LHCb experiment, which is designed for the studies of b hadron physics. The ATLAS and the CMS experiments are general-purpose experiments, measuring the standard model particles and searching for new elementary particles.

2.2 The Compact Muon Solenoid Detector

The construction of the Compact Muon Solenoid (CMS) detector [20] was motivated by the search for the Higgs boson and the search for physics beyond the standard model. The former one has already been found [1, 2], but the search for alternative theories like supersymmetry or extra dimensions is still ongoing. Besides of these two big tasks, the CMS detector is used for several other studies, like the measurement of the couplings of different particles [21]. In order to achieve these goals, the detector is required to measure the properties of all particles with best possible precision. A sketch of a slice of the CMS detector with its different sub-detectors is shown in figure 2.2. To store only the interesting events and reduce the amount of data to an amount that can be stored for further analyses, a filter system, the so-called trigger system, is applied. In this section, the sub-detectors and the trigger system are briefly described. At the beginning the coordinate system used within the CMS detector is introduced.

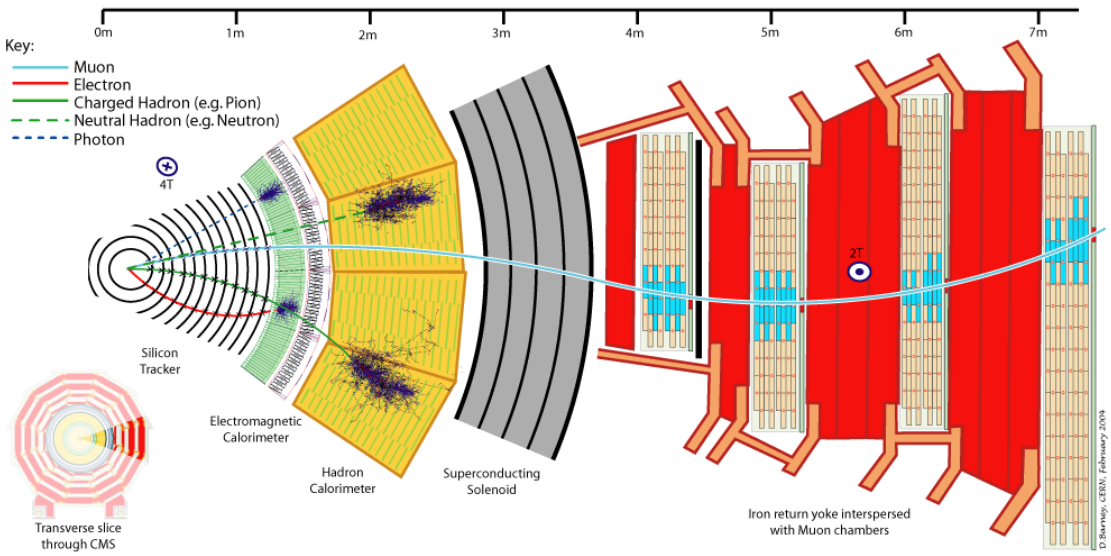


Figure 2.2: Slice of the CMS detector. The scheme shows the different detector systems as well as the tracks and the interactions of different particles with the detector [22].

Coordinate System

The origin of the coordinate system is chosen to be the intended collision point. The x axis lies in the plane of the collider ring and points to its center. The y axis points vertically upwards and the z axis points in beam direction in a way that a right handed coordinate system is given. The azimuthal angle ϕ is defined as the angle in the x-y plane measured from the x-axis. The radial distance is $r = \sqrt{x^2 + y^2}$. The polar angle θ is measured from the z axis. Another important quantity is the rapidity, which is a measure for the velocity of a particle and defined as

$$y = \operatorname{artanh}\left(\frac{v}{c}\right) \quad , \tag{2.5}$$

where v is the velocity of the particle and c is the speed of light. y is in contrast to v unlimited and can take values in the range of $-\infty$ to ∞, further one can just add two rapidities together instead of using the relativistic velocity-addition formula. The pseudorapidity η is a frequently used quantity in particle physics, in the relativistic approximation, η is equal to y and given by the equation

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \quad , \tag{2.6}$$

which is used to denote the angle between a vector and the z axis. A value of $\eta = 0$ is equal to $\theta = 90°$; an angle of $\theta = 0°$ equals $\eta = \infty$. The pseudorapidity is preferred over θ since the number of produced particles per η-interval is constant and

the difference between two particles $\Delta\eta$ is invariant under Lorentz boosts in z direction. The momentum and energy of the initial partons is not determined in a hadron collider. Because of this, mainly the quantities transverse to the beam axis are of interest. These quantities, the transverse momentum p_T and the transverse energy E_T are invariant under Lorentz boosts in z direction. They are defined as

$$p_\mathrm{T} = \sqrt{p_x^2 + p_y^2} \quad \text{and} \quad E_\mathrm{T} = \sqrt{E_x^2 + E_y^2} \quad . \tag{2.7}$$

Due to momentum conservation, the transverse momenta of all final particles sum up to zero. In the measurement there is always a missing amount of transverse momentum caused by gaps in the detector and neutrinos, which are not detectable within the CMS detector. The missing amount of transverse momentum is denoted as $p_\mathrm{T}^\mathrm{miss}$.

Silicon Tracker

From the collision of two partons, charged and neutral particles arrive at the so called primary vertex (PV). Heavy objects are formed which decay after a short time and distance at secondary vertices (SVs). The innermost part of the detector is of highest importance to resolve the PV and SVs as far as possible; it has to fulfill several requirements. First, a precise measurement of trajectories of charged particles is necessary to identify the PV and SVs. This also allows a momentum measurement as the full tracker is inside the superconducting solenoid, which produces a homogeneous magnetic field of 4 T. At each bunch crossing, about 1000 particles on average are created and travel through the detector. To distinguish these particles from each other and to assign the trajectories to the corresponding vertices, a high granularity is needed. Further, the time between two bunch crossings is about 25 ns, therefore the response time and the cooling-off period has to be short. Last, the detector components have to be resilient against this high amount of radiation.

To fulfill these requirements, silicon technology is used for the whole tracker system [23]. When charged particles travel through these materials, electron-hole pairs are produced, which in turn generate an electric current in the read-out electronics. In the innermost part from $r = 4.4\,\mathrm{cm}$ to 10.2 cm, three barrel layers of pixel detectors are surrounding the beam axis. Each pixel detector has a size of $100 \times 150\,\mu\mathrm{m}^2$. With this design, the desired impact parameter[1] resolution is achieved. The pixel detectors are surrounded by ten layers of silicon micro-strip detectors up to a radius of 1.1 m. As the particle flow decreases with the radius, larger detector elements can be used. This also reduces the amount of read-out electronics. The inner micro strip detectors have a size of $10\,\mathrm{cm} \times 80\,\mu\mathrm{m}$, whereas the size of the outer ones is $25\,\mathrm{cm} \times 180\,\mu\mathrm{m}$. At

[1] The impact parameter (IP) is defined as the distance of the closest approach of a track to the PV and can be measured in three spatial dimensions (3D), in the plane transverse to the beam line (2D) or in one dimension along the beam line (longitudinal).

the endcaps, two disks of pixel detectors, three smaller and nine larger discs of strip detectors are installed. This allows a reconstruction of charged particle tracks up to $|\eta| < 2.5$. In total, the silicon tracker has a length of 5.8 m with a diameter of 2.5 m. At each bunch crossing, the occupancy of the detectors is at about 2-3% for the inner micro strip detectors and 1% or lower for the other ones. The track resolutions for charged particles of $1\,\text{GeV} < p_\text{T} < 10\,\text{GeV}$ and $|\eta| < 1.4$ are typically 1.5% in p_T, 25–90 μm in the transverse and 45–150 μm in the longitudinal impact parameter [24].

Electromagnetic Calorimeter

The electromagnetic calorimeter (ECAL) [25] surrounds the silicon tracker system hermetically and measures the energy of electrons, positrons and photons. Other electromagnetically interacting particles leave tracks but are not fully absorbed in the ECAL. The ECAL consists of several elements, the pseudorapidity interval up to $|\eta| < 1.48$ is covered bu the barrel part while the endcap has a coverage of $1.48 < |\eta| < 3$. The ECAL consists of homogeneous lead tungstate ($PbWO_4$) crystals which is the absorber and scintillator material at the same time. When electromagnetically interacting particles propagate through it, they produce photons through bremsstrahlung, which in turn create electron-positron pairs through pair production. An electromagnetic shower emerges. When the energy of the electrons drops to a critical amount, they mainly excite the atoms, which emit photons with characteristic wavelengths between 360 nm to 570 nm. This process is called scintillation. The photons are then measured with photomultipliers. The number of photons is proportional to the deposited energy of the particle. The radiation length of $PbWO_4$ is $X_0 = 0.89\,\text{cm}$. This means that after this distance, the energy of an electron or positron is reduced on average by the factor of $1/e$. Moreover, a photon has an average free path length of $9/7\ X_0$ before it undergoes a process of pair production. The thickness of the ECAL with 23 cm is equivalent to 25.8 X_0, therefore all electrons, positrons and photons are absorbed completely.

The ECAL was designed to measure the $H \to \gamma\gamma$ decay as precise as possible. Therefore, it provides a good energy resolution. To identify the background in this decay channel, a preshower detector is installed between the tracker and the ECAL. This element helps to identify neutral pions that decay into two photons; it also improves the position determination of electrons, positrons and photons.

Hadron Calorimeter

Hadrons propagate through the ECAL losing only a small fraction of their energy. Therefore, a hadron calorimeter (HCAL) [26] surrounds the ECAL, which absorbs all hadrons and measures their energy. The HCAL is important for measuring the properties of hadron jets as well as the total energy of the collisions and the resulting missing

transverse energy. Several elements allow to measure forward jets up to $|\eta| = 5.2$.

The HCAL is a sampling calorimeter which exists of alternating absorbing layers and scintillating layers. The absorbing layer consists of brass and has a nuclear interaction length of λ_I =16.42 cm. This is the mean distance for a hadronic particle after which it undergoes an inelastic nuclear interaction. Similar to the ECAL, a chain reaction starts and a hadronic shower emerges. As the HCAL is located inside the solenoid, its size is limited. The total absorber material of ECAL and HCAL makes up less than 7 λ_I. For hadrons that are not fully absorbed, an additional outer hadron calorimeter is installed outside the solenoid. For the scintillating layers, plastic is used. When the hadronic shower hits these layers, the scintillator material gets excited and photons with characteristic wavelengths are emitted. They are guided through optical cables and measured by photodiodes.

Muon Detector

The muon detector [27] is the outermost part of the CMS detector and covers the pseudorapidity interval of $|\eta| < 2.4$ without any gaps. The only particles that reach this part of the detector are neutrinos, which cannot be detected directly with the CMS detector, and muons. Muons have a high mass and are often high energetic. Because of this, they lose only small parts of their energy by propagating through the inner parts and are usually not fully absorbed in the muon detector. Muon identification and position determination is done with gaseous detectors. As they pass through these detectors, muons are ionizing the gas atoms. The resulting electrons and ions can then be measured as an electric current. The muon system consists of an iron return joke to guide the magnetic field lines of the solenoid through this part of the detector. This creates a magnetic field of 2 T. As a consequence, muons travel a curved path. The momentum can then be determined by measuring the curvature of its track.

Trigger System

The LHC has a bunch crossing rate of about 40 MHz at each interaction point. This huge amount of data makes it impossible to store every event. Since interesting processes typically have low rates, it is sufficient for most analyses to store only a small fraction of promising events. To perform this data reduction, a trigger system [28] consisting of two filtering steps is used at the CMS detector.

The first filter is called the Level-1 (L1) trigger and is realized by hardware electronics. It decides in a few µs whether it is worth keeping an event. In this short time, only a part of the information from the calorimeters and the muon chambers can be used. The trigger checks if the information is consistent with a physics object like an electron

or a muon. This procedure reduces the event rate to less than 100 kHz.

The second filter step is done by the High-Level Trigger (HLT). The HLT is realized with software tools where complex calculations and a more precise reconstruction can be done. In addition, the information of the tracking system is now included. This trigger reduces the event rate further to the order of 1 kHz [29]. These events are then stored offline for more detailed analyses.

3 Simulation and Reconstruction of Events

All the available information about a particle collision is the measured deposited energy of each detector component and their position. Together with some results of the online reconstruction e.g. trigger signals, this information is stored for further reconstruction. In the offline reconstruction, physical objects like electrons and hadrons are reconstructed. To compare the measured data with the theory prediction, or to investigate the efficiencies and robustness of algorithms, a simulation is needed. In this chapter the simulation process is described followed by an explanation of the reconstruction process. Additionally, methods to correct differences between the measured data, shortly denoted as data, and the simulation are summarized.

3.1 Event Simulation

The simulation is done using as much theory input as possible. In the same order as the physical process, from the proton-proton scattering to the outgoing particles, one propagates the simulation. Due to insufficient theoretical descriptions, phenomenological models are used when necessary. After the outgoing particles are generated, the interaction with the detector materials is simulated. The emulation for the readout electronics generates signals that have the same signature as the data.

3.1.1 Proton-Proton Scattering

A proton-proton scattering process leads to the production of a final state that is one point in the phase space and is determined by the final state particles and their properties. The differential cross section is proportional to the probability to produce a final state that lies in a given phase space. The determination of the differential cross section $d\sigma$ is done in multiple steps, these steps are outlined in figure 3.1 and explained in the following.

Parton Distribution Functions

As described in section 1.3, the proton is not an elementary particle but consists of so-called partons. The valence quarks (uud) determine the quantum numbers of the proton. Apart from these, also gluons exist inside of protons as they are the exchange particles of the strong interaction. The gluons can split into virtual quark-antiquark-pairs; these quarks are the so-called sea quarks. In a deep inelastic proton-proton

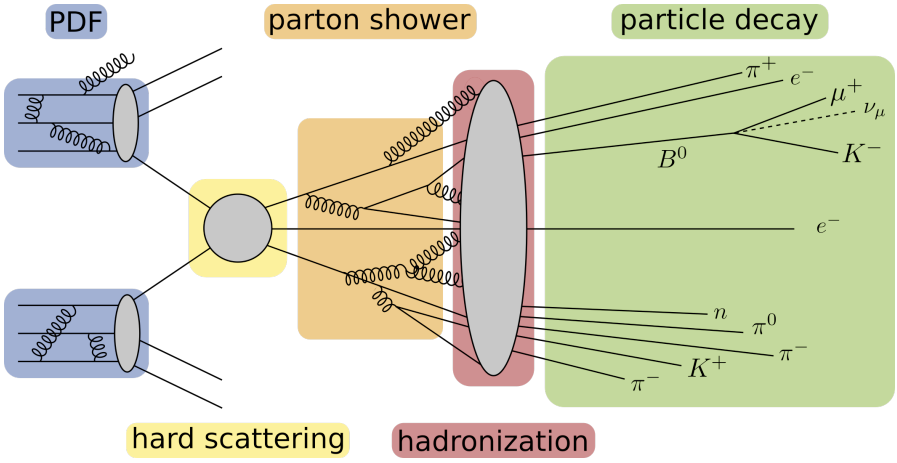


Figure 3.1: Proton-proton scattering process: The parton distribution functions (PDF) describe how the constituents of the proton are distributed. The deep inelastic scattering process of two elementary particles is calculated in the hard scattering. The subsequent processes are the parton shower, the hadronization and the particle decay.

scattering, two partons from two colliding protons interact with each other. Each parton carries a momentum fraction x of the proton. The probability that a specific parton interacts, depends on x and the absolute transfered momentum Q. It is given by a parton distribution function (PDF). As the PDFs cannot be computed by perturbation theory, they are determined by global fits to data from electron-proton scattering, fixed target experiments or hadron colliders. A classic approach is a polynomial ansatz for a specific PDF [30]. A more modern approach is using a three-layer feed-forward neural network [31]. One can define a factorization scale μ and absorb low energetic gluons into the PDF, thereby infrared divergences are avoided. The use of the Dokshitzer-Gribov-Lipatov-Altarelli-Parisi (DGLAP)-equations [32–34] allows the evolution of gluon and quark PDFs to a desired value of Q. The PDFs for two different values of μ are shown in figure 3.2.

Hard Scattering Process

The differential cross section $\mathrm{d}\sigma$ for the hard scattering process is weighted with the PDFs $f_{i,j}$, which give the probabilities to find certain partons with certain momentum fractions x and y of the colliding protons. It is given by

$$\mathrm{d}\sigma = \frac{1}{3} \sum_{i,j} f_i(x, \mu^2)\, f_j(y, \mu^2)\, \frac{1}{2(xys)^2}\, |\mathcal{M}_{i,j\to\text{final}}|\, \frac{\mathrm{d}\cos\theta\, \mathrm{d}\phi\, \mathrm{d}x\, \mathrm{d}y}{8(2\pi)^2} \quad , \tag{3.1}$$

with the square of the center-of-mass energy s and the transition amplitude $\mathcal{M}_{i,j\to\text{final}}$ for an initial state to a final state, the so-called matrix element. $\mathcal{M}_{i,j\to\text{final}}$ is obtained by summing up all Feynman diagrams for a specific process in a perturbation series.

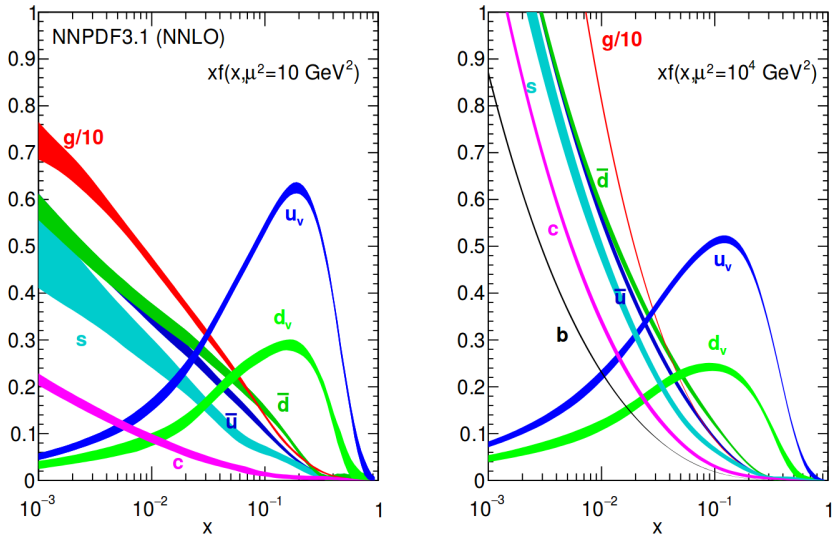


Figure 3.2: Neural Network Parton Distribution Functions (NNPDFs). Shown are the PDFs of gluons, valence quarks and sea quarks for different momentum fractions x of the proton momentum. At lower values of the factorization scale (left) the valence quarks carry the majority of the proton momentum. At higher values of factorization scale (right) gluons carry the dominant part. The PDFs are computed using the DGLAP-equations in next-to-next-to-leading-order (NNLO) [31].

This is possible because the momentum of the partons is in a regime where $\alpha_s \ll 1$ and convergence of the series is given. Often it is sufficient to take the leading order (LO) Feynman diagrams, which are the ones with the minimal amount of vertices that are needed for a specific process. Including next-to-leading order (NLO) Feynman diagrams improves the accuracy of the simulation but is more complex and computationally intensive in generation.

The simulation is done with the help of Monte Carlo (MC) methods [35]. MC methods can be used for numerical integration of higher dimensional phase spaces. In principle, to obtain one simulated event that obeys equation 3.1, the following can be done [36, p. 48]. Uniform distributed random numbers (θ, ϕ, x, y, i, j) are taken in the corresponding value ranges and $\langle d\sigma \rangle$ is calculated at one specific phase space point. The event is further processed if another uniform distributed random number g in the range $0 < g < d\sigma_{max}$ is lower than $\langle d\sigma \rangle$, where $d\sigma_{max}$ is the supremum of $d\sigma$. Otherwise it is discarded. Repeating this procedure equals an integration of the total phase space of equation 3.1.

Parton Shower, Hadronization and Particle Decay

In addition to the final-state particles of the hard scattering process, gluons and photons can be produced in initial- or final-state radiation. Gluons can radiate off another gluon or split into quark-antiquark pairs; photons can split into fermion pairs. These processes can recur and are described by the DGLAP-equations. A so-called parton shower emerges. Because of the confinement (section 1.3), the partons form hadrons. This process cannot be described in a perturbation series as the assumption of $\alpha_s \ll 1$ does not hold for lower energies of the partons. The effect is described by phenomenological models like the Lund string model [37]. The hadrons can further decay, as most of them are unstable.

Underlying Event and Pileup

Besides the hard scattering process of two partons, the remaining partons can interact themselves, producing additional signals in the detector at the same time. This effect is called underlying event. In the same bunch crossing, several other proton pairs are interacting with each other, this is called in-time pileup. Out-of-time pileup considers previous bunch crossings. The pileup distribution depends on the collider's properties. The more protons are in one bunch and the more focused the beams are, the more pileup events happen. In the data recorded in 2016, there were in average about 27 pileup events per bunch crossing in the CMS detector [19]. The pileup events can be simulated as additional random scatterings and merged with the main event.

3.1.2 Simulation Tools

In the event simulation, several tools are used for the different steps. For the matrix element, one common generator is MadGraph5 [38]. It sums up all Feynman diagrams in LO for given initial- and final-state particles. The matrix element is generated but not evaluated yet. The evaluation is done with the MadEvent package [39] at given phase space points with the MC method and the cross section is calculated. For this reason a simulated event is often called MC event.

MadGraph5_aMC@NLO [40] is a further developement of MadGraph5, combining the features of MadGraph5 with the ones of aMC@NLO, where aMC@NLO can generate matrix elements at NLO in QCD. These tools also provide different parton-shower matching schemes to combine the final-state particles with the parton shower. The parton shower itself has to be added with a parton shower simulation tool. As explained earlier, the parton shower can cause additional gluons in the final state, but such gluons can also be produced in NLO matrix-element generators. To avoid double counting, the parts of the parton shower that match the NLO Feynman diagrams are subtracted. This is implemented by using negative event weights.

