Universität Karlsruhe (TH)

Analyse des $t\bar{t}H$-Kanals am CMS-Detektor des LHC mit Neuronalen Netzen

Dennis Schieferdecker

Diplomarbeit

an der Fakultät für Physik
der Universität Karlsruhe

Referent: Prof. Dr. Günter Quast
Institut für Experimentelle Kernphysik

Korreferent: Prof. Dr. Michael Feindt
Institut für Experimentelle Kernphsyik

1. Februar 2006

Deutsche Zusammenfassung

Die Entdeckung atomarer und subatomarer Strukturen hat einen wesentlichen Beitrag zum Fortschritt der Menschheit im letzten Jahrhundert geleistet. Elektronik, neue Materialien und die moderne Medizin sind letztendlich alle der fortwährenden Forschung im Bereich der Teilchenphysik zu verdanken.

Die Physik bietet heute ein sehr gutes Verständnis der elementaren Bausteine der Natur in Form des Standardmodells der Teilchenphysik (SM). Die Vorhersagen dieses Modells wurden bisher größtenteils mit Hilfe von Teilchenbeschleunigern überprüft, indem Teilchenstrahlen bei sehr hohen Energien auf ein festes Ziel geschossen oder zur Kollision miteinander gebracht und die entstandenen Zerfallsprodukte untersucht wurden. So wurden zum Beispiel die Austauschteilchen der schwachen Wechselwirkung, das W- und das Z-Boson, 1983 an einem Ringbeschleuniger am CERN[1] in Genf entdeckt. Seit damals konnten der später am CERN errichtete "Large Electron Positron" Beschleuniger (LEP) sowie der "Stanford Linear Collider" und das Tevatron am Fermilab in den USA weitere Erfolge in der Erforschung der kleinsten Teilchen und ihrer Wechselwirkungen erzielen.

Bisher konnten alle Experimente die Aussagen des Standardmodells bestätigen. Es gibt aber noch viele ungeklärte Fragen. Zum Beispiel, warum besitzen Teilchen Masse und woher kommt diese? Momentan wird davon ausgegangen, dass ein weiteres Teilchen, das so genannte Higgsboson existiert, durch das die anderen Teilchen ihre Masse erlangen. Weitere ungelöste Probleme umfassen die Existenz und Beschaffenheit der dunklen Materie und Energie sowie die Vereinigung aller vier fundamentalen Kräfte in einer gemeinsamen Theorie, der "Great Unified Theory" (GUT).

Schon heute wird an bestehenden Teilchenbeschleunigern nach Hinweisen für die Existenz des Higgsbosons gesucht. Für eine gründlichere Untersuchung wird aber ein Beschleuniger mit einer wesentlich höheren Energie, wie zum Beispiel den momentan im Bau befindlichen "Large Hadron Collider" (LHC) benötigt. Er wird am CERN im ehemaligen Tunnel des LEP aufgebaut und soll 2007 in Betrieb genommen werden. In ihm werden zukünftig Protonen mit einer Schwerpunktsenergie von 14 TeV zur Kollision gebracht, um anschließend die entstehenden Zerfallsprodukte in Detektoren aufzuzeichnen.

Eines der vier am Beschleunigerring errichteten Experimente ist der "Compact Muon Solenoid" (CMS) Detektor. Hierbei handelt es sich im Prinzip um einen Allzweckdetektor, der allerdings

[1] Conseil Européen pour la Recherche Nucléaire

für die Entdeckung des Higgsbosons bis hin zu einer Masse von $1\text{TeV}/c^2$ optimiert ist. Sobald der Detektor in Betrieb geht, wird er riesige Mengen an Daten produzieren. Davon wird zwar nur weit weniger als ein Promille zur weiteren Analyse gespeichert werden, dabei handelt es sich allerdings immer noch um mehrere Petabytes pro Jahr. Um diese Herausforderung zu meistern, mussten neue Technologien entwickelt werden, angefangen bei der Detektorelektronik bis hin zu modernen Grid-Technologien für die weltweit verteilte Auswertung und Speicherung der Daten.

Neben der Technik mußten allerdings auch bestehende Analysemethoden weiterentwickelt werden, um die Daten, die der CMS Detektor liefern wird, auswerten zu können. In früheren Analysen wurden einfache Schnitte auf Variablenwerten durchgeführt, um Untergrundereignisse zu selektieren und zu entfernen und dadurch ein klareres Signal zu erlangen. Später wurden "Likelihood"-Werte berechnet, um die in den Daten enthaltenen Informationen besser ausnutzen zu können. Neuronale Netze bieten sich als weitere Methode zu Datenanalyse an. Sie sind mittlerweile gründlich nach wissenschaftlichen Standpunkten erforscht worden und stellen ein ausgereiftes Analysewerkzeug dar. Ein Netz ermöglicht es, die in einem Datensatz enthaltenen Informationen vollständig auszureizen – falls es korrekt verwendet worden sind. Dieser Ansatz wird in der Hochenergie-Teilchenphysik seit etwa einem Jahrzehnt mit wachsendem Erfolg benutzt.