Another matrix-element generator that includes Feynman diagrams up to NLO in QCD is POWHEG [41–43]. POWHEG uses a different method than aMC@NLO to avoid double counting and negative event weights. It demands a parton shower simulation that generates the emission with the highest p_T first and then corrects the NLO emission.

PYTHIA [44] is a general purpose tool for the simulation of the hard process as well as the parton shower, the hadronization, the particle decay and the underlying event. The hard process is only calculated in LO, but PYTHIA provides several interfaces to external programs. Therefore, the matrix-element generators explained above can be used in combination with PYTHIA.

To be able to compare simulation with data, the detector response has to be simulated as well. This is done with the GEANT4 [45] toolkit. In GEANT4, the different materials of the detector system, including dead materials, are modeled. The particles are propagated through these materials while taking into account the effect of the magnetic field on charged particles. Interactions like bremsstrahlung, multiple scattering and photon pair-production are simulated. The generated signals in the active detector materials are taken as input for emulators of readout electronics and trigger systems. After this step, the detector information of a simulated event is in the same manner as for data. To get the best comparison between data and simulation, the event reconstruction of the detector signals is done in the same way as for data. This will be subject of the next section.

3.2 Event Reconstruction

In this chapter, the aim to reconstruct physics objects out of the measured or simulated detector information is described. In a first step of the event reconstruction, particle tracks and the corresponding interaction points, called vertices, are reconstructed. In several subsequent steps, the identification and reconstruction of particles that penetrate the detector is done. These are electrons, muons, photons, charged hadrons, neutral hadrons and neutrinos. Afterwards, adjacent particles are bundled to so-called jets.

3.2.1 Tracks and Vertices

Tracks of charged particles are reconstructed from the tracker information. Therefore, tracker hits are created from clustering signals of the silicon pixel and strip detectors. The tracks are produced using an iterative track finding algorithm. From tracker hits in the inside of the detector, where the occupancy of each detector element is low, tracker seeds are created. The track finding is performed with the Kalman filter approach [46].

In the first iteration step, the seeds have tight constrains. This gives only a moderate efficiency, but also a minimal fake rate. The energy deposits of the corresponding hits are then removed, which simplifies the track finding problem. At the next iteration step with looser seeding criteria, a higher efficiency is achieved. Repeating this, one gets an efficiency to find isolated muons of 99.5% and more than 90% to find charged hadrons while the fake rate is of about 1% [47]. Each track has several quality criteria like the number of hits, a normalized χ^2 value and the compatibility to a certain vertex.

Because of pileup, there exists more than one vertex for each event. The vertex from where the hard scattering originates is the primary vertex (PI), it has to be identified in order to remove the tracks originating from pileup. The vertex reconstruction starts with a set of possible vertices and each track is extrapolated to a point z_i on the beam line. For each track and vertex pair a probability c_{ik} is estimated that the track is originating from the regarded vertex. A χ^2 function, weighted with these probabilities

$$\chi^2 = \sum_{i,k} c_{ik} \frac{(z_i - z_k)^2}{\sigma_i^2} \quad , \tag{3.2}$$

is minimized to determine the exact position of the vertices z_k. As the probability depends on the vertex position, an iterative procedure is used. After a final selection, a set of vertices is obtained. The primary vertex is selected as the one with the highest sum of p_{T}^2 of its tracks.

3.2.2 Particle Reconstruction

For reconstructing particles as precise as possible, the combination of all subdetectors is used within the particle flow (PF) algorithm [48]. The identification of PF candidates, namely electrons, muons, photons, charged hadrons and neutral hadrons is explained in the following.

Calorimeter Clustering

The tracker system gives the most precise information about the position and through the curvature of the track, it gives also a determination of the momentum of charged particles. The calorimeters support the determination of the energy and direction of charged particles and are decisive in cases of high p_{T} or low quality tracks. Moreover, the energy and direction of photons and neutral hadrons is measured while the contributions of charged particles are separated. This is done in the calorimeter clustering process, where each calorimeter element, the ECAL barrel, the ECAL endcap and the several HCAL elements, are handled separately. The process starts on calorimeter cells with a local maximum of deposited energy. Then, adjacent cells are combined, if their deposited energy exceeds a certain threshold. For each calorimeter element, one can

have several calorimeter clusters. Together with the charged-particle tracks and the muon tracks, PF elements can be constructed.

Electrons

An electron is identified if a charged-particle track matches an ECAL cluster with a typical electron shower profile. A fit starting with a track seed combines both signals and the momentum is determined. A complimentary algorithm for electrons with high p_{T} starts with taking a seed from the ECAL. In this algorithm, several ECAL clusters are combined to so-called ECAL superclusters which have a characteristic shape for electrons [49]. The quality of an electron fit is described by several quantities like the matching of the track to the cluster or the distance of the track to the primary vertex. A tight requirement on these quantities can lower the fake rate.

Muons

For the muon reconstruction, tracks in the muon chamber are computed in a similar way as in the tracker. These muon tracks are propagated towards the beam line and combined if possible with the most suitable track of the inner tracker. A refit using the information of both sub detectors gives an improved muon track. A reconstruction process for low p_{T} muons that do not create a muon track, starts with propagating an inner tracker track and tries to find a minimal muon information in the muon tracker. Again one can choose looser or tighter demands on muon tracks afterwards to further reduce the fake rate.

Hadrons and Photons

For the reconstruction of charged hadrons, the energy depositions of clusters in ECAL and HCAL are compared to the momentum of the charged particle tracks. In case of an agreement with one or more tracks, a combined fit is performed. In case of disagreement, several scenarios including additional neutral hadrons or photons are considered. After all charged-particle tracks are matched to PF elements, their energy deposits are removed and neutral particles are identified. Neutral particles do not produce any hits in the tracker, therefore they have less precise direction information and an assignment to a vertex is not possible. Because of this, they cannot be distinguished from neutral particles coming from pileup.

3.2.3 Jet Reconstruction

A jet is a particle shower of hadrons, leptons and photons that originates in the most cases from a single particle. There are different ways to reconstruct a jet, the default method used in the CMS experiment collects several nearby PF elements in a cone up

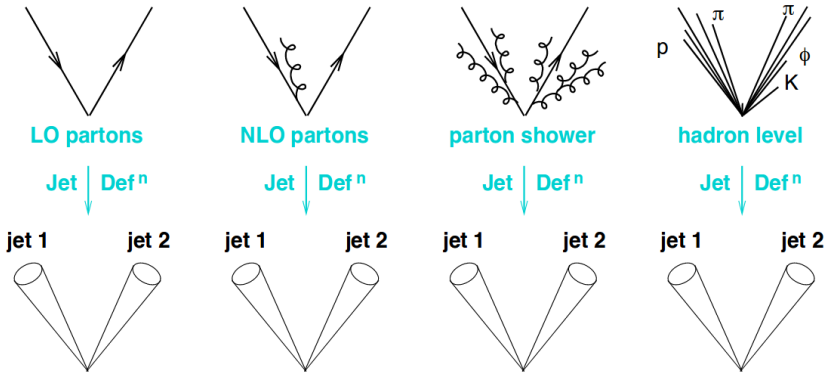


Figure 3.3: Infrared and collinear safety. A jet can be defined on different levels. In a theoretical description, it originates from one single particle and can therefore defined as this particle. In perturbation theory, several vector bosons can be radiated in addition. On the detector level, the jet is defined by color neutral objects. With an infrared and collinear safe jet clustering algorithm, the resulting jets are independent of the level of definition [51].

to a given size in a sequential recombination process. This is done with the anti-k_t jet clustering algorithm [50].

An advantage of this algorithm is its infrared and collinear safety. This means, that the same jet is found with and without soft or collinear radiation, for example a gluon radiation or a gluon splitting. Therefore, this algorithm is independent of the technical level of the jet definition as shown in figure 3.3. For the clustering process, a distance measure d_{ij} between two particles i and j is defined as

$$d_{ij} = \min(k_{t,i}^{-2}, k_{t,j}^{-2}) \frac{\Delta R_{ij}^2}{R^2} \quad , \tag{3.3}$$

where $k_{t,i}$ is the momentum of a particle, $R^2 = (\mathrm{d}y^2 + \mathrm{d}\phi^2)$ is a fixed parameter that determines the maximal size of the jet in the y-ϕ plane with the rapidity y and $\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$ is a radial distance of two particles. A distance measure d_{iB} between a particle i and the beam axis is defined as

$$d_{iB} = k_{t,i}^{-2} \quad . \tag{3.4}$$

For the general jet reconstruction of data taken from run 2 of the CMS experiment, a radius of $R = 0.4$ was chosen. In the sequential recombination algorithm, two particles with the smallest d_{ij} are combined to new pseudo-particles. This is repeated until d_{iB} is the smallest distance. The remaining pseudo-particles become then the jets. In this way, the hardest particles (the ones with the highest p_T) are combined with nearby soft particles (the ones with low p_T). If two hard particles have a distance of $R < \Delta R_{ij} < 2R$, they split the softest particles in between among them and create two separate jets.

3.2.4 Lepton Isolation

Muons and electrons often appear inside of QCD jets originating from b or c quark decays. Isolated leptons on the other hand arrive from electroweak processes in the hard scattering process or, less frequent, from leptonic top quark decays ($t \to b + \nu_l + l^+$). Therefore, an isolation criteria is useful to find events with bottom quarks without directly looking at them. An isolation can be described using the momenta and energies p_T^{charged}, E_T^{neutral} and E_T^{γ} of the PF elements for the charged hadrons, the neutral hadrons and the photons respectively inside a cone of

$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} = 0.4 \quad . \tag{3.5}$$

Since the tau lepton decays in the beam line, only for an electron or a muon with its momentum p_T^l an isolation can be defined as

$$I_l^{\text{PF}/\Delta\beta} = \frac{\sum p_T^{\text{charged}} + \max\left(0.0, E_T^{\text{neutral}} + \sum E_T^{\gamma} - 0.5 \sum E_T^{\text{charged}}(\text{PU})\right)}{p_T^l} \quad . \tag{3.6}$$

The so-called $\Delta\beta$ correction is included where contributions of the neutral particles from pileup interactions are subtracted. They are estimated as half of the energy coming from charged particles in pileup $\sum E_T^{\text{charged}}(\text{PU})$. A lower value of $I_l^{\text{PF}/\Delta\beta}$ corresponds to a more isolated electron or muon.

3.2.5 Reconstruction of Neutrinos and Leptonically Decaying W Bosons

At the LHC, the transverse momentum $\vec{p}_T = (p_x, p_y)^T$ of the colliding particles is zero, therefore the sum of the $\vec{p}_T$ of all generated particles after the collision has to be zero as well. As neutrinos cannot be measured within the CMS detector the missing part of the transverse momentum can be attributed to neutrinos. For historical reasons and as the momentum is equal to the energy in the relativistic regime using natural units, the missing part is denoted as the missing transverse energy

$$\vec{\not{E}}_T = -\sum_i \vec{p}_{T,i} \quad . \tag{3.7}$$

Decays of the W boson into a charged lepton and a neutrino are called leptonic W boson decays. With the measured $\vec{\not{E}}_T$ and the momentum of the lepton, the transverse mass of the W boson $m_{T,W}$ can be reconstructed

$$m_{T,W} = \sqrt{(\vec{p}_{T,l} + \vec{p}_{T,\nu})^2 - (p_{x,l} + p_{x,\nu})^2 - (p_{y,l} + p_{y,\nu})^2} \quad , \tag{3.8}$$

with $\vec{p}_{T,\nu}$ the neutrino momentum set to $\vec{\not{E}}_T$. This method allows to reconstruct the W boson if there are no additional neutrinos or leptons in the event.

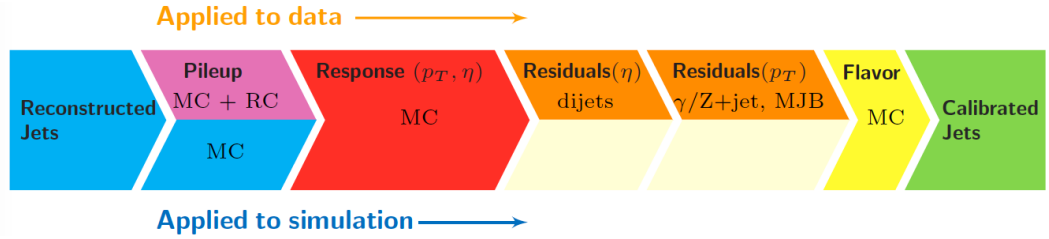


Figure 3.4: Jet Energy Corrections. In the first step, based on information from simulation, energy contributions from pileup and electronics noise are removed. In data, the residual differences between data and simulation as a function of η is flattened as well. Next, the response of the detector is corrected. Therefore, the residual differences in simulation between the reconstructed p_T and η and the corresponding true values are flattened out. The resulting correction is performed in data and simulation. A further step is done only on data, here small residual differences between data and simulation in p_T and η are corrected. An optional last step is the correction in dependence of the flavor of the jet using the truth information from the simulation [53].

3.3 Differences between Data and Simulation

As mentioned before, the simulation is not exact since in each step smaller or larger inaccuracies are made. These differences lead to discrepancies between data and simulation after the reconstruction. Several steps can be done to improve the agreement afterwards, these are explained in the following.

3.3.1 Jet Energy Corrections

In general, the measured jet energy does not match the true parton energy, where the jet originated from. One reason is pileup and underlying events. Due to the tracks, the charged particles from pileup can be removed reasonably well but the neutral particles not. Also neutrinos are not measured and they appear also in jets, therefore, their energy is missing. Moreover, there is a disagreement in data compared to simulation due to inaccuracies in the simulation. With jet energy corrections (JEC) [52], these differences are mitigated in several steps with some variation for data and simulation. These steps are outlined in figure 3.4.

3.3.2 Pileup Reweighting

The simulation is done before or during the actual data taking period, therefore a pileup distribution has to be assumed in the simulation. To correct most of the disagreements caused by the different pileup interactions, the simulated events can be assigned with a weight in a way that the pileup distributions in data and simulation are equal. The pileup weights are given by

$$w_{\mathrm{PU},i} = \frac{n_{\mathrm{data},i}}{n_{\mathrm{MC},i}} \quad , \tag{3.9}$$

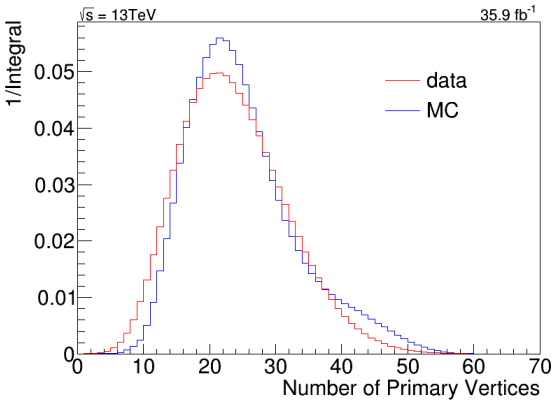


Figure 3.5: Pileup distribution for data and simulation. Shown is a normalized histogram of the events as a function of the number of primary vertices. In each bin where the data is above the simulation, the simulation gets up-weighted; when the data is lower than the simulation, the simulation gets down-weighted.

with the number of data events $n_{\text{data},i}$ and the number of simulated events $n_{\text{MC},i}$ in dependency of the number of primary vertices i. The pileup distributions for the 2016 data and simulation are shown in figure 3.5.

3.3.3 Trigger and Lepton Efficiencies

The efficiency of the triggers is also different in data and simulation. In order to mitigate this discrepancy, scale factors are used to correct the simulation

$$\text{sf}_{\text{trigger}} = \frac{\epsilon_{\text{data}}(p_{\text{T}}, \eta, \dots)}{\epsilon_{\text{MC}}(p_{\text{T}}, \eta, \dots)} \quad , \tag{3.10}$$

where the efficiencies $\epsilon_{\text{data,MC}}$ are functions of different quantities. For example a muon trigger has a strong dependency on the p_{T} of the corresponding muon.

Lepton scale factors are needed whenever a selection of leptons is performed as the efficiencies of the identification of a lepton are different in data and simulation. The efficiencies are moreover dependent of the selection of the lepton, they are often separated for different parts of the selection (for example for the tracker selection and the isolation criteria) and can be combined. For some selection criteria, scale factors are already measured.

4 Artificial Neural Networks

Artificial neural networks (ANNs) are a class of algorithms inspired by the biological brain and are widely used in machine learning and pattern recognition. In this thesis, ANNs were used for a multivariate classification task, this problem is outlined in section 4.1. ANNs consist of several artificial neurons, in section 4.2, the functionality of a single artificial neuron is explained. In section 4.3, the kind of ANNs used in this thesis are described. The core subject of this thesis is the domain adaptation as an extension of ANNs to improve its data-to-simulation agreement. The domain adaptation is described in section 4.4. A comprehensive introduction to machine learning techniques is provided in the references [54, 55].

4.1 Multivariate Classification

In a multivariate classification task one tries to assign an element with characteristic variables to one of several classes. If the distributions of the characteristic variables overlap for different classes, the elements are not completely separable. That means only the probability that one element belongs to a certain class can be obtained. Consider two classes, denoted as class one and class two. Taking the output of a classifier, a so-called test statistic can be constructed to separate the two classes. The test statistic can be chosen on a fixed value such that all elements with a test statistic higher (lower) than this value are said to be class one (two). Due to the fact that the classes are not completely separable, not all elements can be classified correctly. Consider the identification of class one elements, the true positive rate (TPR) is the fraction of elements from class one that are correctly classified. The false positive rate (FPR) is the fraction of class two elements that are mistakenly classified as class one. The closer the TPR is to 1 and the closer the FPR is to 0, the better the classifier performance.

The multivariate classification in this thesis is done with ANNs by constructing a function that maps a vector $\vec{x}$ of input features[1] to an output vector $\vec{y}$ that describes the probabilities of the classes. An ANN has various free parameters. In a process called training, these parameters are adjusted to yield a function with high TPR and low FPR.

[1] Feature is in the field of machine learning the common description of a characteristic of an element from a statistical population. The term feature is used for the measured characteristics as well as transformations of them, also when performed by machine learning algorithms.

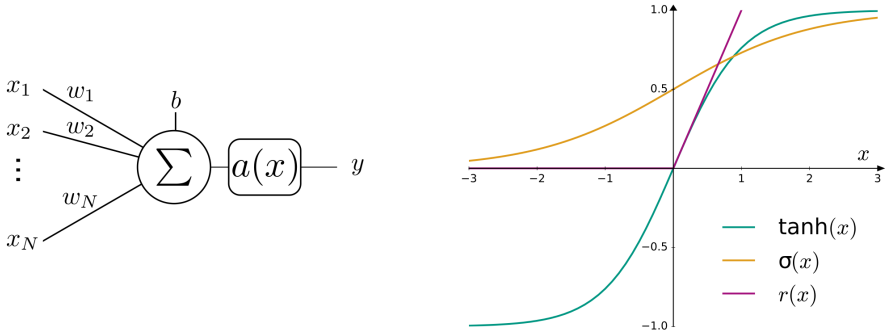


Figure 4.1: Perceptron and activation functions. An ANN consisting of one neuron as shown on the left is called perceptron. Several neurons can be connected to a multilayer perceptron. Three common activation functions are shown on the right.

The process of training is done using so-called training samples. For each element of these samples the corresponding class is known, therefore the $\vec{y}$ is labeled.

4.2 Perceptron

One single neuron can be used to construct simple mathematical functions. It was first introduced by Rosenblatt and is called the Rosenblatt perceptron [56]. The computation is done by taking all input values x_i, multiplying them with adaptable weights w_i and adding an adaptable bias b. The result is mapped by an activation function a to one single output value

$$y = a\left(\sum_i x_i w_i + b\right) \quad . \tag{4.1}$$

Commonly used activation functions are the sigmoid function

$$\sigma(x) = \frac{1}{1+\exp(-x)} \quad , \tag{4.2}$$

and the hyperbolic tangent $\tanh(x)$. However, these are problematic when applied to more complex neural networks [57], therefore the rectifier activation function (ReLu) [58]

$$r(x) = \max(0, x) \quad , \tag{4.3}$$

is preferred for them. The structure of a perceptron and the different mentioned activation functions are shown in figure 4.1. This perceptron is able to solve linearly separable problems. For more complex problems, an ANN consisting of several neurons has to be used.

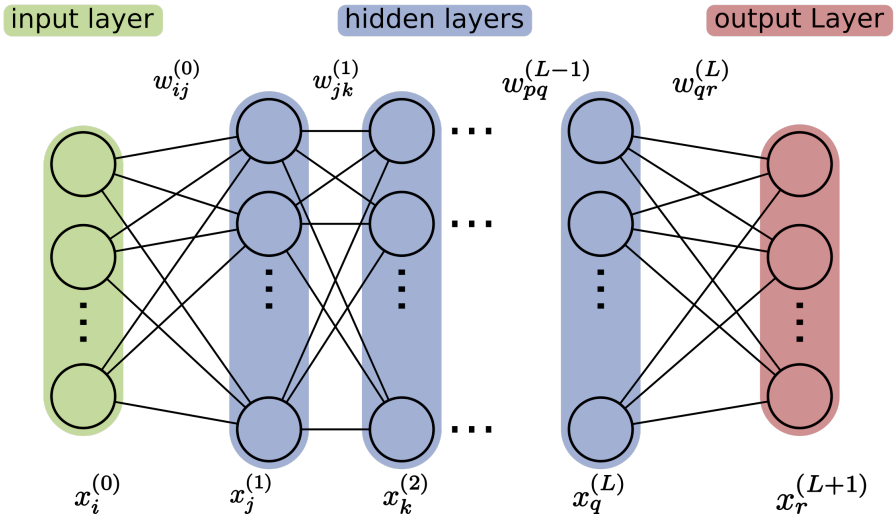


Figure 4.2: A multilayer perceptron with $L + 2$ fully connected, feed-forward layers. The first layer (green) is called the input layer and the last layer (red) is called the output layer. In between there are L layers (blue), referred to as hidden layers. The circles represent the neurons while the connections between them are the edges. Each edge has a weight w_{ij} where i and j indicate its source and target neuron respectively. Each neuron has a bias, which is not shown and delivers a value x_i.

4.3 Multilayer Perceptron

A multilayer perceptron (MLP) is the standard type of ANN where several neurons are aligned in layers. An MLP can be further distinguished according to its architecture. The architecture describes the configuration of the MLP, it determines the number of layers, the number of neurons of each layer, how the neurons are connected among each other and the activation functions used. With the use of an appropriate architecture and training procedure, the output values can take on meaningful values.

4.3.1 Fully Connected Feed-Forward Networks

If the MLP has a feed-forward structure, the connections between neurons do not form circles. It is additionally fully connected, if each neuron takes input values from all neurons of the preceding layer while the output values are connected to all neurons of the following layer. An example of a fully connected feed-forward MLP is shown in figure 4.2. The number of input and output neurons is determined by the number of input features and classes of the specific multivariate classification task. The number of hidden layers and the number of neurons in each hidden layer can be chosen freely. These determine the number of free parameters, the biases and the weights. MLPs with more than one hidden layer are called deep neural networks (DNNs). The output of an

arbitrary neuron can be calculated recursively by

$$x_i^{(k+1)} = a_i^{(k+1)} \left(\sum_{j=1}^{N^{(k)}} x_j^{(k)} w_{ij}^{(k)} + b_i^{(k)} \right) \quad , \tag{4.4}$$

where k denotes the layer and $N^{(k)}$ the number of neurons in this layer. As an activation function in the final layer, it is reasonable to use a softmax function

$$y(\vec{x})_k = \frac{\exp(x_k)}{\sum_{j=1}^{N} \exp(x_j)} \quad , \tag{4.5}$$

to normalize the sum of all output values to 1. With the presented architecture, the input values can be propagated through each layer and the output values can be calculated. This process is called forward propagation.

4.3.2 Convolutional Layers

When several groups of input features exist that have the same structure, the use of shared weights between the features of the different groups reduces the amount of free parameters and can lead to a less complex and more stable algorithm. This concepts is applied in convolutional layers [59]. Convolutional layers are widely used in image recognition to find specific patterns, regardless of their positions. But they can also be used to subsample a high amount of input features to some few features without loosing a lot of information.

The convolutional layer is applied on input features, arranged in one or more dimensions. The input for each convolutional neuron is computed through a discrete convolution of adjacent input features with a convolution kernel, the so-called filter. Each filter has a kernel size that determines how many features are included in one convolution. Convolutions are performed stepwise over the input features until all input features have been convoluted; for each step, one output neuron is generated. The free parameters are determined by the filter, which means all output neurons share the same set of weights for one filter. Several filters can be used, resulting in independent sets of shared weights and corresponding output neurons. As usual, each output neuron can take an additional adjustable bias and maps the sum of its input values to an activation function. A two-dimensional convolutional layer is illustrated in figure 4.3 (a).

For historical reasons from image recognition, convolutional layers are typically implemented with an additional dimension, the so called channels. The convolution is then done for all neurons in the channel dimension. This means a 1D convolutional layer with a channel dimension bigger than one takes a two dimensional input. In this thesis, convolutional layers are used for subsampling to compress several features to a

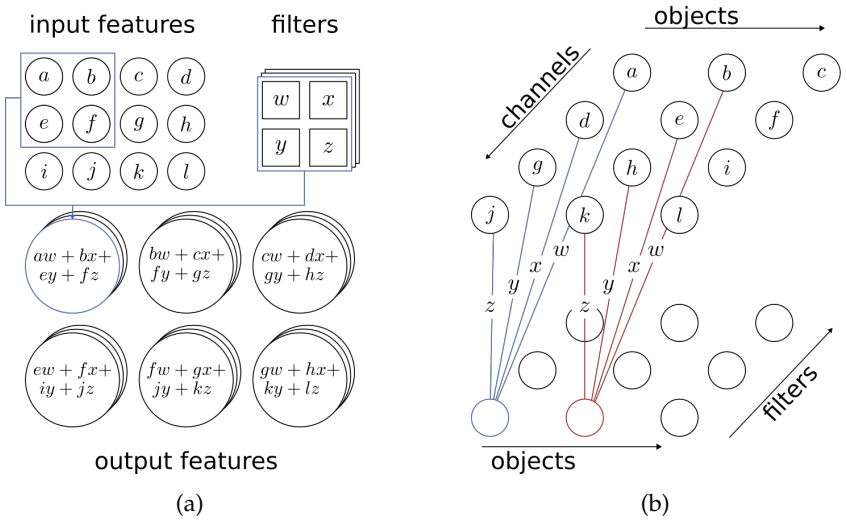


Figure 4.3: Convolutional layers. In the left example, a kernel size of 2×2 was taken, each filter has therefore four adjustable weights. The illustrated case with four times three input features results in six output neurons for each filter. A subsampling can be achieved with the right architecture. In this case, the input features are grouped in three objects. The kernel size is 1×1 and four channels for the four input features of each object are used. Thereby, the input features of each object are subsampled to the amount of filters used.

few ones without loosing much information. Therefore, 1×1 convolutional layers are used, these are 1D convolutional layers with a kernel size of 1. Using these layers, the features in the channel dimension, are reduced to an amount of features equal to the number of used filters. An example is shown in figure 4.3 (b).

4.3.3 Long Short-Term Memory Layers

Neural networks consisting of long short-term memory (LSTM) layers belong to the type of recurrent neural networks. LSTM layers take several sequences of features as input, while previous sequences can influence later ones. Each LSTM unit contains a memory cell for storing information. Furthermore, it consists of three gates with adjustable parameters to regulate the use of the memory cell. The output of each LSTM unit depends on all units of one LSTM layer, the computation is done layerwise.

The input gate of each LSTM unit determines to which degree a new input vector is influencing the memory cell. The values of these gates in an LSTM layer are given by the input gate vector

$$i_t = a_g \left(W_i x_t + U_i h_{t-1} + b_i \right) \quad , \tag{4.6}$$

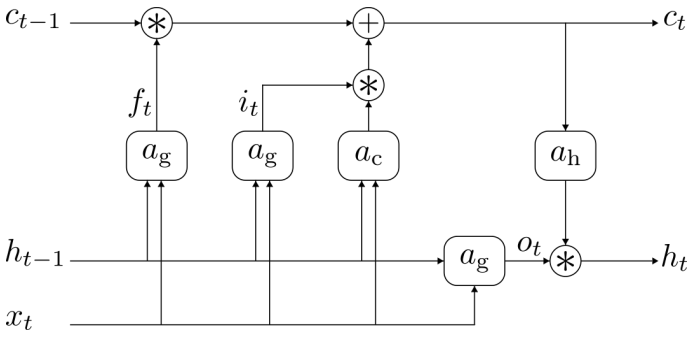


Figure 4.4: Long short-term memory (LSTM) unit. The LSTM unit transports the input vector x_t and the output vector of the last sequence h_{t-1} to three gates. The input gate i_t determines how much a new value influences the memory cell c_t, the forget gate f_t describes the degree the value of c_t is forgotten and the output gate defines the impact of c_t in the output.

where t indicates the sequence, x_t is the input vector of features from the current sequence, h_{t-1} is the output vector of the LSTM layer from the last sequence, W_i and U_i are adjustable weight matrices, while b_i is an adjustable bias vector. The a_g is the activation function, used for all gates and mostly chosen as the sigmoid function.

The forget gate controls to which degree the old value of the memory cell will be retained, the corresponding forget gate vector f_t is computed as before with independent weights and biases

$$f_t = a_\mathrm{g}\left(W_\mathrm{f} x_t + U_\mathrm{f} h_{t-1} + b_\mathrm{f}\right) \quad . \tag{4.7}$$

The output gate combines the input value with the value of the memory cell to provide an output value. The output gate vector o_t at these gates is

$$o_t = a_\mathrm{g}\left(W_\mathrm{o} x_t + U_\mathrm{o} h_{t-1} + b_\mathrm{o}\right) \quad . \tag{4.8}$$

The memory cell vector c_t is updated for each sequence through

$$c_t = f_t * c_{t-1} + i_t * a_\mathrm{c}\left(W_\mathrm{c} x_t + U_\mathrm{c} h_{t-1} + b_\mathrm{c}\right) \quad , \tag{4.9}$$

where $*$ denotes a discrete convolution. For the memory cell, again an independent set of parameters is used. The activation function a_c is typically chosen as the sigmoid function. The output vector h_t of the LSTM layer is given by

$$h_t = o_t * a_\mathrm{h}(c_t) \quad , \tag{4.10}$$

with the activation function a_h is typically chosen as the hyperbolic tangent function. An example of the described LSTM unit is shown in figure 4.4.

Note that the most information can be extracted from the last sequence, therefore, the sequences have to be ordered in a way that the most important sequence is inserted last. Different architectures of LSTM layers exist, the one described in this section is the type used in this thesis.

4.3.4 Training

The training is done by minimizing a function that takes the computed output values of the ANN for given training samples and measures the deviation from the corresponding labels, which is the desired outcome. This function is called the error function, it calculates one single value, the so-called loss.

Consider a binary classification problem where the ANN has one single output neuron with a sigmoid activation function. Moreover, each element of the training sample has a label l with $l = 1$ for one class and $l = 0$ for the other. An appropriate error function is the binary cross-entropy

$$E(\vec{x}; \vec{w}, \vec{b}) = -\sum_{n=1}^{N} \left(l_n \ln y_n + (1 - l_n) \ln(1 - y_n)\right) \quad , \tag{4.11}$$

where n indicates an elements of the training sample with N elements in total and $y_n = y_n(\vec{x}; \vec{w}, \vec{b})$ the computed output value of the ANN. One can show that the binary cross-entropy together with the sigmoid activation function (equation 4.2) is equivalent to the negative log likelihood function [54], for this kind of problem.

Now assume a multiclass classification problem where the labels $\vec{l}$ are in the 1-of-K coding scheme, which means that each entry is binary ($l_k \in \{0, 1\}$) and is 1 for the corresponding class and 0 otherwise. A good choice for the error function is the categorical cross-entropy

$$E(\vec{x}; \vec{w}, \vec{b}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} l_{nk} \ln y_{nk} \quad , \tag{4.12}$$

where k indicates one of K classes. Given that the softmax function (equation 4.5) is used in the output layer, one can show in a similar way to the before mentioned binary cross-entropy that one obtains the negative log likelihood function [54]. This allows to interpret the calculated output values as probabilities for the different classes. For this reason, the calculated output values can be called predictions for the specific classes.

The high-dimensional error functions are in general too complex to compute the position of the global minimum. Typically, the global minimum cannot be found. Instead, it is aimed to find a local minimum that has a low value and is flat[2]. This is done by a minimizing algorithm, the so-called optimizer.

A simple optimizer is the gradient descent. The error function is calculated for all training samples and the gradient is computed with respect to the weights and biases.

[2]The flatness leads to better generalizability as for small deviations of the minimum the loss value is still low.

The free parameters $\vec{\theta} = (\vec{b}, \vec{w})^{\mathrm{T}}$ are then updated against the gradient

$$\vec{\theta} \quad \leftarrow \quad \vec{\theta} - \eta \frac{\partial E}{\partial \vec{\theta}} \quad , \tag{4.13}$$

with a so-called hyperparameter, the learning rate η that determines the step size of the update. This is repeated several epochs. An epoch is defined as a full training cycle, which is when every element in the sample is processed. When the gradient is zero, a minimum or a saddle point is found. To minimize the risk of getting stuck in saddle points or shallow local minima, the training samples are split into batches and the updates in equation 4.13 are done for each batch. After each epoch, the samples are shuffled. This is called stochastic gradient descent (SGD) and has also the advantage that broader minimums are found, which are more generalizable.

Widely used is also the Adam optimizer [60] that takes previous gradients into account and leads to a faster convergence with less fine tuning of free hyperparameters. In this case, the updates are given by

$$\vec{\theta} \quad \leftarrow \quad \vec{\theta} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \vec{\hat{m}} \quad , \tag{4.14}$$

with

$$\begin{aligned} \vec{\hat{m}} = \frac{\vec{m}}{1-\beta_1} \qquad & \vec{m} \quad \leftarrow \quad \beta_1 \vec{m} + (1-\beta_1) \frac{\partial E}{\partial \vec{\theta}} \quad , \\ \hat{v} = \frac{v}{1-\beta_2} \qquad & v \quad \leftarrow \quad \beta_1 v + (1-\beta_1) \left(\frac{\partial E}{\partial \vec{\theta}} \right)^2 \quad . \end{aligned} \tag{4.15}$$

The hyperparameters are the learning rate η, a damping parameter ϵ and the decay rates β_1 and β_2. The decay rates β_1 and β_2 determine how much the past gradients effect the current batch update. In practice, the default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ proposed by [60] can be used and only an appropriate η has to be chosen. When a stationary point is obtained, a fine tuning of the optimizer can be done by a method called reduce-on-plateau. In this method, the learning rate of the optimizer is reduced if after a certain number of epochs, the training loss has not decreased.

The updating of the free parameters $\vec{\theta}$ is performed with the help of error backpropagation. Starting at the output layer, the chain rule can be applied:

$$\frac{\partial E}{\partial \theta_{ji}} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial \theta_{ji}} \quad , \tag{4.16}$$

where a_j is the activation function of the corresponding output neuron. By re-applying the chain rule, the gradients for the free parameters in the hidden layers can be calculated.

4.3.5 Regularization

Regularization methods are used to prevent the training process from overfitting. Overfitting is an effect where the ANN approximates the training data too close and gets sensitive to statistical fluctuations. As a consequence it loses the generalizability to make predictions for other data that originates from the same distribution. There are several different methods, the ones used in this thesis are explained in the following.

A common regularization method in the field of machine learning is called early stopping [61]. The available data is split into two parts, the training dataset and the validation dataset. After each epoch of training, the so-called validation loss is calculated by evaluating the loss function on the validation dataset. The ANN parameters, for which the validation loss is minimal, are stored. Even if the training loss further decreases in later epochs, the stored parameters for which the validation loss is minimal are applied after completion of training.

Additionally one can monitor the validation loss instead of the training loss for the above explained reduce-on-plateau method.

Another regularization method used in ANNs is dropout [62]. During the training, randomly selected hidden neurons are deactivated for each batch update. In this way, one single neuron does not carry the total information for a certain prediction and the ANN spreads the information over several neurons. As a result the ANN is less able to memorize single samples and becomes more generalizable. The fraction of neurons that are deactivated in each batch update is given by the dropout rate and can be chosen freely.

A regularization method that has a similar effect is the use of batch normalization layers [63]. They normalize each neurons output value by subtracting the mean μ_B and dividing by the standard deviation σ_B^2 of the corresponding batch B. Afterwards, it multiplies the output by an adjustable scale factor γ and adds another adjustable shift factor β. The output is then given by

$$y_i = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \quad , \tag{4.17}$$

where ϵ is a small constant value that prevents the fraction of a division by zero. With this it is possible to scale and shift the output value by only changing two values. Without batch normalization, several weights would have to be adjusted to achieve the same. It prevents individual neurons from becoming too important if there is no general gain for the optimizer. Moreover, it speeds up the training since the input values are more in the scope of the common activation functions.

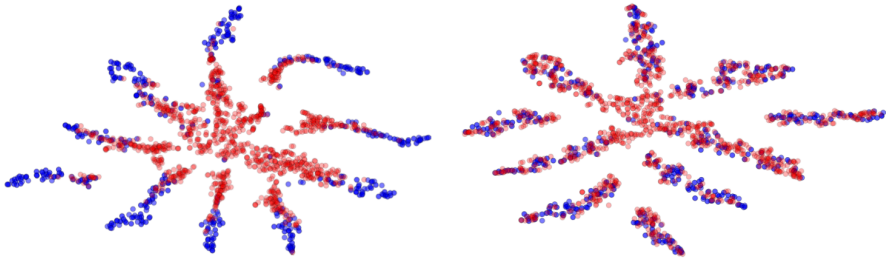


Figure 4.5: The effect of domain adaptation. The blue (red) points represent samples from the source (target) domain. Each point represents the position of an element in a two-dimensional feature space. Shown are features from the last hidden layer of a DNN classifier. In the left plot no domain adaptation is performed. The source samples form separated groups while the target samples are more centered and less separated. In the right plot, the domain adaptation is used during the training. Source and target samples cover the same areas, the predictions of target samples are more reliable [67].

4.4 Domain Adaptation in Deep Neural Networks

For the training of a DNN, a huge set of labeled samples is needed, but often not available. The DNN is therefore trained on a similar, but differently distributed set of samples where labels are known, the so-called source domain. When applied to the unlabeled set of samples, the so-called target domain, it results in worse performance and reliability for the predictions. The objective of unsupervised domain adaptation is to transfer the target domain and the source domain into one common domain-invariant feature space. Such a feature space is exemplified with and without domain adaptation in figure 4.5.

Various methods for unsupervised domain adaptation methods have been proposed [64]. For example reweighting the source domain [65] or using an explicit feature space transformation [66]. In this thesis, two promising methods were tested on a multiclass classification task to mitigate the difference between simulated events and measured data, the source and target domains respectively. Compared to the previous mentioned methods, the following ones can be implemented straightforwardly to complement the existing DNN. They both aim at reducing the domain discrepancy in a latent feature space, but they use different mechanisms. The first method, the moments method, is realized through a modified loss function that punishes the domain discrepancy. The second method, the domain-adversarial method, uses an extension of the DNN via additional network components to implicitly achieve this goal by backpropagation. Both methods are explained in the following.

4.4.1 Moments Method

Two distributions are the same if the moments of these distributions are identical. The first moment of a distribution is the mean

$$\mu = E[X] = \frac{1}{N} \sum_{\{x \in X\}} x \quad , \tag{4.18}$$

with the expectation value $E[X]$, the elements x from the empirical distribution X and the total number of elements N . Central moments are defined as

$$\mu^{(k)} = \frac{1}{N} \sum_{\{x \in X\}} E\left[\left(x - E[X]\right)^k \right] \quad , \tag{4.19}$$

where (k) denotes the order of the moment. The second central moment is the variance. Higher central moments are normalized by the standard deviation to the power of k

$$\hat{\mu}^{(k)} = \frac{\mu^{(k)}}{\sigma^k} \quad . \tag{4.20}$$

These moments can be used to measure the domain discrepancy in certain features. A simple measure is the maximum mean discrepancy (MMD) [68]

$$\mathrm{MMD}_i(X_\mathrm{S}, X_\mathrm{T}) = \left| \mu_i(X_\mathrm{S}) - \mu_i(X_\mathrm{T}) \right| \quad , \tag{4.21}$$

where i denotes the feature and S (T) denotes the source (target) domain. The MMD is used in many domain adaptation approaches, examples are given in the references [69, 70]. It can be used as an additional term in the loss function to construct a domain-invariant feature space. The resulting loss function is given by

$$L = L_\mathrm{C}(X_\mathrm{S}, y) + \lambda \sum_i \mathrm{MMD}_i^2(X_\mathrm{S}, X_\mathrm{T}) \quad , \tag{4.22}$$

where the $L_\mathrm{C}(X_\mathrm{S}, y)$ is the classification loss with the labels y of the samples from the source domain. The hyperparameter λ determines how strong the domain discrepancy is punished. This approach is generic, one can use any features, for example the ones of some hidden layer or the output layer. The training is performed with samples from both domains at the same time while only samples from the source domain can be used for the training of the classification.

A better measure for the domain discrepancy includes higher central moments. The loss can be defined as

$$L = L_\mathrm{C}(X_\mathrm{S}, y) + \alpha \sum_i \mathrm{MMD}_i^2(X_\mathrm{S}, X_\mathrm{T}) + \beta \sum_i \left(\hat{\mu}_i^{(2)}(X_\mathrm{S}) - \hat{\mu}_i^{(2)}(X_\mathrm{T}) \right)^2 + \dots \quad , \tag{4.23}$$

with the hyperparameters α and β. The dots represent the possibility to include arbitrary higher moments. The contribution of higher moments are typically smaller, to include the first moment and the second central moment only is often sufficient.

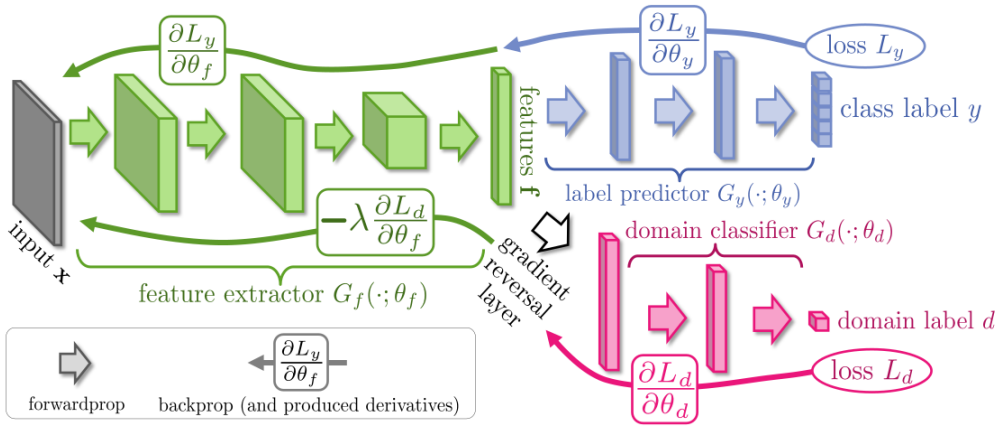


Figure 4.6: The domain-adversarial method. The input features $\vec{x}$, the feature extractor (green) and the label predictor (blue) form a conventional DNN. Additionally, a domain classifier (red) is added. For a more detailed description refer to the text. Source: [67]

4.4.2 Domain-Adversarial Method

The domain-adversarial method (or domain adaptation by backpropagation) was introduced in reference [67, 71], the proposed architecture is shown in figure 4.6. The method is realized through an extension to the original feed-forward DNN. The original DNN is divided into two parts: the first part is the feature extractor $G_f(\cdot;\theta_f)$[3] which maps the input features to a latent feature vector $\vec{f}$ and forwards them to the second part. The second part is the label predictor $G_y(\cdot;\theta_y)$, which predicts the class labels y. The additional part is the domain classifier $G_d(\cdot;\theta_d)$, which is used to distinguish between the source domain and the target domain. As the label predictor, the domain classifier takes $\vec{f}$ as its input features. The first layer is a gradient reversal layer (GRL) followed by an arbitrary number of feed-forward layers and one binary output neuron. The GRL has no free trainable parameters, it is the identity function in forward propagation. During backpropagation, the gradient is multiplied by the negative constant $-\lambda$.