Die vorliegende Arbeit handelt vom Gebrauch aktueller Netztechnologien im Bereich der Teilchenphysik. Im Speziellen wird ihre Anwendbarkeit auf bestimmte Analyseaufgaben im Rahmen des CMS Experiment untersucht. Konkret wird der semileptonische $t\bar{t}H$ Zerfallskanal mit einem Muon im Endzustand betrachtet. Hierbei handelt es sich um die erste Anwendung neuronaler Netze in der CMS Gruppe am Institut für Experimentelle Kernphysik in Karlsruhe (IEKP), um zukünftige Analysen mit neuronalen Netzen auf diesem Gebiet zu erleichtern. Es werden zwei unterschiedliche Netzwerkpakete verwendet und ihre Leistungsfähigkeit verglichen, namentlich NeuroBayes®, das von der Firma $<\text{Phi}-\text{T}>$ vertrieben wird, sowie die Klasse `TMultiLayerPerceptron` des kostenlosen Analysepakets ROOT.

Der $t\bar{t}H$-Kanal am LHC hat das Potential zu einer Entdeckung des Higgsbosons im Bereich kleiner Massen unterhalb von 200 GeV/c^2 beizutragen. Er wurde zusammen mit dem zugehörigen $t\bar{t}b\bar{b}$ Untergrundkanal als idealer Anwendungsfall für neuronale Netze ausgewählt, denn auf Grund seiner Komplexität können die Netze ihre Fähigkeiten voll ausnutzen. Der Endzustand dieses Kanals umfasst sechs hadronische Jets, ein Lepton und ein Neutrino. Diese müssen den detektierten Teilchen zugeordnet werden, was aufgrund der vielen kombinatorischen Möglichkeiten erhebliche Schwierigkeiten bereitet. Die daraufhin folgende Unterscheidung in Signal- und Untergrundereignisse stellt eine weitere Herausforderung dar, da ihre Endzustände eine sehr ähnliche Topologie und Kinematik aufweisen.

Deshalb wurde zuerst ein neuronales Netz entworfen, dass die detektierten Teilchen korrekt den Teilchen im Endzustand des $t\bar{t}H$- bzw. $t\bar{t}b\bar{b}$-Zerfalls zuordnen soll. Dieser Schritt ist besonders wichtig, da jede weitere Analyse die korrekte Teilchenzuordnung benötigt. Eine bereits existierende Analyse basierend auf "Likelihood"-Werten [1] wurde hierfür auf ein neuronales Netz übertragen und erweitert. Die Leistungsfähigkeit dieses Netzes ist in Abb. 1 dargestellt. Die korrekte Teilchenzuordnung gelingt in 33,4% aller Fälle für Signalereignisse und für 31,6% der

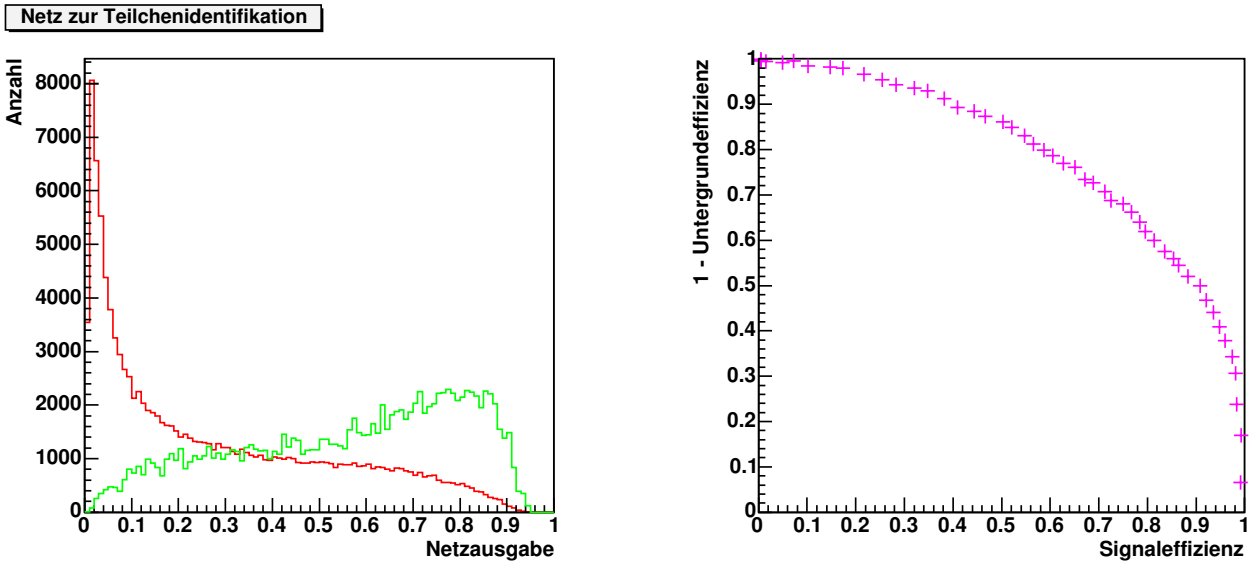

Figure 1: *Auf der linken Seite ist Ausgabe der Analyse zur Teilchenzuordnung dargestellt (Signal: grün, Untergrund: rot). Auf der rechten Seite ist das zugehörige Effizienzdiagramm aufgetragen.*

Untergrundereignisse. Dies ist vergleichbar zu den Ergebnissen der vorherigen Analyse. Zudem wurde festgestellt, dass sich das `TMultiLayerPerceptron` Netz nicht zur Bewältigung dieser Aufgabe eignet.

Als zweites wurde ein neuronales Netz für die eigentliche Analyseaufgabe dieser Arbeit konstruiert. Sein Ziel ist die Unterscheidung von $t\bar{t}H$ Signal- und $t\bar{t}b\bar{b}$ Untergrundereignissen, um sie anschließend voneinander trennen zu können. Hierfür wurden aufbauend auf einer CMS Note [2] weitere Variablen bestimmt, die hilfreich für die Trennung der Ereignisse sind.

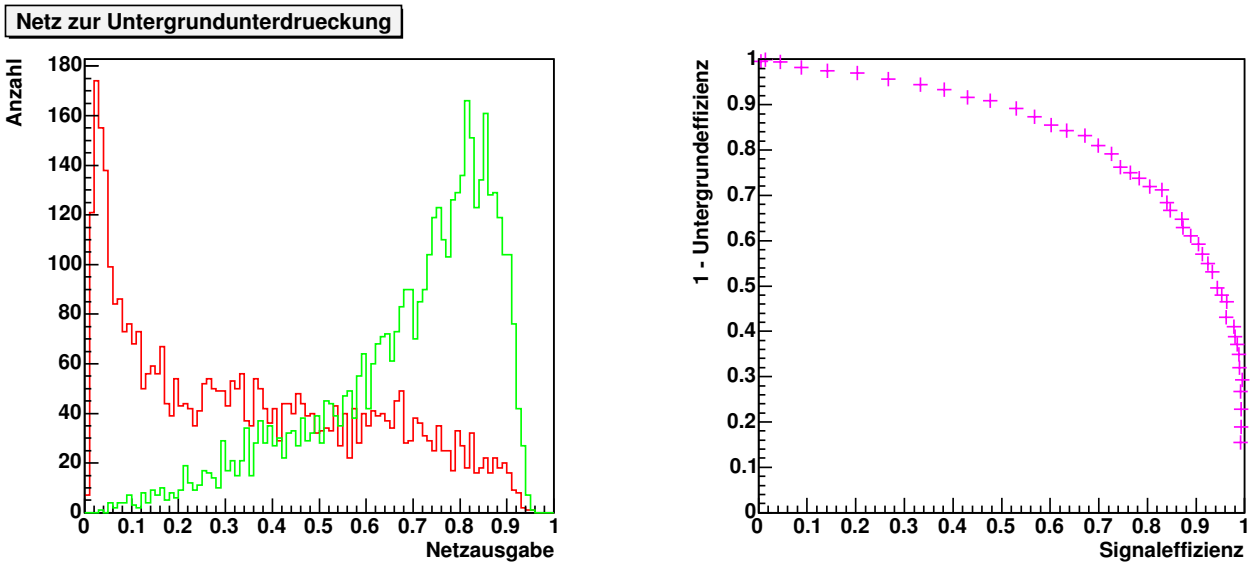

Figure 2: *Auf der linken Seite ist Netzausgabe der Analyse zur Untergrundunterdrückung dargestellt (Signal: grün, Untergrund: rot). Auf der rechten Seite ist das zugehörige Effizienzdiagramm aufgetragen.*