During training, samples from the source domain and target domain are used at the same time. As a loss function for the label predictor, denoted as class loss L_y, the categorical cross-entropy can be used. The class loss takes only the labeled samples from the source domain into account, backpropagation is used to update the free parameters θ_y of the label predictor and the free parameters θ_f of the feature extractor. For the domain classifier, a binary cross-entropy can be used as a loss function, denoted as domain loss L_d. The domain loss includes samples from both domains. The free parameters θ_d of the domain classifier are updated by backpropagation in a way that the discrimination between the domains is improved. Through the gradient reversal

[3] The dot in $G_f(\cdot;\theta_f)$ denotes the corresponding set of input features.

layer, the gradient is flipped and the θ_f are updated towards a maximum of the domain loss. As a result $\vec{f}$ becomes indistinguishable between source and target domain, this is called $\vec{f}$ becomes domain-invariant. This also leads to domain-invariant class labels. The updated free parameters can be computed using a standard optimizer. For the SGD, they are given by

$$\begin{aligned} \theta_f &\leftarrow \theta_f - \eta \left(\frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f} \right) \quad , \\ \theta_y &\leftarrow \theta_y - \eta \left(\frac{\partial L_y}{\partial \theta_y} \right) \quad , \\ \theta_d &\leftarrow \theta_d - \eta \left(\frac{\partial L_d}{\partial \theta_d} \right) \quad . \end{aligned} \tag{4.24}$$

Note that this is done implicitly through the use of the GRL, no further changes have to be done to the optimizer. The first and third equation in 4.24 show that θ_d minimizes the domain loss, while θ_f maximizes it. The factor λ is a trade-off factor between these two mechanisms. The domain classifier can therefore also be seen as an adversarial network, similar to the generative adversarial network (GAN) approach in reference [72].

5 Heavy-Flavor Jet Tagging

Jets that originate from the heavy-flavor quarks c and b have characteristic properties compared to jets that originate from gluons or light-flavor quarks, which are u, d, and s quarks. The properties are essentially determined through the bound state hadrons formed by the quarks and are explained in section 5.1. These properties can be used to distinguish jets originating from different quark flavors (tagging). This is done using a fully connected feed-forward MLP in the DeepCSV algorithm, which is explained in section 5.2. A more advanced DNN is used by the DeepFlavour algorithm that is described in 5.3.

5.1 Heavy-Flavor Jet Properties

Hadrons containing b (c) quarks have a lifetime in the order of 1.5 ps (1 ps) [73] in their rest frame. Depending on their momentum, this results in a flight distance of some mm up to cm in the CMS detector. After this distance they decay at a secondary vertex (SV) where daughter particles arise. For the SVs characteristic properties can be derived, for example the vertex mass, which is the invariant mass of all particles originating from the corresponding SV. Another important variable is the flight distance (FD), defined as the distance from the PV to the SV. The FD can be measured in three spatial dimensions (3D), in the plane transverse to the beam line (2D) or in one dimension along the beam line (longitudinal). The daughter particles of a SV tend to have larger IPs compared to tacks that originate from the PV, they have displaced tracks. These aspects are outlined in figure 5.1. The IP can be measured similar to the FD in 3D, 2D or in longitudinal direction. The higher mass of hadrons containing b or c quarks compared to hadrons consisting of only light-flavor quarks results in daughter particles with higher momentum perpendicular to the jet axis. Furthermore, b or c quark decays often happen in flavor-changing charged currents under the emission of an electron or muon. Therefore, b (c) jets contain an electron or a muon in about 20% (10%) [73] of all cases. Since b quarks decay mostly first in c and than in light quarks (see equation 1.5), they have in general several SVs and a higher probability of containing electrons or muons.

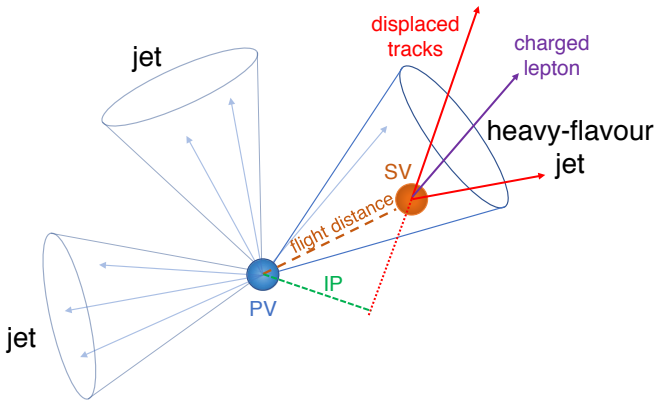


Figure 5.1: A heavy-flavor jet. Several jet properties can be used to identify a heavy-flavor jet. The main identifying properties are secondary vertices (SV) and displaced tracks with an impact parameter (IP) towards the primary vertex (PV) [73].

5.1.1 Secondary Vertex Reconstruction

The reconstruction of the SV is done with the inclusive vertex finding (IVF) algorithm [74]. The IVF algorithm takes all tracks with $p_T > 0.8\,\text{GeV}$ and longitudinal IP $< 0.3\,\text{cm}$ into account. In a first step, tracks with IP and the IP significance[1] values, which exceed certain thresholds are identified as seed tracks. Next, tracks are clustered if they form a compatible SV. The clustered set of tracks is then used to fit a SV. Afterwards, SVs are only kept if their 2D FD significance (same definition as IP significance) is larger than 2.5 and their 3D FD significance is larger than 0.5 and no other SV is compatible with the tracks. Furthermore tracks are removed if ΔR between the track and the SV flight direction is larger than 0.4. After this cleaning, a repeated SV fitting is performed followed by a repeated cleaning. The remaining SVs have a FD significance of 10 or more and less than 20% of their tracks shared with other SVs.

5.2 The DeepCSV Algorithm

The DeepCSV algorithm used in this thesis consists of a DNN with five hidden layers with 100 neurons each. The DNN uses up to 66 input features to achieve a good separation of each jet into four different categories. This results in 66 neurons in the input layer and 4 neurons in the output layer. The implementation is done using the high-level API Keras [75] with the machine learning framework TensorFlow [76] as back end.

[1] The IP significance is the IP value divided by the standard deviation of the IP value.

For the hidden layers, ReLu activation functions are used while the last layer has softmax activation functions. The loss function is the categorical cross-entropy. For each hidden layer, a dropout rate of 0.1 is applied (see section 4.2 and 4.3).

5.2.1 Categories

In this thesis, jets were distinguished in four categories, which are b, bb, c and udsg. Jets containing two b hadrons are categorized as bb. Jets containing one b hadron are categorized as b. Jets in which no b hadron is present, but at least one c hadron, are categorized as c. If neither a b hadron nor a c hadron exists, the jet is categorized as udsg. Jets that originate from pileup remain uncategorized (they get a label with zeros only). The simulated jets are labeled corresponding to their categories in the 1-of-K coding scheme (as in section 4.3.4).

5.2.2 Input Features

The DeepCSV algorithm makes use of the properties described in section 5.1. It takes into account global jet properties with twelve features, six charged particle candidates with seven features each, four neutral particle candidates with one feature each and one secondary vertex with eight features. The particle tracks from PF elements and the reconstructed SVs are first selected by demanding some quality criteria. Afterwards, the six most displaced charged tracks and the four most displaced SVs are taken as input. For the neutral particles, the ones with the highest p_{T} are selected. The used variables and their description are found in reference [73].

During the training, the convergence of the DNN is faster if all input features have the same mean and variance. Therefore, each input feature i is rescaled by using its mean value μ_i and standard deviation σ_i. The normalized input values $\hat{x}_i$ are then given by

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i} \quad , \tag{5.1}$$

and have a mean value of 0 and a standard deviation of 1. If a feature is missing for a certain jet, a default value is assigned that is not overlapping with the core of the feature distribution and close to zero.

5.3 The DeepFlavour Algorithm

The DeepFlavour algorithm [77] is a further development of the DeepCSV algorithm. In contrast to the DeepCSV algorithm, the selection of the tracks and vertices are much looser and much more candidates with more features of each PF candidate are included.

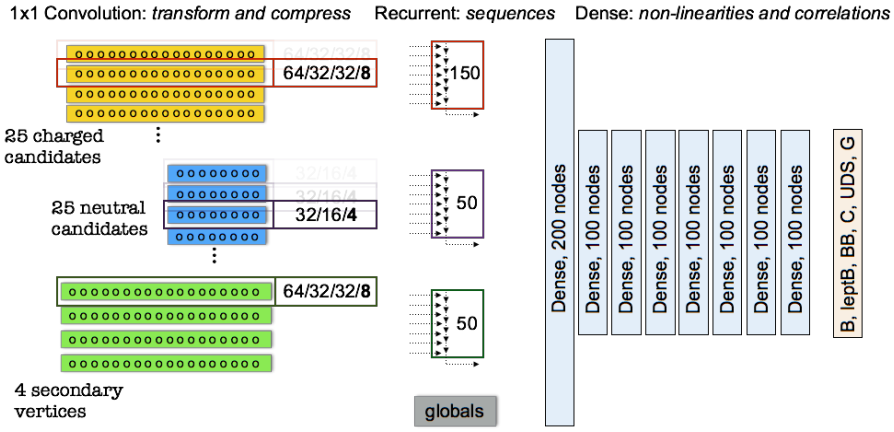


Figure 5.2: Architecture of the DeepFlavour algorithm. Charged particle tracks, neutral particle tracks, SVs and global jet information are inserted separately in the network and compressed through $1x1$ convolutional layers. The subsequent use of recurrent sequences leads to a more flexible structure with respect to the number of objects that are present in the jet. Together with the global features, the correlations between the object can be extracted. Source: With kind permission of Jan Kieseler

Moreover, the number of PF candidates varies for each jet. The construction of high-level variables with strong separation power is to a greater degree left to the neural network. This requires an alternative flexible network structure that compresses the high amount of information. More information of jet tagging with advanced machine learning tools can be found in reference [78].

5.3.1 Architecture

The network structure of the DeepFlavour algorithm is shown in figure 5.2. The features are divided into four groups, namely the charged candidates, the neutral candidates, the secondary vertices and the globals. The first three groups consist of several objects of which each object has the same features. Therefore, in a first step, these objects are subsampled to reduce the high amount of input features.

Each group is arranged separately as a two-dimensional feature layer. The first dimension represents the different features while the second dimension represents the various objects. A maximum of 25 charged particles, 25 neutral particles and 4 secondary vertices is given, but there is no issue for jets with less objects. The two-dimensional layer is inserted into a $1 \times n$ convolutional layer where n is the number of features of each object. For the charged particles and SVs, four $1 \times n$ convolutional layers are connected in a row, they use 64, 32, 32 and 8 filters respectively. The neutral particles are processed in three $1 \times n$ convolutional layers with 32, 16 and 4 filters each. Each two-dimensional output of the last $1 \times n$ convolutional layer is then inserted into an LSTM layer. The LSTM layer takes as sequences the various objects with the corresponding set of fea-

tures as input. 150 LSTM units were taken for the charged particles and 50 each for the neutral particles and SVs. The outputs of the three LSTM layers from the three categories respectively are concatenated together with the global features into one flat layer. These are connected to a fully connected layer with 200 neurons, followed by seven fully connected layers of 100 neurons each.

The LSTM layer uses sigmoid activation functions for the gates and the memory cell. A hyperbolic tangent activation function is used for the output of each LSTM unit. For the other hidden layers, ReLu activation functions are used. For the output layer, the softmax function was taken as usual. The loss function is again the categorical cross-entropy and dropout layers are applied after each hidden layer. Additionally, a batch normalization layer is applied after each hidden layer (see section 4.2 and 4.3).

5.3.2 Categories

The categories are similar to the ones used in the DeepCSV algorithm with the difference that the DeepFlavour algorithm distinguishes between light quarks (u, d, s) and gluons (g), where the physics definition is used as explained in reference [79]. Moreover it differentiates between b hadron jets where the b hadron decays leptonically or hadronically. The labeling is done in the same way as in the DeepCSV algorithm.

5.3.3 Input Features

Compared to the DeepCSV algorithm, the set of input features has been increased. 15 global features, 16 features for each charged particle, 6 features for each neutral particle and 12 features for each SV were used. For the sequential processing of the LSTM layers, the objects have to be ordered. Charged PF candidates and SVs are sorted with respect to the 2D IP significance in descending order. Charged PF candidates that were used in the PV fit are sorted in ascending order of the $\Delta R(\text{cPF,SV})$ value (ΔR of the charged PF candidate and its closest SV in the jet). For jets without SV, the p_{T} of the charged PF candidate is used in descending order as a third criterion. Neutral PF candidates are sorted as a first criterion in ascending order by their $\Delta R(\text{nPF,SV}$ value and as a second criteria descending by their p_{T}.

5.4 Performance of b Tagging Algorithms

To measure the performance of a tagging algorithm, the so-called receiver operating characteristic (ROC) curve can be used. The ROC curve is computed by varying the test statistic (section 4.1) of the classifier output from its minimal to its maximal value. At each point the TPR and the FPR are calculated. The ROC curve is commonly computed

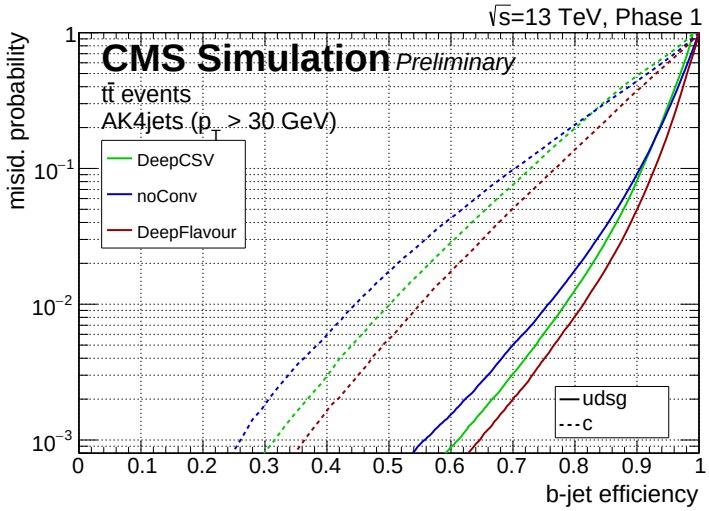


Figure 5.3: Performance of b tagging algorithms. The plot shows the misidentification probability (which is the FPR) against the b tagging efficiency (which is the TPR) measured on simulated samples. The TPR for b tagging against light quarks and gluons (continuous lines) and against c quarks (dashed lines) are shown for three different tagging algorithms. The noConv algorithm has the worst performance, it is used for comparison only and has the same input features and architecture as the DeepFlavour algorithm, but without the convolutional layers. The DeepCSV is the currently proposed b tagging algorithm for CMS analyses. The DeepFlavour has the best performance, it is under development and little is known about the performance on real data [77].

on simulated samples with the available labels. For the DeepCSV and DeepFlavour algorithms, the ROC curves are shown in figure 5.3. The classifier is applied at a fixed value of the test statistic, the so-called working point. The working point is defined by the FPR. The loose working point (L) has a FPR of 10%, the medium working point (M) has a FPR of 1% and the tight working point (T) has a FPR of 0.1%.

The training of the DeepFlavour Algorithm was performed with simulated QCD and $t\bar{t}$ events for 2016 data. The $t\bar{t}$ samples were produced using the Powheg generator, subsequent processes were simulated by the Pythia package while the detector simulation was done with Geant4. The QCD events were simulated with Pythia only. The fraction of gluons was reduced that a total amount of 80M jets were available for training, validation and testing. During the training a dropout rate of 0.2 was applied. The Adam optimizer was used with a start learning rate of 0.001. The learning rate was reduced during training with the reduce-on-plateau method. Afterwards, a layer wise retraining was performed with very low learning rates. The training of the DeepCSV algorithm was performed on similar samples, the performances presented in this section were produced using a slightly different architecture as the one used in this thesis, for more information refer to [73].

To calculate the TPR in data, more complex methods have to be applied. Since these methods have model assumptions and the number of jets from data events is much more limited, they have high systematic and statistical uncertainties. With the TPR on data ϵ_{data} and on simulated jets ϵ_{MC} with identical FPRs, a scale factor can be computed as

$$\text{sf}_b = \frac{\epsilon_{\text{data}}}{\epsilon_{\text{MC}}} \quad . \tag{5.2}$$

Scale factors for the DeepCSV and CSVv2 algorithm, which is a well established b tagging algorithm in the CMS experiment, computed with various methods are shown in figure 5.4. As expected, the scale factor is in the most cases lower than one, this originates from the fact that the algorithms are trained on simulation and have therefore worse performance on data. Desirable is a scale factor close to one, since in this case, analyses would not necessarily have to apply scale factors for jet identification algorithms. This would simplify analyses which in turn saves a high amount of man-power. Also important is a scale factor with low uncertainty. However, this is compatible with the first point as scale factors which are far from one tend to have higher uncertainties. For the DeepFlavour algorithm, there exists no official measurement of the scale factors, but scale factors with higher uncertainties are expected since the tracks and vertices used by the DeepFlavour algorithm undergo less quality requirements and are therefore less well modeled compared to the ones used by the DeepCSV algorithm.

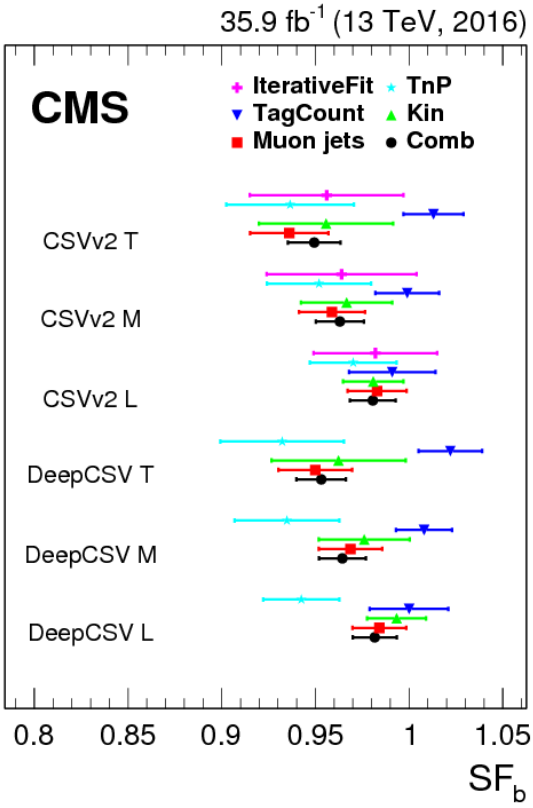


Figure 5.4: Scale factors for b tagging algorithms. The scale factor for the b tagging algorithms at different working points are shown for five methods. The measured scale factors agree in their uncertainties. The black dot refers to the combined value and is mainly dominated by the Muon Jets method [73].

6 Event Selection

The subject of this thesis is to study the effect of domain adaptation methods on flavor tagging algorithms to improve the data-to-simulation agreement. In section 4.4, two domain adaptation methods were introduced. Both have the goal to transfer the source domain and the target domain into a domain-invariant feature space that the classifier output becomes domain invariant. To achieve this goals, the source and the target distribution of their input features have to be similar. Furthermore, both domains should have the same distributions in their class compositions, which means in this case they need to have the same amount of jets of each quark flavor in data and simulation. In order to meet these requirements, an event selection is needed.

The event selection has several requirements to fulfill. It has to be performed in the same way for data and simulation and the data-to-simulation agreement for the selected events shall be as good as possible. The mismodeling of jets in the simulation with respect to the reconstruction of jets in data shall be the dominant reason for the domain discrepancy. The event selection has to be independent of the specific jets in each event. Moreover, the selected events should contain balanced fractions of jets of the different categories used in the classifier. Finally, the amount of jets should be high enough that the statistical uncertainties are negligible. The last two points are in direct conflict, as categories like bb occur mostly for high-p_{T} jets of which not many exist. Therefore, the demand of balanced fractions is secondary.

6.1 Top Quark Decay

A good choice is to search for top quarks as they decay almost every time into a b quark and a W boson. Moreover, the top quark pair production ($\mathrm{t\bar{t}}$) is under investigation by many analyses (for example reference [80, 81]), these can be taken into comparison when making a selection. The W boson decays either into a lepton and a neutrino, or into a quark-antiquark pair. Top quarks are mainly produced in pairs at the LHC with a cross section of $\sigma_{\mathrm{t\bar{t}}} = 831\,\mathrm{pb}$ at $\sqrt{s} = 13\,\mathrm{TeV}$ [82]. Their decay can be categorized by the decay of the W bosons as the potential lepton is a good indicator for those events. If both W bosons decay into quarks, it is called a hadronic decay. If both W bosons decay into leptons, it is called a dileptonic decay. If one W decays into a lepton while the other into quarks, it is called semileptonic decay. The decay rates for the W boson and $\mathrm{t\bar{t}}$ pairs are listed in table 6.1. The leading Feynman diagrams for top quark pair

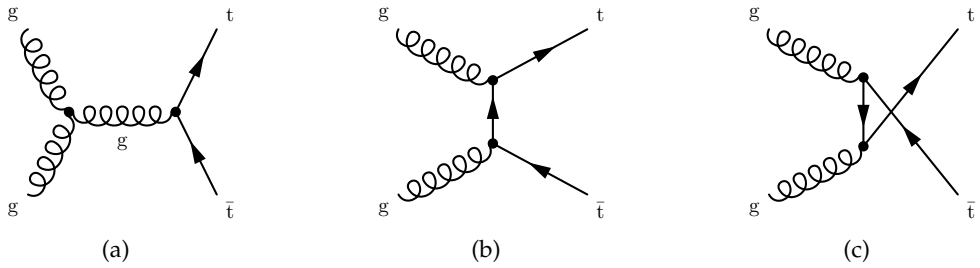


Figure 6.1: Dominant processes of top quark pair production at the LHC. Top quark pair production by gluon fusion is the dominant process at the LHC. It occurs in the *s*-channel (a), in the *t*-channel (b) and in the *u*-channel (c).