Die Leistung des erstellten Netzes ist gut, wie man Abb. 2 entnehmen kann. Das zu erwartende Signal zu Wurzel über Untergrund Verhältnis nach einem Jahr der Datennahme am CMS Detekor bei niedriger Lumninosität liegt bei 10,8. Um die invariante Masse des Higgsbosons zu bestimmen, wurden alle Ereignisse mit einer Netzausgabe kleiner 0.4 als Untergrund klassifziert, was einer Signaleffizienz von 0.9 entspricht. Die aus den restlichen Ereignissen rekonstruierte Higgsmasse ergibt $m_{Higgs} = 100.0 \pm 0.5\ GeV/c^2$ zusammen mit einer Standardabweichung von $\sigma_{Higgs} = 30.7 \pm 0.4\ GeV/c^2$. Dies entspricht nahezu dem erwarteten Wert nach der Detektorsimulation von $m_{Higgs} = 105.1 \pm 0.4\ GeV/c^2$ mit $\sigma_{Higgs} = 23.0 \pm 0.3\ GeV/c^2$ und stellt somit ein sehr gutes Ergebnis dar. Die angegebenen Werte wurden hierbei mit Hilfe einer Gauss-Ausgleichskurve aus Abb. 3 bestimmt. Diese Analyse konnte mit beiden Netzwerkpakete erfolgreich durchgeführt werden.

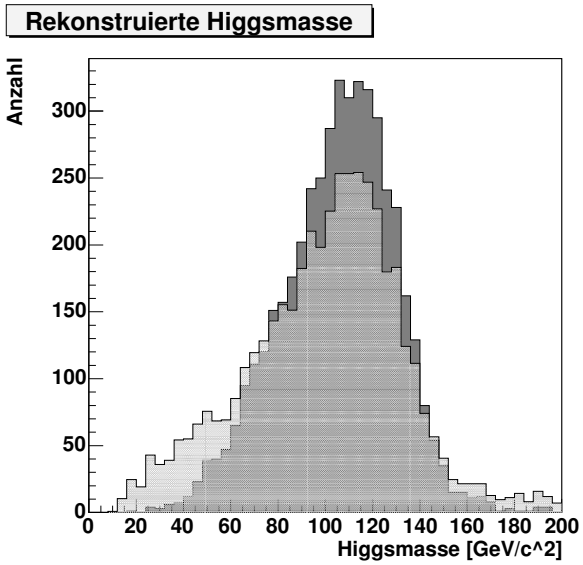

Figure 3: *Verteilung der Higgsmasse (dunkel: simuliert, hell: rekonstruiert).*

Nachdem beide neuronalen Netze entworfen, trainiert und ausgewertet worden sind, wurden sie zu einer kombinierten Analyse verbunden. Hierbei erhält das Netz zur Untergrundunterdrückung die Teilchenidentifikationen des anderen Netzes, anstatt wie bisher auf die wahren Werte zurück-

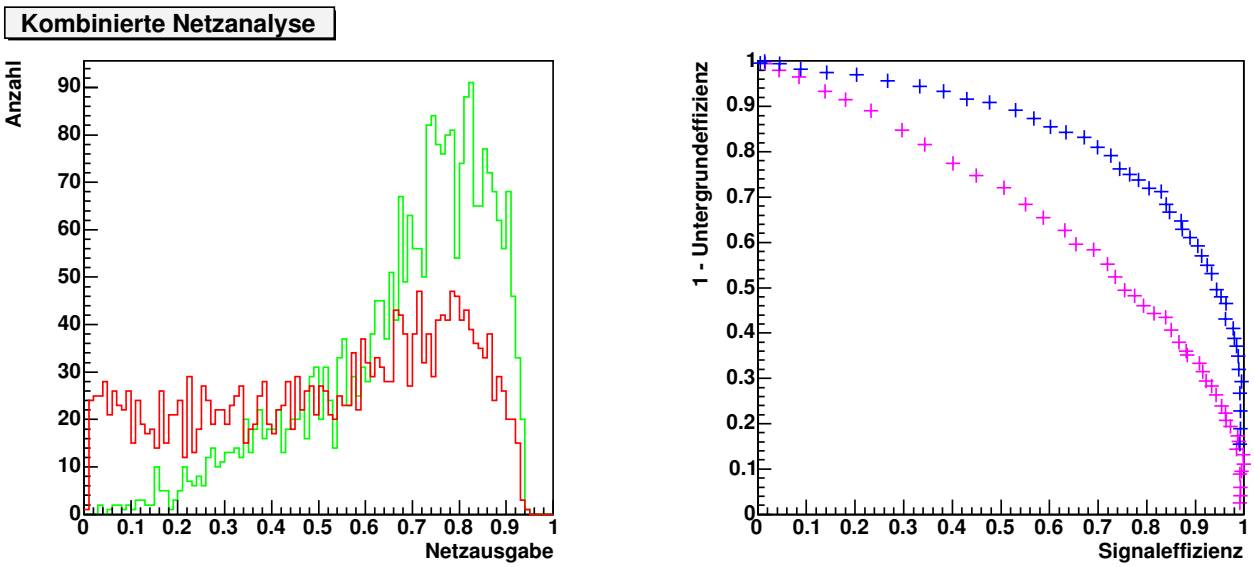

Figure 4: *Auf der linken Seite ist die Netzausgabe der kombinierten Analyse dargestellt (Signal: grün, Untergrund: rot). Auf der rechten Seite ist das zugehörige Effizienzdiagramm aufgetragen (pink: kombinierte Analyse, blau: Analyse unter Verwendung der wahren Teilchenidentitäten).*

zugreifen. Das so entstandene Netz ist weiterhin in der Lage die Analyseaufgabe – im Rahmen der sich geänderten Voraussetzungen – zu meistern (Abb. 4). In der Verteilung der rekonstruierten Higgsmasse in Abb. 5 lässt sich allerdings eine deutliche Verschiebung zu niedrigeren Werten erkennen. Diese ist auf die nicht perfekte Zuordnung von detektierten Teilchen zu den simulierten Teilchen des $t\bar{t}H$ bzw. $t\bar{t}b\bar{b}$ Endzustandes zurückzuführen. Wie man an der gestrichelten Linie erkennen kann, tritt diese Verschiebung der Higgsmasse direkt nach der Teilchenzuordnung in den Signalereignissen auf und rührt somit nicht von einer schlechten Untergrundunterdrückung her. Somit ist die mangelnde Teilchenidentifikation die entscheidende Schwachstelle der Analyse. Aus diesem Grund wird eine Verbesserung in diesem Punkt als nächstes Ziel angestrebt.

Die vorliegende Arbeit hat gezeigt, dass neuronale Netze auch im Bereich der Hochenergiephysik ein nützliches und mächtiges Werkzeug zur Datenanalyse darstellen. Sie können die mit herkömmlichen Analysemethoden erzielten Ergebnisse reproduzieren und sogar noch übertreffen. Dabei bieten beide entworfenen Netze noch Raum für zusätzliche Optimierungen. Es existieren noch weitere interessante Variablen in den Ereignissen und Optionen der Netzwerkpakete, die untersucht werden können.

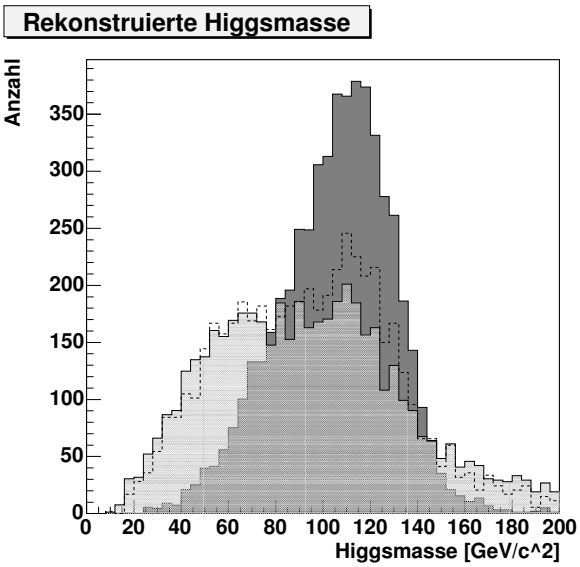

Figure 5: *Verteilung der Higgsmasse (dunkel: simuliert, gestrichelt: nach der Teilchenzuordnung, hell: rekonstruiert).*

Während dieser Diplomarbeit wurde zudem ein Framework für neuronale Netze entwickelt, dass es erlaubt Analysen unabhängig von dem tatsächlich verwendeten Netzwerkpaket durchzuführen. Eine Schnittstelle für die beiden benutzten Pakete wurde bereits entworfen, um ihre Leistungsfähigkeit einfach vergleichen zu können. In Zukunft können weitere Netzwerkpakete in dieses Framework eingebracht werden, so dass das jeweils beste Netz für eine bestimmte Aufgabe herangezogen werden kann, ohne jeweils eine komplett andere Programmierschnittstelle verwenden zu müssen. Zudem kann das Framework einfach um weitere Analyse-Funktionalitäten ergänzt werden.