Table 6.1: Decay modes of W bosons and top quark pairs. Shown are measured branching ratios [14]. In almost 50 % of the cases of a hadronic W decay, a c quark occurs, the X denotes either a d or s quark.

Particle	Decay mode	Branching ratio in %
W^+	$e^+ + \nu_e$	10.71 ± 0.16
W^+	$\mu^+ + \nu_\mu$	10.63 ± 0.15
W^+	$\tau^+ + \nu_\tau$	11.38 ± 0.21
W^+	$q + \bar{q}'$	67.41 ± 0.27
W^+	$c + X$	33.3 ± 2.6
$t\bar{t}$	hadronic	45.7
$t\bar{t}$	semileptonic	43.8
$t\bar{t}$	dileptonic	10.5

production are shown in figure 6.1.

The hadronic $t\bar{t}$ decay mode has the highest branching ratio but is inappropriate for a selection as only jets are in the final state and other processes have the same signature and occur far more often. The semileptonic $t\bar{t}$ decay mode has also a relatively high branching ratio and with the lepton in the final state, a good indicator for this process exists. As listed in table 6.1, semileptonic $t\bar{t}$ decays also provide c quarks. Therefore, selection criteria are applied to enrich samples of the semileptonic $t\bar{t}$ decay mode. This is described in section 6.3. The dileptonic $t\bar{t}$ decay mode has a much lower branching ratio. But events of this category, where one W boson decays into an electron and the other, charge conjugated W boson, decays into a muon, produce the clearest signature.

6.2 Particle Identification

The τ decays before it can be directly identified, therefore it is less suited for a selection. The main identifying objects are electrons and muons. In a first selection step, quality criteria are applied on these particles and an isolation is required. The set of remaining muon and electron candidates have a low fake rate and can be used for the decay mode selection. Moreover, jets have a significant fake rate as for example photons are sometimes misidentified as jets. This requires a further selection.

6.2.1 Muon Selection

Muon-quality requirements are used in many analyses, several requirements are combined in the so-called muon IDs. In this thesis the muon is required to fulfill the requirements of the tight muon ID. The tight muon ID contains requirements on the number of hits of the muon track in various detector components or a minimum value of χ^2 divided by the number of decreases of freedom of the fit. Apart from this, in this thesis muons with $|\eta| > 2.4$ are discarded as these are not fully covered from the muon chambers. A minimum p_T value is required as the modeling of low-p_T muons is worse, the p_T cut has a different value in each selection. Additionally, an isolation criterion of $I_l^{PF/\Delta\beta} < 0.15$ is applied as described in section 3.2.4. The set of selected muons is denoted as good muons.

6.2.2 Electron Selection

For the electrons, similar requirements are applied by the tight electron ID. Furthermore, to ensure the best performance of the silicon tracker, the electron is required to have a value of $|\eta| < 2.4$. However, there is a gap in the ECAL between the barrel and the endcap part, therefore electrons in the interval of $1.4442 < |\eta_{SC}| < 1.5660$ are excluded, where η_{SC} refers to the position of the ECAL supercluster of the electron. Additionally, cuts on the IP have to be performed to reduce pileup contributions. The transverse IP has to be lower than 0.05 (0.1), while the longitudinal IP has to be lower than 0.1 (0.2) for electrons with $|\eta_{SC}| < 1.4442$ ($|\eta_{SC}| > 1.5660$). Furthermore, a $p_T > 20\,\text{GeV}$ is required due to the mismodeling of low-p_T electrons. The set of selected electrons is denoted as good electrons.

6.2.3 Jet Selection

The jet selection is performed by requiring the jet $p_T > 30$ as jets with lower p_T have an increased probability to come from the underlying event or pileup. A value of $|\eta| < 2.4$ is required to ensure the full capability of the detector. A neutral electromagnetic energy fraction < 0.99 is demanded to exclude photons reconstructed as jets, and a charged electromagnetic energy fraction < 0.99 is demanded to exclude electrons. Furthermore,

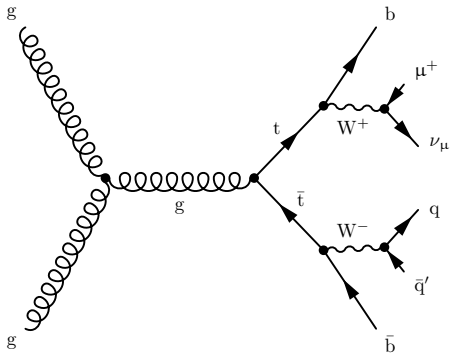


Figure 6.2: Semileptonic decay of a top quark pair. The process is in LO of QCD with respect to the final-state particles. To be unbiased of the selected jets, no selection for b jets is done. The quarks from the W boson decay are preferably light-flavor or c quarks. The charged conjugated process is also possible.

it is required a neutral hadron energy fraction < 0.99 to exclude single neutral hadrons and HCAL noise reconstructed as jets. Jets with a charged hadron energy fraction zero are excluded as jets should have charged hadrons. The sum of charged and neutral candidates has to be bigger than one while at least one charged candidate has to be present. Finally, jets are rejected if a good electrons or a good muons has a closer distance than $\Delta R = 0.4$. The set of selected jets is denoted as good jets.

6.3 Top Quark Pair - Semileptonic Selection

The final-state particles of a semileptonic $t\bar{t}$ decay in LO in QCD are one lepton, one neutrino, two b quarks, one up-type and one down-type quark. The best signal is obtained in the case where the lepton is a muon, the selection criteria are

- one good muon with $p_T > 30$,
- no additional good muon or good electron,
- four good jets with no further requirements on the jets,
- a transverse mass of the W boson $m_{T,W} > 50\,\text{GeV}$ to take the neutrino into account,
- a trigger selection, this will be explained later in this section.

As an example, a semileptonic $t\bar{t}$ process that satisfies this selection is shown in figure 6.2. The main contributions from other processes are from W boson production in association with jets and Drell-Yan processes ($qq \to X \to \ell\ell$ where X is a photon or Z boson including their interferences) in association with jets where one of the leptons fails the identification criteria. A minor contribution is coming from single top quark production in association with jets.

6.3.1 Datasets

The datasets for recorded events are categorized with respect to L1 triggers. The appropriate dataset for this selection is the single-muon dataset. Each event was accepted by one or several HLTs of this category. For later steps, events are only selected if they passed one of two specific HLTs for isolated muons with $p_T > 24\,\text{GeV}$. The use of a HLT selection on data and simulation ensures that no events from other datasets are represented in the simulated samples. In this case it ensures that no event which is not in the single-muon dataset would pass the selection. For the data recorded in 2016, events are chosen under requirements on the detector state at the corresponding time; for example, events were sorted out, when they were recorded while not all detector elements were active. The total recorded integrated luminosity is $\mathcal{L} = 35.14\,\text{fb}^{-1}$, it was computed using the BRILCALC [83] tool.

The simulated samples are categorized by the hard scattering process. The $t\bar{t}$ process used in this thesis was generated by POWHEG including NLO processes in QCD. For the W boson process with additional jets, an (inclusive) sample with NLO in QCD exists, this sample was modeled with aMC@NLO, the events have negative lhe weights (as explained in section 3.1.2) and the amount of events is low. Alternative simulated (exclusive) samples exist in LO with more events. These are grouped in the number of additional jets in the final state of the hard process. The samples for a W boson with two, three and four additional jets are used in this thesis. The Drell-Yan processes were generated with aMC@NLO and grouped with respect to the invariant mass M_{inv} of the virtual photon or Z boson. Two samples, one for $10\,\text{GeV} < M_{\text{inv}} < 50\,\text{GeV}$ and one for $M_{\text{inv}} > 50\text{GeV}$ are used. The single top process is grouped with respect to their production mode and generated with POWHEG. The four main contributing production channels were taken, these are single top quark production in the t-channel (tq), in association with a W boson (tW) and the corresponding processes for top antiquarks ($\bar{t}$q and W$\bar{t}$). The parton shower and hadronization of all processes were modeled by PYTHIA 8, while the detector simulation is performed with GEANT4. The number of events and the cross section of each process are listed in table 6.2.

6.3.2 Data-to-Simulation Comparison

To compare the data with the simulation, the events from the simulation of a certain process i have to be scaled. This is done by assigning a weight for each event depending on the process. The weight is determined by the corresponding cross section σ_i and the efficiency. The efficiency is the number of selected events divided by the number of generated events $N_{i,\text{gen}}$. The events have also to be scaled with the luminosity of the corresponding data. Each event of the simulation gets an event weight of

$$w_i = \mathcal{L}\frac{\sigma_i}{N_{i,\text{gen}}} \quad . \tag{6.1}$$

Table 6.2: Generated processes and their cross sections. Given are the number of generated events N_{gen} and the cross section σ for each simulated process. The inclusive W + jets sample is generated with NLO in QCD but less events are available. The exclusive W + n jets samples are modeled with LO in QCD and more events are available.

Dataset	N_{gen}	σ in pb
$t\bar{t}$	154 948 894	831.76 [82]
DY($10\,\text{GeV} < M_{\text{inv}} < 50\,\text{GeV}$)	47 946 519	6025.2 [84]
DY($M_{\text{inv}} > 50\,\text{GeV}$)	81 781 052	22 635.1 [84]
W + jets (inclusive)	16 497 031	61 526.7 [84]
W + 2 jets (exclusive)	29 878 415	3161 [85]
W + 3 jets (exclusive)	19 798 117	948.2 [85]
W + 4 jets (exclusive)	9 170 576	494.6 [85]
tq	67 240 808	136.02 [86, 87]
$\bar{t}q$	38 811 017	80.95 [86, 87]
Wt	6 952 830	35.6 [86, 87]
$W\bar{t}$	6 933 094	35.6 [86, 87]
WW	994 012	118.7 [88]
WZ	1 000 000	44.9 [88]
ZZ	998 034	15.4 [88]

The efficiency results from summing up all weights. Further improvements can be achieved by applying reweighting for each event depending on known scale factors, as described in section 3.3. In this selection, scale factors were applied for the tight muon ID (sf^{ID}), for the muon isolation (sf^{Iso}), for the muon efficiency in the tracker (sf^{Track}) and for the HLT selection (sf^{HLT}). These depend on the muon p_{T} and η value. Furthermore, a pileup reweighting w_{PU} was performed, depending on the number of true interactions of each event. Additionally, one has to take negative event weights w_{lhe} into account. These appear in the simulation of the Drell-Yan process. For this simulated sample, one has to compute an effective number of generated events $N^{\text{eff}}_{i,\text{gen}}$, which is the number of events with positive weights subtracted by the number of events with negative weights. The scale factors are different for the various periods of the total data-taking in 2016. Therefore, they have to be weighted with the corresponding luminosity of each period. The event weight for each event is then given by

$$\begin{aligned} w_i(n_{\text{PV}}, p_{\text{T}}, \eta, w_{\text{lhe}}) = \frac{\sigma_i}{N^{\text{eff}}_{i,\text{gen}}} \cdot w^{\text{PU}}(n_{\text{PV}}) \cdot w_{\text{lhe}} \\ \cdot \sum_{r \in \{\text{Periods}\}} \mathcal{L}_r \cdot \text{sf}^{\text{ID}}_r(p_{\text{T}}, \eta) \cdot \text{sf}^{\text{Iso}}_r(p_{\text{T}}, \eta) \cdot \text{sf}^{\text{Track}}_r(\eta) \cdot \text{sf}^{\text{HLT}}_r(p_{\text{T}}, \eta) \end{aligned} \tag{6.2}$$

The data-to-simulation agreement can be investigated by looking at distributions of different variables. Four meaningful variables are shown in figure 6.3. The fact that the integral of events in simulation is lower than the integral in data is a hint for an additional process which was not included here. This could be QCD-multijet processes. However, QCD-multijet events are more complicated to model. Since the shape is in good agreement, and this is the more important fact, the selection is sufficient for the domain adaptation studies.

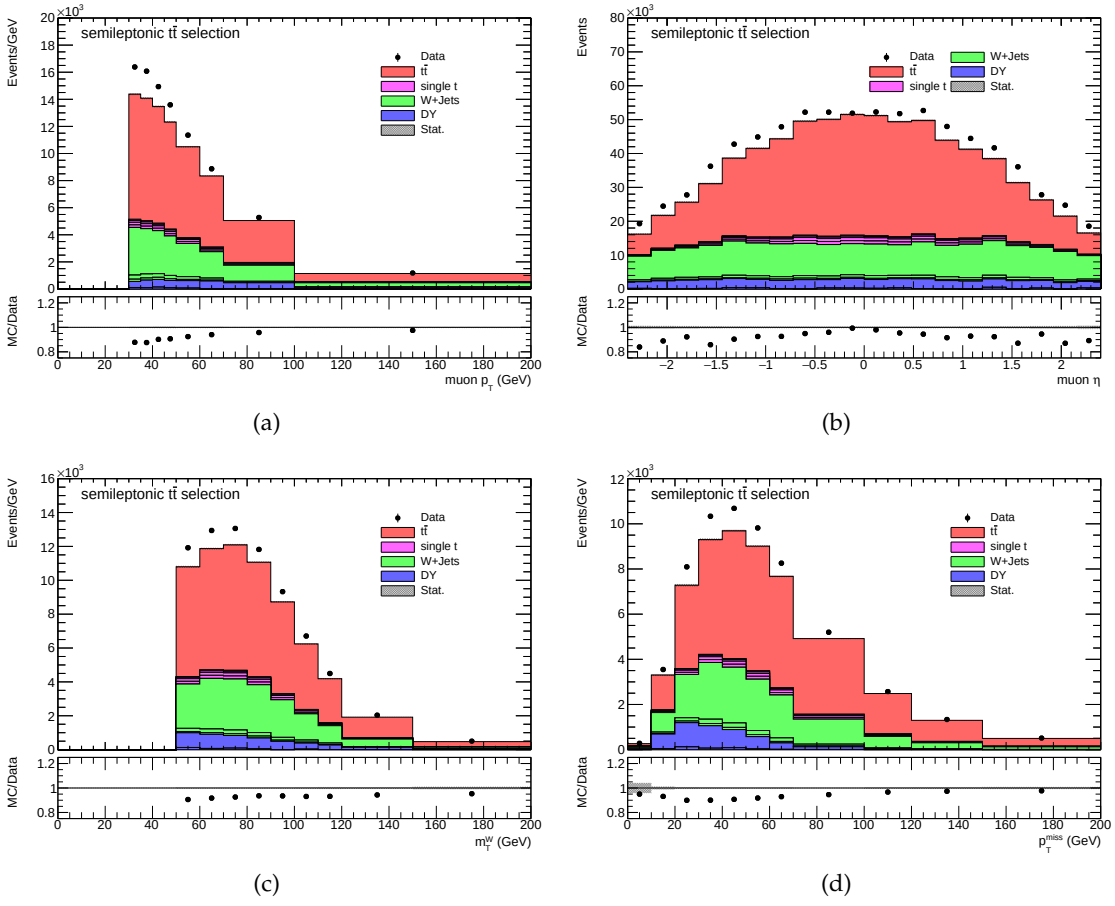


Figure 6.3: Distributions of variables from the semileptonic $t\bar{t}$ selection. Shown are the distributions of muon p_{T} (a), muon η (b), reconstructed transverse mass of the W boson $m_{\mathrm{T}}^{\mathrm{W}}$ (c) and the missing transverse momentum $p_{\mathrm{T}}^{\mathrm{miss}}$ (d). The total amount of events in data is about 7 % higher compared to the simulation. The shape in each distribution is in good agreement. The statistical uncertainties are negligible, systematic uncertainties have not been computed.

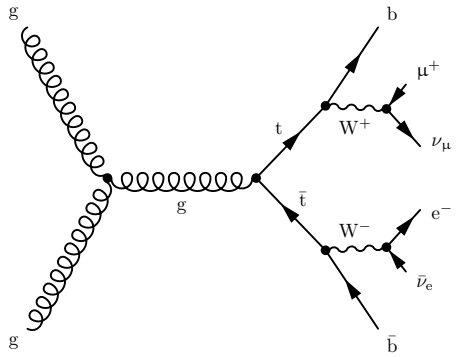


Figure 6.4: Dileptonic decay of a top quark pair. With a selection of the muon and electron, a high fraction of $t\bar{t}$ events can be achieved, and therefore many b jets are obtained. Further cuts on the amount of jets or the missing p_T^{miss} are not necessary for this thesis. Additional jets through initial- or final-state radiation, pileup or through the parton shower are possible.

6.4 Top Quark Pair - Dileptonic Selection

The dileptonic decay mode can be further distinguished with respect to the leptons in the final state. The selected events with two muons or two electrons in the final state are dominated by the Drell-Yan process which has mainly additional light-flavor quark and gluon jets. The decay mode with a muon and an electron with opposite charge, shown in figure 6.4 has the clearest signal for $t\bar{t}$ and is therefore preferred. The selection criteria are

- one good muon with $p_T > 20$,
- one good electron with opposite charge,
- $M_{inv} > 20\,GeV$ of the electron-muon pair to reject backgrounds,
- the good electron or the good muon with $p_T > 25\,GeV$,
- a trigger selection, which is explained in the following.

The main contribution from other processes is the Drell-Yan process where, for example a $\tau^+\tau^-$ pair is produced and one *tau* decays into an electron while the other decays into a muon. Minor contributions arise from the production of vector boson pairs (WW, WZ and ZZ), W bosons in association with jets and single top quarks. Due to the demand of one good electron, the dileptonic $t\bar{t}$ selection is orthogonal to the semileptonic $t\bar{t}$ selection which has a veto against a good electron.

6.4.1 Datasets

Selected data events are taken from the muon-electron/photon dataset. Two HLTs were used in the selection where both require one isolated muon and one isolated electron. The first (second) has a limit for the muon of $p_T > 23\,GeV$ ($p_T > 8\,GeV$) and for the electron of $p_T > 12\,GeV$ ($p_T > 23\,GeV$). (Unfortunately there was a problem during the

data-taking in 2016; two different pairs of these HLTs were used.) The total recorded luminosity for this dataset was again computed using the BRILCALC tool to a value of $\mathcal{L} = 35.20\,\text{fb}^{-1}$.

For the simulation, the same simulated samples were used for the $\mathrm{t\bar{t}}$, Drell-Yan, Wt, and $\mathrm{W\bar{t}}$ processes as for the semileptonic $\mathrm{t\bar{t}}$ selection. In contrast to this, for the W boson process with additional jets, the inclusive W + jets sample was used here. Single top quark production in the *t*-channel has a negligible contribution and is therefore not included. Instead, processes with two bosons were included. These were simulated in LO in QCD using PYTHIA 8 only, the detector simulation was done with GEANT4. The datasets are given in table 6.2.

6.4.2 Data-to-Simulation Comparison

The comparison was done in the same way than for the semileptonic $\mathrm{t\bar{t}}$ selection in the previous section. The fact that two different sets of trigger were used during the periods of the 2016 data has to be taken into account for the simulation. The event weights are therefore additionally multiplied with a flag $f \in \{0, 1\}$ of the HLT of the corresponding period. Moreover, a trigger scale factor for the isolated electron $\mathrm{sf}^{\mathrm{ID+ISO}}$ has to be included. The event weight is then given by

$$\begin{aligned} w_i(n_{\mathrm{PV}},\, p_{\mathrm{T},\mu},\, \eta_\mu,\, p_{\mathrm{T,el}},\, \eta_{\mathrm{el}}, \eta_{\mathrm{SC,el}},\, w_{\mathrm{lhe}},\, f_r) = \frac{\sigma_i}{N^{\mathrm{eff}}_{i,\mathrm{gen}}} \cdot w^{\mathrm{PU}}(n_{\mathrm{PV}}) \cdot w_{\mathrm{lhe}} \\ \cdot \sum_{r \in \{\mathrm{Periods}\}} \Big(f_r \cdot \mathcal{L}_r \cdot \mathrm{sf}_r^{\mathrm{ID}}(p_{\mathrm{T},\mu}, \eta_\mu) \cdot \mathrm{sf}_r^{\mathrm{Iso}}(p_{\mathrm{T},\mu}, \eta_\mu) \cdot \mathrm{sf}_r^{\mathrm{Track}}(\eta_\mu) \\ \cdot \mathrm{sf}_r^{\mathrm{ID+ISO}}(p_{\mathrm{T,el}}, \eta_{\mathrm{SC,el}}) \cdot \mathrm{sf}_r^{\mathrm{HLT}}(\eta_{\mathrm{el}}, \eta_\mu) \Big) \end{aligned} \tag{6.3}$$

The selection has no requirement of the number of jets. The number of jets together with distributions for the leading (trailing) lepton, which is the good muon or good electron with the highest (lowest) p_{T}, are shown in figure 6.5. The integral of the events in simulation is about 7% higher in data compared to the integral in data. The reason could be that not all scale factors are included. A presumption is that something is not correct with the processing of the scale factors for the different HLTs. However, as the shape between data and simulation is in good agreement, the selection is sufficient for the domain adaptation studies.