Universität Karlsruhe (TH)

Analysis of the $t\bar{t}H$ Channel at the CMS Detector of LHC with Neural Networks

Dennis Schieferdecker

Diploma Thesis

on the Department of Physics
at the University of Karlsruhe

Referee: Prof. Dr. Günter Quast
Institut für Experimentelle Kernphysik

Coreferee: Prof. Dr. Michael Feindt
Institut für Experimentelle Kernphsyik

1st February 2006

"I hear and I forget,
I see and I remember,
I do and I understand."
(accredited to Confucius)

Contents

Introduction

The discovery of the atomic structure of matter has played a large part in human's progress in the last century. Electronics, new materials and modern medicine are ultimately all the result of the continuous expansion of our understanding of the fundamental particles and their interactions.

Physicists now have a good comprehension of nature's basic building blocks, compiled in the Standard Model of particle physics (SM). The predictions of this model thus far have all been verified by observing collisions of particle beams at very high energies in large accelerators. For example, the W and Z bosons which mediate the weak force have been discovered in 1983 using a circular particle collider at CERN[2] in Geneva. Since then the Large Electron Positron Collider (LEP) subsequently built at CERN, and also the Stanford Linear Collider and the Tevatron Collider at Fermilab in the USA, have further probed the properties of these particles and found hard evidence for the existence of the other proposed particles.

Up to now, all these experiments have indicated that the Standard Model works well. However, it is far from complete and there are still many unanswered questions remaining. For instance, why do particles have mass and what is its origin? The currently most accepted theory implies the existence of a new particle called the Higgs boson that gives mass to the other particles through interactions with them. Other open questions include the imbalance between matter and anti-matter in the universe or the unification of four interactions in one ultimate theory, the Great Unifying Theory (GUT).

Physicists are already searching for evidences of the Higgs boson at existing colliders. However, for a more thorough investigation a collider with much higher energies is required. This machine, the Large Hadron Collider (LHC), is now being built at the CERN site in the former LEP tunnel. It will collide intense beams of protons at a centre of mass energy of 14 TeV.

One of the four experiments installed at the LHC ring will be the Compact Muon Solenoid (CMS) detector. It is a multipurpose detector optimised for the discovery of the Higgs boson at masses of up to 1 TeV/c^2. When the collider takes up its service, the CMS detector will record a large amount of data each second. Far less than a promille will even be stored for investigation by offline end-user analysing software. This data will still range in the region of millions of gigabytes each year. To meet this challenge, new technologies had to be devised, from detector electronics up to grid technologies for worldwide computation and storage. But not only technology has to scale with the requirements, analysing methods have too.

[2]Conseil Européen pour la Recherche Nucléaire

Early analyses in former experiments have used simple cuts on variables to enrich signal events and suppress background events. Later, likelihood ratios were calculated to make better use of the information inherent to the recorded data. By now, neural networks have been studied to a great extent and their capabilities have been scientifically evaluated. Originally designed to mimic the functionality of the human brain, they now represent a sophisticated method for data analyses, able to take full advantage of all available information – if handled correctly. This neural network approach to data analysis has been used in the field of high energy physics with growing approval since about a decade.

In this thesis current neural network technologies are applied and their usefulness for specific analysing problems in the upcoming CMS experiment are investigated. It is the first usage of neural networks in the CMS context at the Institut für Experimentelle Kernphysik in Karlsruhe (IEKP). Here their functionality is exploited to analyse the semi-leptonic $t\bar{t}H$ decay channel with a muon in the final state and to pave the way for future analysis with neural networks in this field.

The $t\bar{t}H$ decay channel at the LHC has the potential to contribute to a discovery of the Higgs boson in the lower mass ranges up to 200 GeV/c^2. It was chosen as a benchmark channel for this thesis along with one of its background channels ($t\bar{t}b\bar{b}$). Due to its complex topology consisting of six hadron jets, one lepton and missing energy in the final state, it poses an ideal field of application for neural networks. The allocation of detected hadrons to the original particles is difficult because of the large number of possible combinations. In addition the differentiation between signal and background events is challenging in itself, due to their final states having a similar topology and similar kinematics. Here neural networks will be able to fully utilise their classification capabilities.

At first, this thesis will give an overview of the future measuring instrument used, the CMS detector, followed by a brief introduction to the physics of the Higgs boson, needed for the comprehension of the following analysis. In particular, the properties and difficulties of detecting the Higgs boson at the LHC are described. Subsequently, neural network technologies, the basis for the analysing work in this thesis are elaborated and other used analysis tools are presented. The actual analysis is prefaced by an introduction to the pairing challenge of the examined decay channel and the neural network approach to it, followed by a study on suppressing background, also using neural networks as modus operandi. At the end, both neural networks are combined for the final analysis.