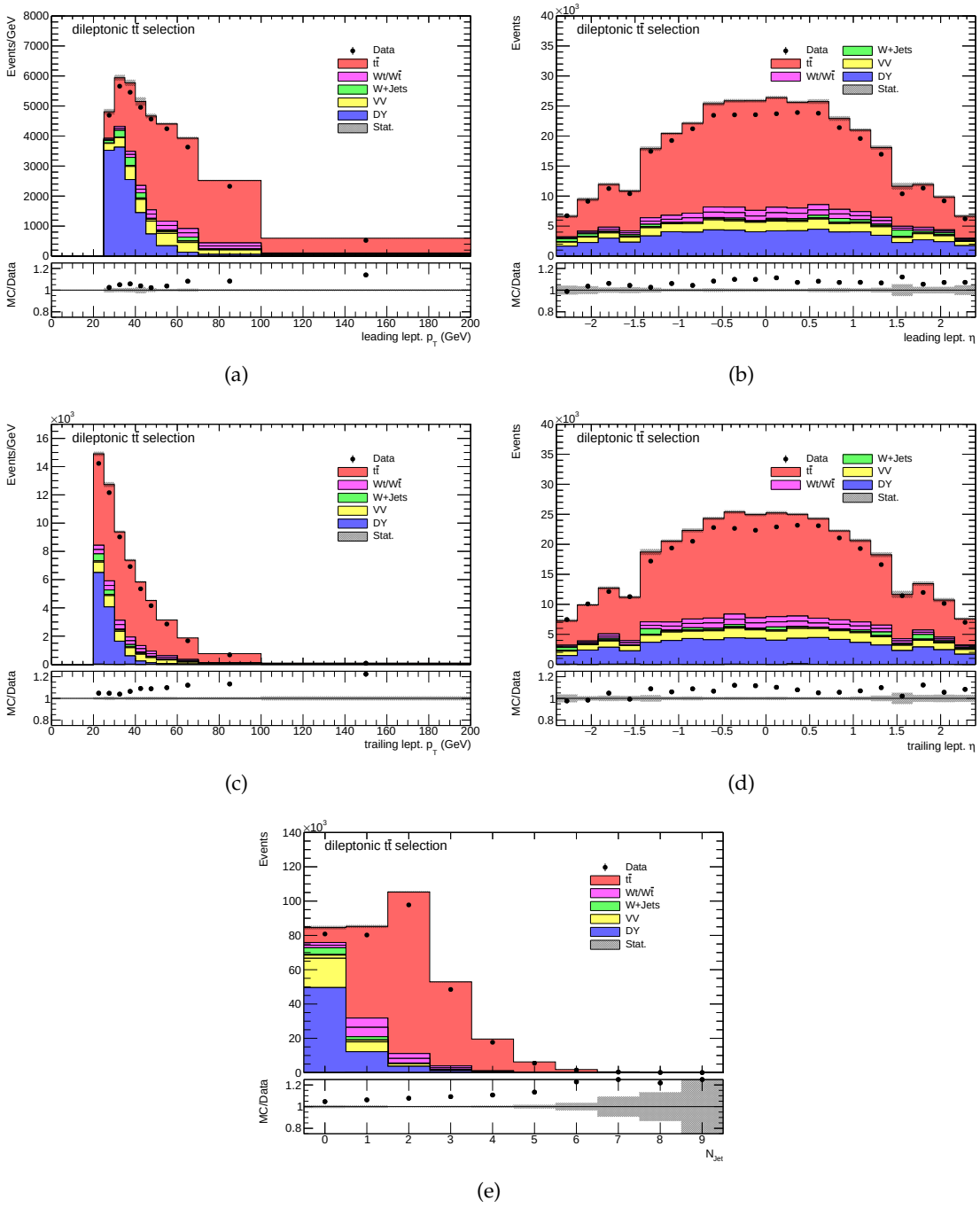


Figure 6.5: Distributions of variables from the dileptonic $t\bar{t}$ selection. Shown are the distributions of p_T (a) and η (b) of the leading lepton, and the p_T (c) and η (d) of the trailing lepton and the number of jets (e) in each event. The integral of simulated events is about 7 % higher than the integral of events in data. The reason for this behavior is assumed in wrong scale factors. However, the shape is again in good agreement. The statistical uncertainties are higher but still negligible, the systematic uncertainties are not given.

7 Domain Adaptation Studies

Domain adaptation methods are well examined in empirical studies on standard machine learning tasks [64, 67, 69]. The results are different for the various methods and depend on the specific task. In many classification tasks, it is possible to achieve an improvement of the classification accuracy on the target domain, compared to a model trained on the source domain only. The degree of improvement varies and depends on the domain discrepancy and the complexity of source and target domain. In some cases, where the target domain is more complex than the source domain, no improvement with respect to a model trained on the source domain only, was achieved [67]. The outcome of domain adaptation methods applied to a certain problem can often hardly be estimated.

In addition, the task in this thesis is different from the standard machine learning tasks that can be found in the literature. The primary goal is the improvement of the data-to-simulation agreement, while the jet identification efficiencies should be as good as possible. The DNN models used in this thesis are rather complicated compared to the ones used in the references [64, 67, 69], while the discrepancies between the source domain (simulation) and target domain (data) are less obvious and only subtle. Additionally, for the target domain, the exact information about the class composition is not available, as each element is unlabeled. The effect of domain adaptation is therefore difficult to assess. An empirical study of this problem is strictly necessary to see if an application of domain adaptation is worthwhile.

Two domain adaptation methods, the moments method and the domain-adversarial method were applied to the DeepFlavour algorithm, the studies and the results are described in section 7.1. Studies on the DeepCSV algorithm were made at a later stage and are outlined in section 7.2. Based on the findings, described in the first section, only the domain-adversarial approach was used here.

As a figure of merit for the domain-discrepancy, the Kolmogorov-Smirnov test [89, 90] statistic (KS-value) is chosen. The KS-value measures the difference of two one-dimensional empirical distribution functions; it can further be used to perform the KS-test, whether both empirical distribution functions origin from the same distribution. However, the KS-test depends on the total number of elements in each distribution, in most cases of this study, the probability that both distributions origin from the same

source is almost null and therefore not as descriptive as the KS-value.

7.1 Studies on the DeepFlavour Algorithm

For the DeepFlavour algorithm, an existing fully trained model was taken, the training was performed as described in section 5.4. The goal of this study was to use the DeepFlavour model and adjust the weights to a small degree to achieve the domain adaptation. Therefore, only the free parameters of the dense layers and the batch normalization layers between the dense layers were chosen to be adjustable. The free parameters of all other layers were fixed. The domain adaptation training was performed using samples from the dileptonic $t\bar{t}$ selection as described in section 6.4. Scale factors have not been used, the simulated events were only scaled with respect to equation 6.1 including lhe weights for the respective samples.

7.1.1 Moments Method

The moments method was described in section 4.4.1. It was chosen to take the output features as the input distributions for the moments loss (equation 4.23). Only the first moment (mean), and the first central moment (variance) were taken into account. The inclusion of the categorical cross-entropy ensures that the classification performance of the model is retained. As a measure for the b tagging performance, the integral of the ROC curve, the so called AUC score, was used. The closer the AUC score is to one, the better.

The training was done using the Adam optimizer with a learning rate of 10^{-4}, the learning rate was not reduced during the training. A dropout rate of 0.2 was used and a fraction of 10% of the data was taken for the validation samples. To get a meaningful gradient, a big batch size of 10^5 was taken. The training was performed for 100 epochs. The loss function has two free hyperparameters α and β, which determine how strong differences in the moments are punished. For the optimization of these hyperparameters, a grid search with eight values was performed under the assumption that $\alpha = \beta$. The higher the values, the better the data-to-simulation agreement, but the worse the AUC score. For values higher than 2, there was almost no more gain in the data-to-simulation agreement and the AUC score dropped by several percent. Lower values of β were tested but these showed minimal influence in the result.

The total loss function consists of the categorical cross-entropy and two moment terms: the summed differences in the mean values and in the variance values between data and simulation of all output features. During the training, the values of these terms were stored for each epoch. The history of the total loss function together with the

moment terms is shown for $\alpha = \beta = 2$ in figure 7.1. The total loss function is reduced until the training has ended. The moment terms contribute to a reduction for the first 20 to 25 epochs, later on, the reduction is only driven by the categorical cross-entropy.

To compare the data-to-simulation agreement, the models were evaluated on the samples from the dileptonic $t\bar{t}$ selection. The predictions for two classes of the standard DeepFlavour model are shown in figure 7.2 (a) and (b). For the model trained with the moments method using $\alpha = \beta = 2$, the predictions are shown in figure 7.2 (c) and (d). Compared to the standard DeepFlavour model, the KS-values have been reduced and also the maximal differences decreased in some parts of the distributions. In the case of $\alpha = \beta = 2$, the AUC score is found to be 0.5% lower compared to the training corresponding to $\alpha = \beta = 0$. Apart from this, one can observe a change of the shape of the predictions in figure 7.2 (c) and (d). This fact, and the behavior of the loss function in figure 7.1 are consistent with the conclusion that the model is learning the class composition of the dileptonic $t\bar{t}$ selection. To minimize this effect, it is reasonable to stop the training until the moment terms are no longer significantly reduced.

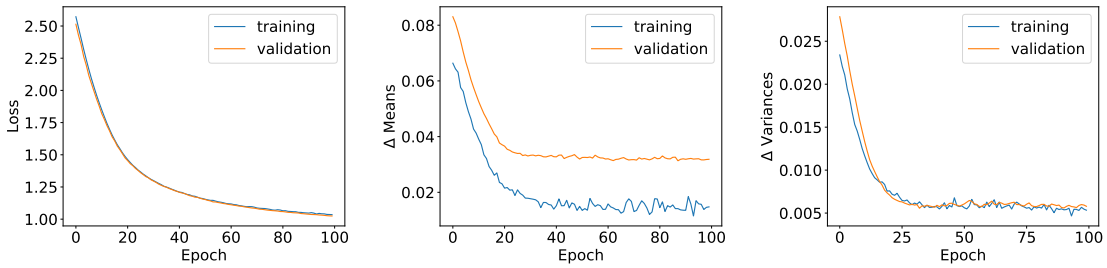


Figure 7.1: Moments method: history of training. The total loss (a) is steadily decreasing while the differences in the mean values (b) and variance values (c) are flatten out at about the 30th epoch. The gap in the differences in the mean values between the training samples and validation samples is due to overfitting.

Further improvements of the data-to-simulation agreement would be possible by the inclusion of higher moments in the loss. Moreover, the input features for the loss function could be changed to an intermediate layer. The disadvantages of this method are the complex implementation, the needed hyper parameter optimization and the unstable training procedure. Another promising approach for the domain adaptation is the domain-adversarial method, this will be described in the following.

Table 7.1: Comparison between domain adaptation methods. The KS-values for all discriminators (Disc.) of the standard DeepFlavour algorithm are compared with the results of the moments method and the domain-adversarial (DA) method. In four of the six cases, the DA method delivers better results.

Disc.	KS-value		
	DeepFlavour	Moments Method	DA Method
P(b)	0.0279	0.0263	**0.0212**
P(bb)	0.0293	0.0252	**0.0100**
P(lept. b)	0.0308	0.0284	**0.0206**
P(c)	0.0453	0.0353	**0.0288**
P(uds)	0.0701	**0.0202**	0.0349
P(g)	0.0402	**0.0224**	0.0351

7.1.2 Domain-Adversarial Method

The domain-adversarial method was explained in section 4.4.2. It was chosen to take the output features of the standard DeepFlavour model as input features for the domain classifier. The domain classifier was chosen to consist of the gradient reversal layer, two hidden layers with 100 neurons each and ReLu activation functions, one dropout layer in between and a single output neuron for domain classification with a sigmoid activation function.

When the domain-adversarial training was applied directly on the pre-trained model, its loss function was not converging. Therefore, the standard DeepFlavour model had to be trained first on the flavor classification only. The Adam optimizer was used with a learning rate of 10^{-4} and a batch size of 10^5. Again, the dropout rate was set to 0.2 and the validation fraction was 10%. After 100 epochs, the domain loss was enabled with a loss weight of 2, the learning rate was reduced to $5 \cdot 10^{-5}$ and the training was continued for further 50 epochs. The trade-off factor λ (equation 4.24) was kept at 1. The result is shown in figure 7.2 (e) and (f).

Compared to the moments method, the domain-adversarial method delivers a similar data-to-simulation agreement with respect to the KS-values and the maximal differences in the residues. The overall result was assessed as slightly better, as in the most cases, the domain-adversarial method has delivered better KS-values (table 7.1). Moreover, the AUC score has not significantly changed compared to the point before the domain loss was enabled. This was achieved without any hyperparameter optimization and the implementation was more straightforward compared to the moments method. Unfortunately, the same behavior as before was observed, the shape of the

output distributions changes as the classifier learns the class composition of the dataset.

This behavior is unwanted, since properties of the DNN, achieved from the training samples used in the first step of the training, are getting lost. These properties are for example that the algorithm does not learn the direct correlation of the jet flavor with the p_T and η value and that a certain prior probability for the different jet flavors can be chosen freely. The first one was achieved by reweighting the jets in the way that the p_T and η distributions for each jet flavor were the same, the latter one is achieved by using class weights. Both is not possible when using the samples for the classification and domain adaptation training at the same time, since the flavor of jets from data is unknown and reweighting of the simulated samples would destroy the data-to-simulation agreement in the class composition.

Apart from this, the risk of overfitting is big as only 8 million labeled jets exist in the dileptonic $t\bar{t}$ selection. Another approach is to train the model from scratch using additionally a big sample of labeled jets. As the DeepFlavour algorithm takes several days to train from scratch, it was chosen to do this study on the DeepCSV algorithm. This will be explained in the following section.

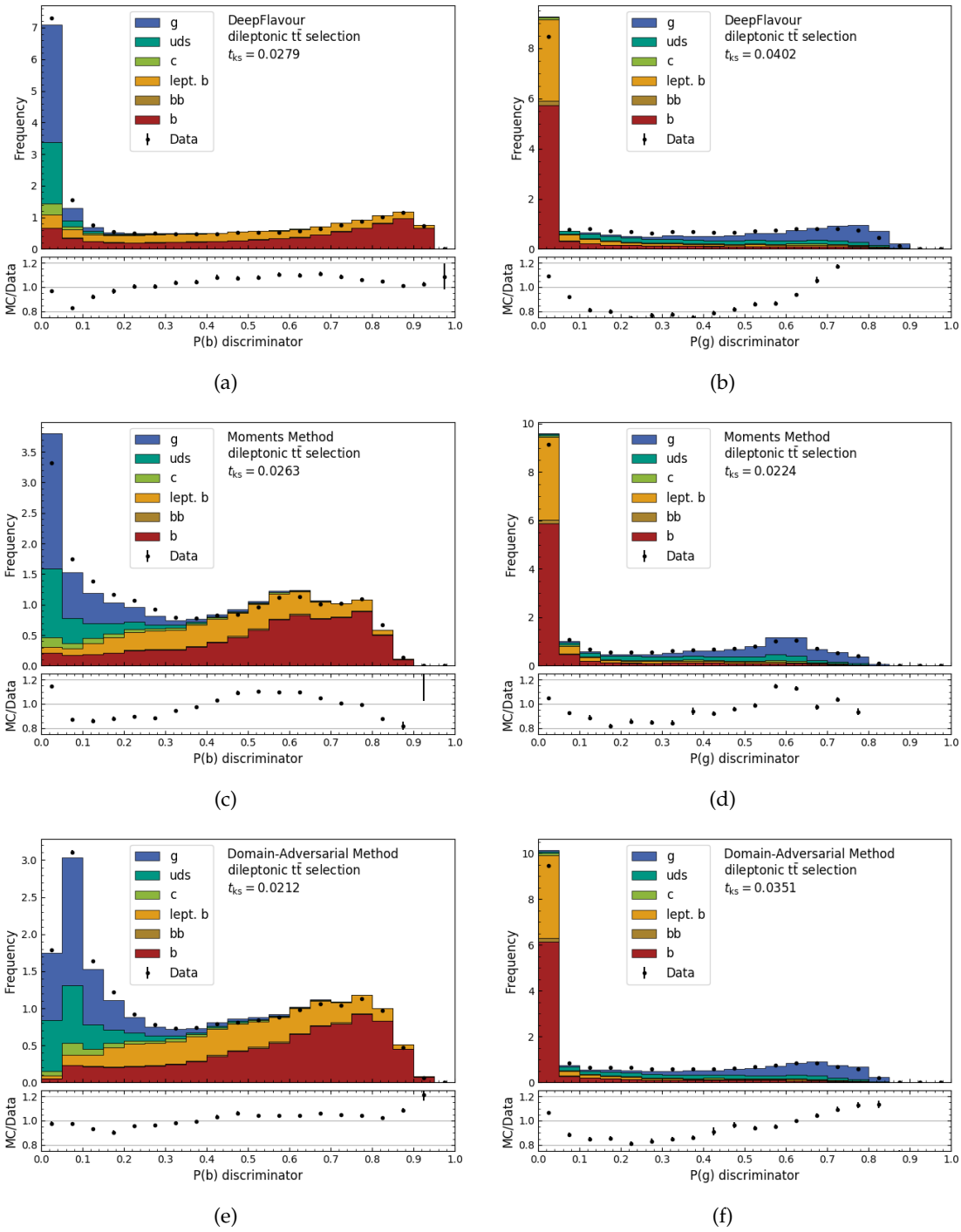


Figure 7.2: Output distributions of the DeepFlavour algorithm. Shown are exemplary two of the six output features for the dileptonic $t\bar{t}$ selection. In the first row, the P(b) and P(g) discriminators are shown in (a) and (b) for the standard DeepFlavour model. In the second row, the same discriminators are shown in (c) and (d) after applying the moments method. In the third row, the results for the domain-adversarial method are given. For each output feature, the KS-value is given as t_{ks}.

7.2 Studies on the DeepCSV Algorithm

The studies on the DeepFlavour algorithm showed that the domain-adversarial method was superior to the moments method in terms of the results and the simplicity of the implementation and hyperparameter optimization. Therefore, only the domain-adversarial method was used for the studies on the DeepCSV algorithm. In the following, the DeepCSV algorithm with the extension of the domain-adversarial method is denoted as domain-adversarial DeepCSV (DA-DeepCSV) algorithm.

In these studies, it was tried to reduce the domain discrepancy using samples from the semileptonic $t\bar{t}$ selection including all scale factors as described in equation 6.2. Additionally, the orthogonal dileptonic $t\bar{t}$ selection with the scale factors as mentioned in equation 6.3 was used to evaluate the DA-DeepCSV algorithm. In contrast to the samples used in the last section, all jets were included, also the ones without a label. These originate for example from the underlying event, pileup effects or tau leptons. As they are present in the recorded data as well, they also have to be included in the simulated samples. This is possible as the labels of these samples are not used in these studies.

The domain-adversarial method was implemented as outlined in section 4.4.2. The first three hidden layers of the DeepCSV model were chosen as the feature extractor. The output features of the third hidden layer were taken as input for the domain classifier. The domain classifier was constructed in the same way as described in section 7.1.2. The label predictor consists of the remaining part of the standard DeepCSV model, two hidden layers with a dropout layer in between and the standard output layer.

7.2.1 Training

To train the DeepCSV algorithm from scratch, a high amount of labeled jets is needed. As the amount of jets from the event selection, as described in chapter 6, is not sufficient, the training of the flavor classification and the training of the domain-adversarial was done on two different sets of samples. The flavor classification was done on simulated jets from $t\bar{t}$ and QCD multijet events, in the following denoted as flavor classification samples. In total about 112 million jets were taken for training and 28 million for validation. The samples from the semileptonic $t\bar{t}$ selection were used for the domain-adversarial training, since they have a higher amount of jets and a more balanced class composition than the samples from the dileptonic $t\bar{t}$ selection. The samples from the semileptonic $t\bar{t}$ selection are denoted in the following as domain-adversarial samples and consist of more than 11 million simulated jets and 3 million jets from recorded data.

Training of the Standard DeepCSV Algorithm

To see the effect of the domain-adversarial method, a training of the standard DeepCSV algorithm was performed first only on the flavor classification samples. The Adam optimizer was used with a start learning rate of $5 \cdot 10^{-4}$ and the reduce-on-plateau method was applied: the loss was reduced by the factor of two whenever the validation loss did not decrease for two epochs and after each reduction, a cooldown of five epochs was set in which no further reduction was made. A batch size of $5 \cdot 10^4$ was taken for 80 epochs and early stopping was applied. The training took about three hours, using a NVidia GeForce GTX Titan X [91] graphics card. The history of the loss value is shown in figure 7.3 (a).