Further information about the implemented neural network interface can be found in the appendices. Here the actual network details and the data samples that were used for the training are also presented. In addition a comparison of the performance of two different neural network packages on the analysis task in this thesis is given.

The LHC and the CMS Experiment

The Compact Muon Solenoid (CMS) experiment [3] - [9], one of the four detectors that are currently being constructed at CERN[1] is going to be our eye in the Large Hadron Collider (LHC) [10] to detect new physics and validate existing theories, like the mechanism responsible for the masses of elementary particles. Therefore huge amounts of data will constantly be recorded and processed. The following sections will give a short overview of the capabilities of the LHC and the CMS detector and highlight the associated data volume that only modern information technologies and data analysis methods can handle.

1.1 The Large Hadron Collider (LHC)

Primarily designed as a proton-proton collider[2], the Large Hadron Collider situated at CERN near Geneva represents the currently largest project in collider physics (fig. 1.1). It is being installed between 30 m and 150 m below the surface in the tunnel where the Large Electron Positron (LEP) collider was previously located. It is expected to be brought online for the first time next year in 2007.

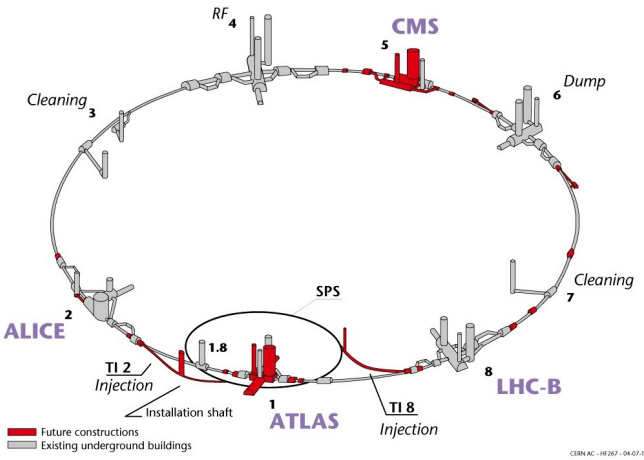

Figure 1.1: *Schematic overview of the Large Hadron Collider (LHC) at CERN [11].*

[1] Conseil Européen pour la Recherche Nucléaire

[2] In addition, heavy ion collisions (Pb-Pb) are also planned, but not part of this thesis.

Around the 27 km long tunnel, about 10,000 superconducting niobium-titanium magnets are installed. They will produce a magnetic field of about 8.33 T to focus the circulating particles and keep them on track – beam lifetimes of up to ten hours are expected. In the first years an operation at "low luminosity" with $L = 2 \cdot 10^{33}$ cm^{-2}s^{-1} and an integrated luminosity of $L_{int} = 20$ fb^{-1} per year is foreseen, followed by a "high luminosity" phase with $L = 10^{34}$ cm^{-2}s^{-1} and $L_{int} = 100$ fb^{-1} per year. This will result in 3.5 and 17.5 interactions per crossing, respectively.

The proton-proton collisions will be at a centre of mass energy of 14 TeV, which is about a factor of ten higher than at previous colliders like the Tevatron[3]. About 10^{11} protons are clustered into a bunch of which about 3000 are in the collider at once. They are accelerated and brought to collision with a frequency of 40 Mhz at four interaction points, where the detectors are located. Two of those, ALICE (A Large Ion Collider Experiment) and LHCb (Large Hadron Collider beauty experiment) have special purposes (heavy ion collisions and b-physics respectively), whereas the other two, ATLAS (A Toroidal LHC ApparatuS) and CMS are designed as complementary multipurpose detectors.

The experiments carried out at the LHC will likely be able to discover new physics. One primarily anticipates to find an evidence for the Higgs boson. This discovery would finally be the experimental proof for the mechanism of electroweak symmetry breaking, which explains the origin of mass of elementary particles. At the very least, the LHC will produce vast amounts of statistics that can be used for precise measurements of known physical parameters.

1.2 The Compact Muon Solenoid (CMS)

The CMS experiment is one of the largest scientific efforts of mankind. An international team of more than 2300 scientists and engineers of 160 institutes from 36 countries contribute to construct and operate this detector. Currently, the various modules of the detector are being built above ground at LHC access point 5 (Cessy, France) and then lowered into the detector cavern for final assemblage.

Although called "compact" it has an overall length of 22 m and a diameter of 15 m with a total mass of 12,500 t. Its central part is a superconducting solenoid, measuring 13 m in length, 6 m in height and weighing 5,000 t. It produces an axial magnetic field of 4 T that is returned via a 1.5 m thick saturated iron yoke with a mass of 7,000 t. With these specifications it is the world's largest superconducting solenoid, providing the highest stored energy of 2.6 GJ ever.