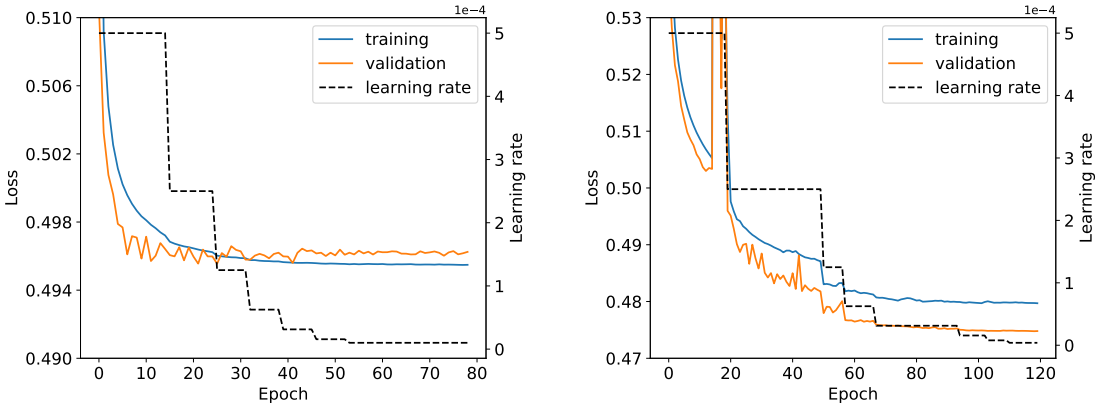
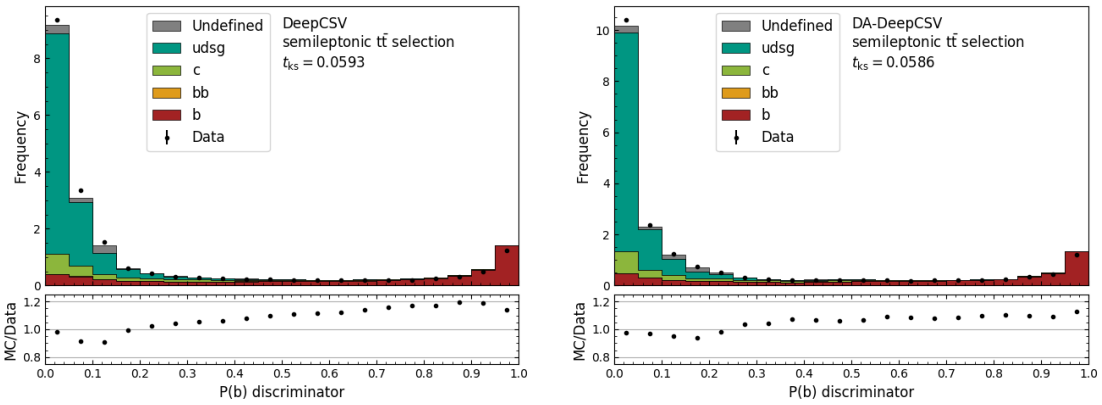


Figure 7.3: Training process for the DeepCSV and DA-DeepCSV algorithm. The left plot corresponds to the DeepCSV algorithm, which fully converged after about 40 epochs. The DA-DeepCSV algorithm needs almost 120 epochs. The different minimal values of the loss function reasons from the different batch sizes. The learning rates are shown in dashed black lines, the values are given on the right axis.

Based on the lowest validation loss, the best model was chosen. The data-to-simulation agreement and the performance is shown in this section together with the results of the DA-DeepCSV algorithm.

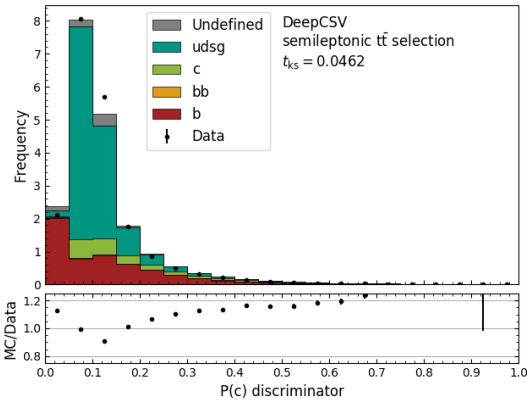
Domain Adversarial Training

The DA-DeepCSV algorithm was trained alternately on two sets of samples. On the flavor classification samples only the flavor loss was enabled and optimized while for the domain-adversarial samples the flavor loss was disabled and the domain loss was

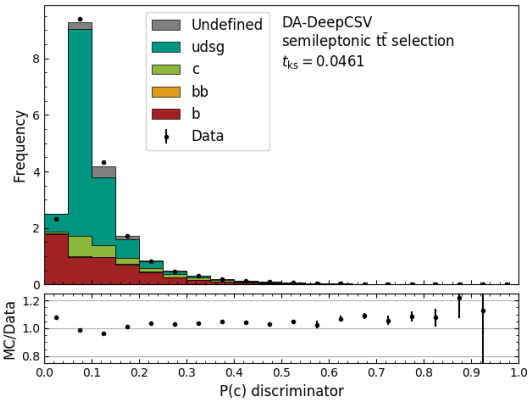


(a)

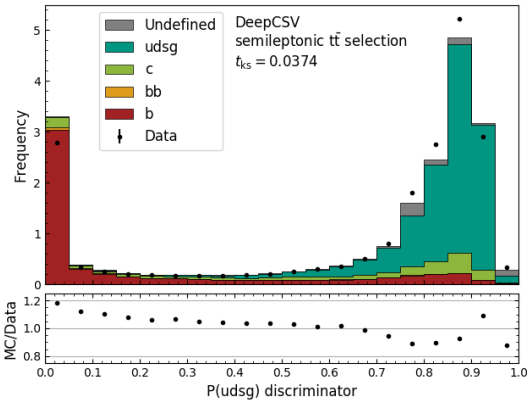
(b)



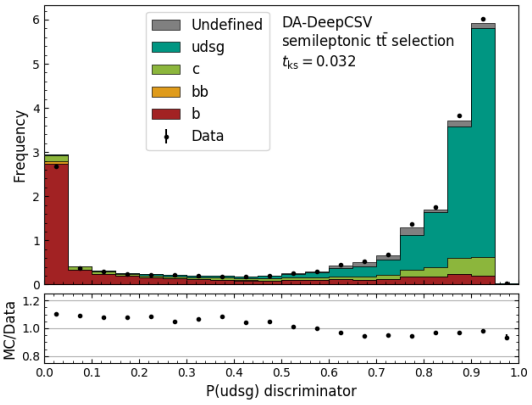
(c)



(d)



(e)



(f)

Figure 7.4: Predictions for samples from the semileptonic $t\bar{t}$ selection. Shown are the results of the standard DeepCSV algorithm (left) and of the DA-DeepCSV algorithm (right). Three of the four predictions are given: the b jet prediction in (a) and (b), the c jet prediction in (c) and (d) and the prediction for light quark and gluon jets (udsg) in (e) and (f). The data-to-simulation agreement has been improved while the shape has not changed much. The biggest change in shape can be observed in the P(udsg) discriminator. The P(bb) discriminator is not shown as jets of this class are not represented in the samples.

enabled and optimized. The training was started with 15 epochs on the flavor classification samples. In the subsequent epochs, the flavor classification sample was trained for five batches followed by the training of one epoch on the domain-adversarial samples. This was performed using KERAS Callbacks [75]. For the optimization, the Adam optimizer was used with a start learning rate of $5 \cdot 10^{-4}$ and the reduce-on-plateau method as described before. A batch size of $5 \cdot 10^5$ ($5 \cdot 10^4$) was used for the flavor classification (domain-adversarial) samples. The training was performed for 120 epochs of the flavor classification samples. A grid search was performed for the optimization of the trade-off factor λ. Five values between $\lambda = 0.5$ and $\lambda = 1.2$ were taken, the best result was achieved for $\lambda = 1$. For values of $\lambda < 1$, the data-to-simulation agreement was worse, while for $\lambda > 1$ the data-to-simulation agreement was better for the training samples, but worse for the independent samples described in section 7.2.2.

For $\lambda = 1$, the loss value and the learning rate are together shown in dependence of the epoch in figure 7.3 (b). The loss value jumps after the 15th epoch as the domain-adversarial training was started. The optimization is much harder and less stable, this becomes clear when comparing the fluctuations of the loss functions of figure 7.3 (a) and (b).

The training time was about 50 hours, which is almost 17 times longer than the training of the standard DeepCSV algorithm. To speed up the training, it was tried to train only batches of the domain-adversarial samples instead of training one whole epoch and using additionally a loss weight > 1 for the domain loss. But either there was hardly any improvement of the data-to-simulation agreement or the loss did not converge. Apart from this, the used training procedure took a high amount of resources as all samples are stored on the memory during the training procedure. Training the DeepFlavour algorithm in this way would exceed the limit of the memory as it takes about ten times more input features as the DeepCSV algorithm. However, it is quite possible that there is a way to speed up the training many times over and save space in the memory at the same time.

A remedy could be provided by using a generator function in the training procedure instead of KERAS Callbacks. The generator function can be used to alternately load batches from the flavor classification samples or domain classification samples on the memory, process them and remove them after processing. In this way, less memory is needed. Also a decreased training time can be expected as the training process is not interrupted through callbacks.

Predictions of Training Samples

The model with the lowest validation loss was taken and used to generate the predictions for the domain-adversarial samples. The results of the output distributions are

shown in figure 7.4. As the domain-adversarial samples contain almost no jets with bb label, the predicted values for this class are all around zero, therefore the P(bb) discriminator is not shown.

With the use of the domain-adversarial method, the maximal residual differences in all output features have been reduced. Furthermore, the shape has only changed to a small degree compared to the studies in section 7.1. The KS-values are only slightly lower with the use of the domain-adversarial method. In the residues it can be noticed that in the P(b) discriminator (figure 7.4 (b)) the data is lower for values of P(b) < 0.2, while the data is almost constantly higher for values of P(b) > 0.3. The inverse effect can be noticed for the P(udsg) discriminator (figure 7.4 (f)). This is consistent with the assumption that QCD multijet processes are represented in the measured data of the semileptonic $t\bar{t}$ selection, but missing in the simulated samples of this selection.

The inclusion of jets from QCD multijet events in the simulated samples of the semileptonic $t\bar{t}$ selection could improve the data-to-simulation agreement in the class composition. This would reduce the domain discrepancy and reveal the mismodeling of the jets. During training, the domain discrepancy would be less reduced by enforcing the same class composition in the predictions, which destroys the tagging efficiency, but more by reducing the differences coming from the mismodeling of the jets. Therefore, better results can be expected.

7.2.2 Evaluation on Independent Samples

The data-to-simulation agreement was improved for the samples which were used for the domain-adversarial training. To check if the model has not learned to reduce the sample specific discrepancy, but a more general jet specific discrepancy, the evaluation of the model on independent samples is needed. Therefore, predictions on the samples from the dileptonic $t\bar{t}$ selection were performed. The results are shown in figure 7.5.

The data-to-simulation agreement with the standard DeepCSV algorithm for the dileptonic $t\bar{t}$ selection is already better than for the semileptonic $t\bar{t}$ selection. With the domain-adversarial method, a further improvement was achieved in all output features. This comes particularly clear when looking at the KS-values: an improvement of the factor of about 4 was achieved in the P(c) and P(udsg) discriminator distribution. In general, one would expect less improvements of the data-to-simulation agreement on the samples of the dileptonic $t\bar{t}$ selection compared to the samples of the semileptonic $t\bar{t}$ selection, which were used in the training. The fact that the results on this independent and orthogonal samples are better, is probably caused by a better data-to-simulation agreement in the class composition in this samples. It shows that the DA-DeepCSV algorithm learned to reduce the influence of the mismodeling in the jets

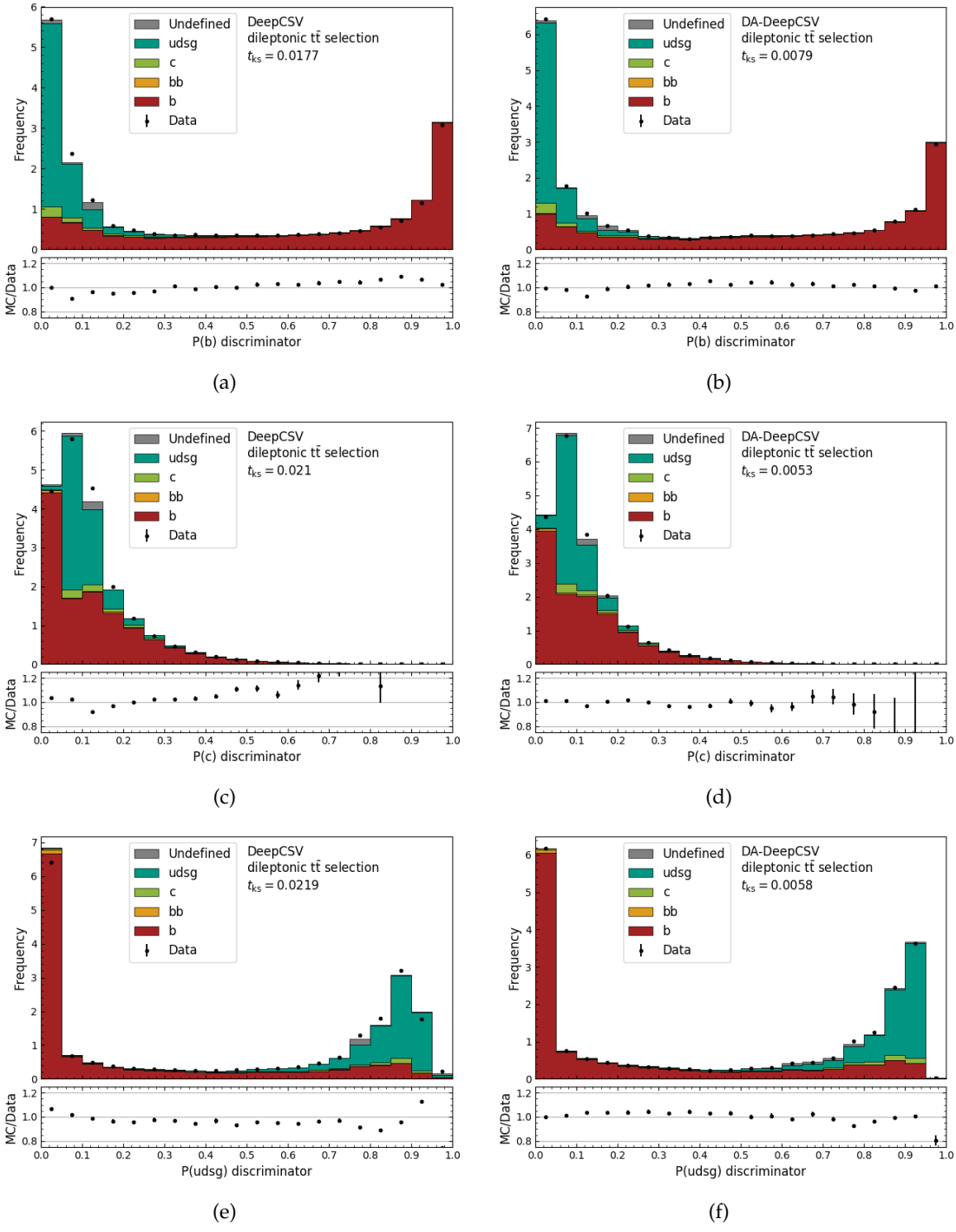


Figure 7.5: Predictions for samples from the dileptonic $t\bar{t}$ selection. Shown are the results of the standard DeepCSV algorithm (left) and of the DA-DeepCSV algorithm (right). Again distributions are shown for the b jet prediction in (a) and (b), for c jet prediction in (c) and (d) and for light quark and gluon jet prediction in (e) and (f).

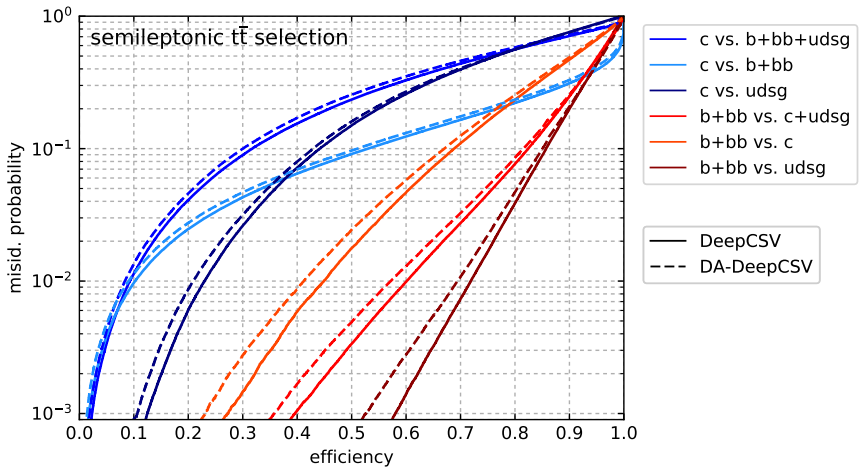


Figure 7.6: ROC curves with and without domain-adversarial extension of the DeepCSV. The tagging efficiency for several categories are shown in different colors. The ROC curves for the standard DeepCSV algorithm are shown as solid lines while the ones for the DA-DeepCSV algorithm are drawn as dashed lines.

even though the discrepancy due to the class composition was bigger in the domain-adversarial training samples. This is an interesting insight as it proves the robustness of the domain-adversarial method.

However, the samples from the dileptonic $t\bar{t}$ selection are similar to the ones from the semileptonic $t\bar{t}$ selection, as those were simulated with almost the same software. It would be interesting to see, how the data-to-simulation agreement looks like in a total different scope, for example for samples from a selection of QCD multijet events. This study requires a more sophisticated event selection and is left for future work.

It is furthermore important to see how the performance has changed. As the domain-adversarial samples were not used for the training of the flavor classification, a ROC curve was generated on them. The ROC curve is shown in figure 7.6. The domain-adversarial method leads to a loss of the tagging efficiency on simulated jets of up to 2% for the loose WPs, up to 4% for the medium WPs and up to 6% for the tight WPs. This was to be expected as the domain-adversarial method uses a complex training procedure and also prevents the use of simulation specific properties that could be useful for a separation. The more important question is how the performance on data has changed.

7.2.3 Two-Tag Counting Method

To measure the b tagging efficiency in data, more sophisticated methods have to be used. A widely used method in the CMS collaboration for crosschecks on b tagging

efficiencies is the so called two-tag counting method [73]. This method works on events from the dileptonic $t\bar{t}$ selection. The events were selected as described in section 6.4 with an additional requirement of exactly two jets. This provides a high fraction of events where both jets are b jets. For the events from simulation, jets labeled as b and bb are taken together, in the following, the label is just noted as b. The b tagging efficiency ϵ_b is given by

$$N_{2\text{ b-tagged}} - N_{2\text{ b-tagged}}^{\text{non-b jet}} = \epsilon_b^2 n_{2\text{ b jets}} \quad , \tag{7.1}$$

where $N_{2\text{ b-tagged}}$ is the number of events where both jets are identified as b jets, $N_{2\text{ b-tagged}}^{\text{non-b jet}}$ the number of events where both jets are identified as b jets but either one of them or both jets are not truly b jets and $n_{2\text{ b jets}}$ the number of events where both jets are truly b jets. As the true information of a jet is not known in data, the two latter numbers are taken from simulation. The number of events, categorized by the two jets, are shown in figure 7.7 (a). Instead of taking the total numbers of each category, it is reasonable to take the respective fractions to reduce the dependency of the cross sections of the simulated processes. The equation can be rearranged to determine the efficiency as

$$\epsilon_b = \sqrt{\frac{F_{2\text{ b-tagged}} - F_{2\text{ b-tagged}}^{\text{non-b jet}}}{f_{2\text{ b jets}}}} \quad , \tag{7.2}$$

with the fraction of events from data $F_{2\text{ b-tagged}}$ where both jets are identified as b jets, the fraction of events from simulation $F_{2\text{ b-tagged}}^{\text{non-b jet}}$ where both jets are identified as b jets, but not both jets are labeled as b, and the fraction of events in simulation $f_{2\text{ b jets}}$ where both jets are labeled with b.

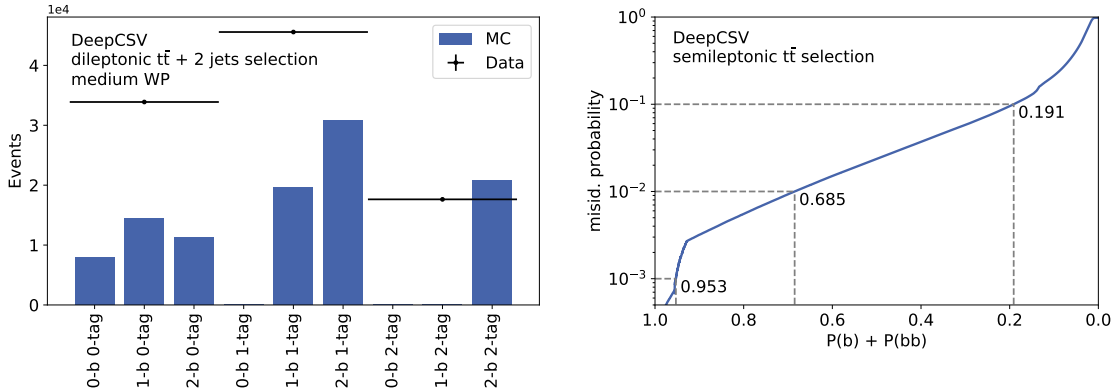
The efficiency in simulation can be calculated as

$$N_{2\text{ b-tagged}}^{2\text{ b jets}} = \epsilon_b^2 n_{2\text{ b jets}} \quad \Rightarrow \quad \epsilon_b = \sqrt{\frac{F_{2\text{ b-tagged}}^{2\text{ b jets}}}{f_{2\text{ b jets}}}} \quad , \tag{7.3}$$

where $N_{2\text{ b-tagged}}^{2\text{ b jets}}$ ($F_{2\text{ b-tagged}}^{2\text{ b jets}}$) is the number (fraction) of events from simulation, which have two b tagged jets and both jets are labeled with b.

For the b tagging criteria, working points are chosen on fixed FPRs. The working points were determined on the simulated samples of the semileptonic $t\bar{t}$ selection, the results are shown for the standard DeepCSV model in figure 7.7 (b). Again, the categories of b and bb were taken together.

The results of the two-tag count method are listed in table 7.2. The b tagging efficiency is lower for the DA-DeepCSV algorithm compared to the standard DeepCSV algorithm at all WPs for simulation as it was already the case in the ROC curve in figure 7.6.



(a) (b)

Figure 7.7: Two-tag counts and working points. In the left plot, events from data are categorized according to the number of identified b jets (n-tag). Events from simulation are additionally categorized according to the number of b labels. The right plot shows the loose, medium and tight working points at 10%, 1% and 0.1% FPR respectively. The jets of the events in the left plot were tagged based on the medium working point.

However, important is only the b tagging efficiency in data, which has dropped by only 0.5% at the loose WP, about 2% at the medium WP and 1% at the tight WP. As a result, the scale factors of the DA-DeepCSV algorithm have values of almost one.

This confirms the before mentioned assumption that with the use of the domain-adversarial method, simulation specific properties are excluded from the classification. It is assumed that the remaining loss in the b tagging efficiency in data is due to a difference in the class composition between data and simulation. With the same class composition, one would probably get no loss in the efficiency for data, while the efficiency in simulation would be reduced to match the efficiency in data. The domain adversarial method acts more like a regularization method between the source (simulation) and target (data) domains, which explains why no gain in the efficiency for the target (data) domain can be achieved.

However, with some small changes of the algorithm, it could also be possible to improve the efficiency in data. A proposal for this is to include the domain label as an input feature for the DNN. This would allow the DNN to treat elements from data and simulation differently. In this way, distributions of features in data that are shifted compared to simulation could be corrected easily. On the other hand, this method could lead to a less good regularization.

One should note that the domain-adversarial training in this thesis was performed

Table 7.2: Two-tag efficiencies and scale factors. The b tagging efficiencies for data (ϵ_{Data}) and simulation (ϵ_{MC}) are listed at three working points (WPs): loose (L), medium (M) and tight (T). They were computed using the standard DeepCSV model and the DA-DeepCSV model. The b tagging scale factors sf_b are closer to one for the DA-DeepCSV model at each WP. The statistical uncertainties are given.

Model	WP	ϵ_{Data}	ϵ_{MC}	sf_b
DeepCSV	L	0.795 ± 0.004	0.818 ± 0.003	0.971 ± 0.008
	M	0.549 ± 0.003	0.574 ± 0.003	0.96 ± 0.01
	T	0.320 ± 0.002	0.331 ± 0.002	0.97 ± 0.01
DA-DeepCSV	L	0.790 ± 0.004	0.803 ± 0.003	0.984 ± 0.008
	M	0.531 ± 0.003	0.531 ± 0.003	1.00 ± 0.01
	T	0.310 ± 0.002	0.309 ± 0.002	1.01 ± 0.01

using selected jets. The jet selection is described in section 6.2.3. For jets with looser selection criteria, the data-to-simulation agreement is probably worse. At the present state, when applying the DA-DeepCSV algorithm as b jet discriminator at the tight or the medium WP, scale factors would not be necessary as these are consistent with unity. This would simplify many analyses and save a lot of time. Moreover, the systematic uncertainties of the scale factors are expected to be reduced compared to the previous ones as they are closer to unity. Therefore, the use of the DA-DeepCSV algorithm would also make sense for every analysis that is limited in its significance by the systematic uncertainties. Analyses, which are limited by the statistical uncertainties may prefer the standard DeepCSV algorithm or the DeepFlavour algorithm as these have the best tagging efficiencies.

8 Summary and Outlook

The identification of heavy-flavor jets is crucial for many physics analyses for the investigation of the standard model and for the search of physics beyond the standard model. Deep neural networks (DNNs) are used to perform a multivariate classification to distinguish jets in terms of their original flavor. Simulated events are used to adapt the free parameters of the DNNs in a process called training. Due to mismodeling, discrepancies in the reconstructed physical objects in simulation compared to measured data occur. The discrepancies of the reconstructed objects lead to a difference in the identification efficiency between jets from measured data and simulated samples. When applying the identification tools in analyses, scale factors have to be used to correct the different identification efficiencies. For large deviations of the scale factor values from unity, a greater systematic uncertainty is typically obtained. This reduces the significance of the analyses. Therefore, it is important to achieve good data-to-simulation agreement for the jet identification with resulting scale factors close to 1.

A promising approach to achieve this is the use of domain adaptation techniques. The goal of domain adaptation methods is to reduce the difference between the source domain (simulation) and target domain (data) for the output of the DNNs. In this thesis, two domain adaptation approaches were tested: the moments method and the domain-adversarial approach. They were applied on two jet identification algorithms: the DeepCSV algorithm, which is currently used as the default jet identification algorithm in the CMS collaboration and the DeepFlavour algorithm, which is a further development of the DeepCSV algorithm with higher complexity and better performance.

For the studies with the DeepFlavour algorithm, the domain adaptation methods were applied on a fully trained DeepFlavour model. By training the model with the domain adaptation methods using jets obtained by an event selection, an improvement of the data-to-simulation agreement in all output variables – which are the jet flavor discriminators – was achieved. Compared to the moments method, the domain-adversarial method provided better results, it was more stable and also simpler to apply. Unfortunately, both methods changed the shape of the output distributions of the DeepFlavour model and important properties of the model, achieved in previous training processes, got lost.

For the studies with the DeepCSV algorithm, only the domain-adversarial method was used and the DeepCSV algorithm was trained from scratch. A bigger set of simulated jets were included for the training of the flavor classification of the model to avoid previously described problems. Selected events were only used for the domain adaptation. In this way, the data-to-simulation agreement was improved in the distributions of each output variable of the DeepCSV algorithm, while the shape of the distributions changed only slightly. This was shown for the events directly used in the training process for the domain adaptation, but also for events from an independent and orthogonal selection. In the latter case, the reduction of the Kolmogorov-Smirnov test statistic up to a factor of 4 was achieved. It was furthermore shown that the scale factor values for b jet identification were closer to unity when applying the domain-adversarial method. For two of the three working points, the scale factors were consistent with 1 within their statistical uncertainties of 1%. However, as a downside, a loss in the b jet identification efficiency in data of maximal 2% was observed.

It can be concluded that the domain-adversarial (DA) method works as a regularizer between the source (simulation) and target (data) domain. Therefore, a prediction of the DA-DeepCSV algorithm is assessed to be more trustworthy in the scope of the jets, used for the domain-adversarial training. The loss in the identification accuracy is acceptable for the gain in the data-to-simulation agreement for many analyses that do not include scale factors due to less complexity and time saving. Moreover, it is not certain whether the standard DeepCSV algorithm in combination with scale factors leads to better data-to-simulation agreement than the DA-DeepCSV algorithm without applying any scale factor. This investigation is left for future research.

If the systematic uncertainties of the DeepCSV algorithm were reduced with the domain-adversarial method, which is expected, it would be worth to apply this model on all analyses that are limited by their systematic uncertainties. A further investigation of the DeepCSV algorithm with the domain-adversarial extension on its scale factors and systematic uncertainties should be performed. One obstacle for the domain-adversarial method, as applied to the DeepCSV algorithm, is the increased time spent on training. An improvement of the work flow is needed to apply the domain-adversarial method in the same way to the DeepFlavour model.

List of Tables

List of Figures

Bibliography

[1] The ATLAS Collaboration, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC", *Physics Letters B*, vol. 716, pp. 1–29, 2012.

[2] The CMS Collaboration, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC", *Physics Letters B*, vol. 716, pp. 30–61, 2012.

[3] The CMS Collaboration, "Observation of Higgs boson decay to bottom quarks", 2018.

[4] The CMS Collaboration, "Observation of $H \to b\bar{b}$ decays and VH production with the ATLAS detector", 2018.

[5] The CMS Collaboration, "Search for the pair production of third-generation squarks with two-body decays to a bottom or charm quark and a neutralino in proton–proton collisions at $\sqrt{s}$ = 13 TeV", *Physics Letters B*, vol. 778, pp. 263–291, 2018.

[6] The ATLAS Collaboration, "Search for the Decay of the Higgs Boson to Charm Quarks with the ATLAS Experiment", *Physical Review Letters*, vol. 120, no. 21, p. 211802, 2018.

[7] F. Abe *et al.*, "Observation of top quark production in $\bar{p}p$ collisions", *Physical Review Letters*, vol. 74, pp. 2626–2631, 1995.

[8] K. Kodama *et al.*, "Observation of tau neutrino interactions", *Physics Letters B*, vol. 504, pp. 218–224, 2001.

[9] D. Hanneke, S. F. Hoogerheide, and G. Gabrielse, "Cavity Control of a Single-Electron Quantum Cyclotron: Measuring the Electron Magnetic Moment", *Physical Review A*, vol. 83, p. 052122, 2011.

[10] F. Halzen and A. D. Martin, "Quarks and Leptons: An Introductory Course in Modern Particle Physics", 1984.

[11] G. L. Kane, "Modern elementary particle physics: the fundamental particles and forces?", updated ed., 4. pr. ed., 1998.

[12] J. Schwichtenberg, "Physics from Symmetry", Undergraduate Lecture Notes in Physics, Springer, Cham, 2nd ed. 2018 ed., 2018.

[13] M. E. Peskin and D. V. Schroeder, "An Introduction to quantum field theory", Addison-Wesley, Reading, USA, 1995.

[14] M. Tanabashi *et al.*, "Review of Particle Physics", *Phys. Rev. D*, vol. 98, p. 030001, Aug 2018.

[15] A. Quadt, "Top quark physics at hadron colliders", *The European Physical Journal C - Particles and Fields*, vol. 48, pp. 835–1000, Dec 2006.

[16] CERN - Education, Communications and Outreach Group, "LHC Guide", Mar 2017. http://cds.cern.ch/record/2255762, last accessed on 2018-06-21.

[17] CERN, "The Large Hadron Collider", 2014. http://home.cern/topics/large-hadron-collider, last accessed on 2018-06-07.

[18] C. De Melis, "The CERN accelerator complex. Complexe des accélérateurs du CERN", 2016. https://cds.cern.ch/record/2197559, last accessed on 2018-06-07.

[19] Lujan P., "CMS Luminosity - Public Results", 2018. https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults#2016_proton_proton_13_TeV_collis, last accessed on 2018-06-28.

[20] The CMS Collaboration, "The CMS Experiment at the CERN LHC", *Journal of Instrumentation*, vol. 3, p. S08004, 2008.

[21] The CMS Collaboration, "Search for the tH(H$\rightarrow$ b$\bar{\text{b}}$) process in pp collisions at $\sqrt{s} = 13$ TeV and study of Higgs boson couplings", CMS-PAS-HIG-17-016, CERN, Geneva, 2018.

[22] S. R. Davis, "Interactive Slice of the CMS detector", 2016. https://cds.cern.ch/record/2205172, last accessed on 2018-06-08.

[23] The CMS Collaboration, "The CMS tracker system project: Technical Design Report", Technical Design Report CMS, CERN, Geneva, 1997.

[24] The CMS Collaboration, "Description and performance of track and primary-vertex reconstruction with the CMS tracker", *Journal of Instrumentation*, vol. 9, p. P10009, 2014.

[25] The CMS Collaboration, "The CMS electromagnetic calorimeter project: Technical Design Report", Technical Design Report CMS, Geneva, 1997.

[26] The CMS Collaboration, "The CMS hadron calorimeter project: Technical Design Report", Technical Design Report CMS, Geneva, 1997.

[27] The CMS Collaboration, "The CMS muon project: Technical Design Report", Technical Design Report CMS, Geneva, 1997.

[28] The CMS Collaboration, "The CMS trigger system", *Journal of Instrumentation*, vol. 12, no. 01, p. P01020, 2017.

[29] A. Perrotta, "Performance of the CMS High Level Trigger", *Journal of Physics: Conference Series*, vol. 664, no. 8, p. 082044, 2015.

[30] S. Dulat *et al.*, "New parton distribution functions from a global analysis of quantum chromodynamics", *Physical Review Journals D*, vol. 93, no. 3, p. 033006, 2016.

[31] R. D. Ball *et al.*, "Parton distributions from high-precision collider data", *The European Physical Journal C*, vol. 77, no. 10, p. 663, 2017.

[32] Y. L. Dokshitzer, "Calculation of the Structure Functions for Deep Inelastic Scattering and e+ e- Annihilation by Perturbation Theory in Quantum Chromodynamics.", *Soviet Physics*, vol. 46, pp. 641–653, 1977.

[33] V. N. Gribov and L. N. Lipatov, "Deep inelastic e p scattering in perturbation theory", *Soviet Journal of Nuclear Physics*, vol. 15, pp. 438–450, 1972. [Yad. Fiz.15,781(1972)].

[34] G. Altarelli and G. Parisi, "Asymptotic Freedom in Parton Language", *Nuclear Physics B*, vol. 126, pp. 298–318, 1977.

[35] N. Metropolis and S. Ulam, "The monte carlo method", *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[36] M. Bachtis, "Event Simulation", pp. 47–54. Springer International Publishing, Cham, 2014.

[37] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand, "Parton fragmentation and string dynamics", *Physics Reports*, vol. 97, no. 2, pp. 31 – 145, 1983.

[38] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, and T. Stelzer, "MadGraph 5 : Going Beyond", *Journal of High Energy Physics*, vol. 06, p. 128, 2011.

[39] F. Maltoni and T. Stelzer, "MadEvent: Automatic event generation with MadGraph", *Journal of High Energy Physics*, vol. 02, p. 027, 2003.

[40] J. Alwall *et al.*, "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations", *Journal of High Energy Physics*, vol. 07, p. 079, 2014.

[41] P. Nason, "A New method for combining NLO QCD with shower Monte Carlo algorithms", *Journal of High Energy Physics*, vol. 11, p. 040, 2004.

[42] S. Frixione, P. Nason, and C. Oleari, "Matching NLO QCD computations with Parton Shower simulations: the POWHEG method", *Journal of High Energy Physics*, vol. 11, p. 070, 2007.

[43] S. Alioli, P. Nason, C. Oleari, and E. Re, "A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX", *Journal of High Energy Physics*, vol. 06, p. 043, 2010.

[44] T. Sjöstrand *et al.*, "An Introduction to PYTHIA 8.2", *Comput. Phys. Commun.*, vol. 191, pp. 159–177, 2015.

[45] S. Agostinelli *et al.*, "G4 - a simulation toolkit", vol. 506, pp. 250–303, 07 2003.

[46] Frühwirth, R., "Application of Kalman filtering to track and vertex fitting", *Nuclear Instruments and Methods A*, vol. 262, pp. 444–450, 1987.

[47] The CMS Collaboration, "Particle-Flow Event Reconstruction in CMS and Performance for Jets, Taus, and MET", Physics Analysis Summaries CMS, 2009.

[48] The CMS Collaboration, "Particle-flow reconstruction and global event description with the CMS detector", *Journal of Instrumentation*, vol. 12, no. 10, p. P10003, 2017.

[49] The CMS Collaboration, "Electron reconstruction and identification at sqrt(s) = 7 TeV", 2010. CMS-PAS-EGM-10-004.

[50] M. Cacciari, G. P. Salam, and G. Soyez, "The Anti-k(t) jet clustering algorithm", *Journal of High Energy Physics*, vol. 04, p. 063, 2008.

[51] G. Salam , "Jets", 2008. `https://gsalam.web.cern.ch/gsalam/repository/talks/2008-cteq-mcnet.pdf`, last accessed on 2018-07-15.

[52] The CMS collaboration, "Determination of jet energy calibration and transverse momentum resolution in CMS", *Journal of Instrumentation*, vol. 6, no. 11, p. P11002, 2011.

[53] V. Sordini, "JEC levels", 2016. `https://twiki.cern.ch/twiki/bin/viewauth/CMS/IntroToJEC`, last accessed on 2018-07-16.

[54] C. M. Bishop, "Pattern recognition and machine learning", Information science and statistics, Springer, New York, 2006.

[55] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning", 2016. `http://www.deeplearningbook.org`, last accessed on 2018-07-26.

[56] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain", *Psychological Review*, pp. 65–386, 1958.

[57] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen". diploma thesis, Technische Universität München, April 1991.

[58] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions", *Computing Research Repository*, vol. abs/1710.05941, 2017.

[59] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation*, vol. 1, pp. 541–551, Dec 1989.

[60] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *Computing Research Repository*, vol. abs/1412.6980, 2014.

[61] L. Prechelt, "Early Stopping — But When?", pp. 53–67. Springer, Berlin, Heidelberg, 2012.

[62] G. E. Hinton *et al.*, "Improving neural networks by preventing co-adaptation of feature detectors", *Computing Research Repository*, vol. abs/1207.0580, 2012.

[63] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *Computing Research Repository*, vol. abs/1502.03167, 2015.

[64] E. Tzeng *et al.*, "Adversarial Discriminative Domain Adaptation", *Computing Research Repository*, vol. abs/1702.05464, 2017.

[65] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function", *Journal of Statistical Planning and Inference*, vol. 90, pp. 227–244, 2000.

[66] S. J. Pan *et al.*, "Domain Adaptation via Transfer Component Analysis", *IEEE Transactions on Neural Networks*, vol. 22, pp. 199–210, Feb 2011.

[67] Y. Ganin *et al.*, "Domain-Adversarial Training of Neural Networks", *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[68] K. Borgwardt *et al.*, "Integrating Structured Biological data by Kernel Maximum Mean Discrepancy", *Bioinformatics*, vol. 22, pp. e49–e57, Aug. 2006.

[69] E. Tzeng *et al.*, "Deep Domain Confusion: Maximizing for Domain Invariance", *Computing Research Repository*, vol. abs/1412.3474, 2014.

[70] M. Long and J. Wang, "Learning transferable features with deep adaptation networks", *Computing Research Repository*, vol. abs/1502.02791, 2015.

[71] Y. Ganin and V. Lempitsky, "Unsupervised Domain Adaptation by Backpropagation", *Journal of Machine Learning Research*, Sept. 2014.

[72] I. J. Goodfellow *et al.*, "Generative Adversarial Networks", *ArXiv e-prints*, June 2014. https://arxiv.org/abs/1406.2661, last accessed on 2018-08-01.

[73] The CMS Collaboration, "Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV", *Journal of Instrumentation*, vol. 13, no. 05, p. P05011, 2018.

[74] The CMS Collaboration, "Measurement of $B\bar{B}$ Angular Correlations based on Secondary Vertex Reconstruction at $\sqrt{s} = 7$ TeV", *Journal of High Energy Physics*, vol. 03, p. 136, 2011.

[75] F. Chollet *et al.*, "Keras", 2015. https://keras.io, last accessed on 2018-07-28.

[76] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems", 2015. https://www.tensorflow.org/, last accessed on 2018-07-28.

[77] The CMS Collaboration, "CMS Phase 1 heavy flavour identification performance and developments", Detector Performance Summary CMS, CERN, Geneva, May 2017.

[78] D. Guest *et al.*, "Jet flavor classification in high-energy physics with deep neural networks", *Physical Review D*, vol. 94, p. 112002, Dec 2016.

[79] T. Speer, "Jet Flavour Identification (MC Truth)", 2018. https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideBTagMCTools, last accessed on 2018-07-31.

[80] The CMS Collaboration, "Measurement of the $t\bar{t}$ production cross section using events with one lepton and at least one jet in pp collisions at $\sqrt{s}$ = 13 TeV", *Journal of High Energy Physics*, vol. 09, p. 051, 2017.

[81] The CMS Collaboration, "Measurement of the $t\bar{t}$ production cross section using events in the eμ final state in pp collisions at $\sqrt{s} = 13$ TeV", *The European Physical Journal C*, vol. 77, p. 172, 2017.

[82] M. Czakon and A. Mitov, "Top++: A program for the calculation of the top-pair cross-section at hadron colliders", *Computer Physics Communications*, vol. 185, no. 11, pp. 2930 – 2938, 2014.

[83] CERN, "Brilview" https://brilview.readthedocs.io/en/stable/, last accessed on 2018-08-03.

[84] R. Gavin, Y. Li, F. Petriello, and S. Quackenbush, "Fewz 2.0: A code for hadronic z production at next-to-next-to-leading order", *Computer Physics Communications*, vol. 182, no. 11, pp. 2388 – 2403, 2011.

[85] The CMS Collaboration, "GenXSecAnalyzer" `https://github.com/cms-sw/cmssw/blob/master/GeneratorInterface/Core/plugins/GenXSecAnalyzer.cc`, last accessed on 2018-09-06.

[86] P. Kant *et al.*, "HatHor for single top-quark production: Updated predictions and uncertainty estimates for single top-quark production in hadronic collisions", *Computer Physics Communications*, vol. 191, pp. 74–89, 2015.

[87] M. Aliev *et al.*, "HATHOR: HAdronic Top and Heavy quarks crOss section calculatoR", *Computer Physics Communications*, vol. 182, pp. 1034–1046, 2011.

[88] J. M. Campbell, R. K. Ellis, and C. Williams, "Vector boson pair production at the LHC", *Journal of High Energy Physics*, vol. 2011, p. 18, Jul 2011.

[89] A. N. Kolmogorov, "Sulla Determinazione Empirica di una Legge di Distribuzione", *Giornale dell'Istituto Italiano degli Attuari*, vol. 4, pp. 83–91, 1933.

[90] N. Smirnov, "Table for Estimating the Goodness of Fit of Empirical Distributions", *Annals of Mathematical Statistics*, vol. 19, pp. 279–281, 06 1948.

[91] NVIDIA Corporation, "NVIDIA: GeForce GTX TITAN X", 2018. `https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-x`, last accessed on 2018-08-13.

Acknowledgements

My thanks goes to everyone from the institute of particle physics for the pleasant working environment. Many thanks goes especially to:

Prof. Dr. Thomas Müller for leading the institute.

The administration team, for maintaining the infrastructure.

Dr. Matthias Mozer for giving me the possibility to do this thesis, for his supervision and the final proofreading of my thesis.

Prof. Dr. Günter Quast for taking the position of the second referee of this thesis.

Dr. Markus Stoye, Dr. Jan Kieseler and Dr. Mauro Verzetti, who had the initial ideas for this work, for the proposals during my work and the help for problems especially with the framework.

Dennis Weyland and Philipp Ott for their discussions at the early stage of my work.

Dr. Nils Faltermann, Dr. Soureek Mitra, David Seith, Denise Müller, Johann Rauser, Jochen Gemmler and Darius Bühler for proofreading my thesis and the many helpful hints during my work.

The lunch group, including Dr. Matthias Mozer, Dr. Thorsten Chwalek, Dr. Nils Faltermann, Dr. Soureek Mitra, David Seith, Denise Müller, Johann Rauser, Darius Bühler, Ansar Iqbal, Kevin Flöh, Max Neukum and Genti Saliu especially for the times, not going to mensa.

I also thank my parents for the pocket money and for always giving me support. Without their backing, this work would not have been possible.