The detector itself can be divided into a barrel region and two endcap regions. Furthermore it is made up of many cylindrical layers around the beam pipe. Each layer is designed for a specific measuring task. Together, these layers will provide all the data allowing CMS to precisely measure all the detectable particles that the collisions at LHC will produce.

[3]The Tevatron is a proton - anti-proton collider at Fermilab, Chicago, with a centre of mass energy of $E_{CM} = 1.96$ TeV.

The various detector layers are described in the following subsections as well as the triggering system needed for data acquisition. For a more in-depth description of the CMS detector, its technical design report [TDRcms] can be consulted; in addition, a detailed overview of the detector is given in figures 1.2 and 1.3.

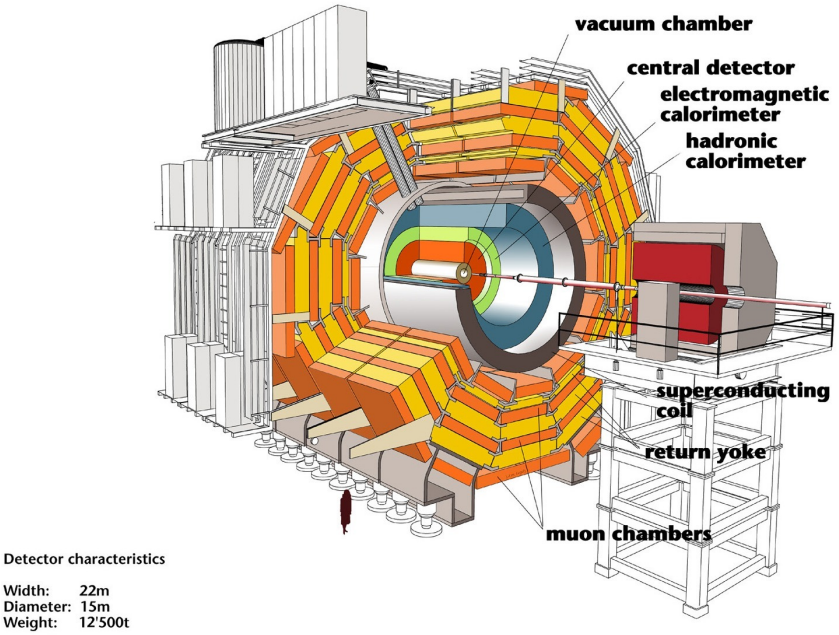

Figure 1.2: *Schematic overview of the Compact Muon Solenoid (CMS) detector [11].*

1.2.1 Tracking System

The tracking system is the innermost part of the detector, situated within the magnetic field of the solenoid. It is used for measuring tracks of charged particles. Thus, it needs to have a good spatial resolution and an efficient pattern recognition. In addition, it has to withstand the high level of radiation around the beam pipe for several years. The system also has to have a small response time in order to minimise pile-up[4] interfering with the measurement. Thus an all-silicon tracking system was chosen for the CMS detector, since it best fulfills the requirements.

Silicon Pixel Detector

The innermost detectors are the pixel detectors, designed for measuring the positions and thus the trajectories of particles. With this data, the particle momenta can then be calculated. In the barrel there are three layers of pixel detectors situated at radii of 4 cm, 7 cm and 11 cm.

[4] At the LHC several events can be in the detector at once due to the high bunch cross rate and also due to the high proton density, which leads to several collisions per bunch crossing. This effect is called pile-up.

The innermost and the middle layer are used for the lower luminosity phase; the middle and the outermost layer are then used for the higher luminosity phase, due to the higher radiation in this phase. In addition, there are two discs of pixel detectors installed in each of the endcaps covering radii from 6 cm to 15 cm.

Each pixel covers a surface of $(150 \times 150)\ \mu m^2$ which allows for a spatial resolution of about 10 μm and 14 μm in Φ and z coordinates, respectively[5]. On the whole, there are about fifty million single pixels that are read out.

Silicon Strip Detector

Ten layers of silicon microstrip detectors are located beyond the pixel detectors. There are 4 layers installed in the inner barrel and 6 layers in the outer barrel covering a radial range from 20 to 110 cm and a longitudinal range of $|z| < 280$ cm. The strips are oriented along the beam axis to allow measurements of the azimuthal coordinates. Each of the two innermost layers are two-sided with slightly shifted strips to allow measurements of the polar coordinate. Three small strip detector discs are located in the inner endcaps and six larger discs are built into the outer endcaps. Here, the strips are oriented radially to allow measurements of the Φ coordinate. In addition, double layers at the inner and outer regions of the endcap discs allow measurements of the radial coordinate. The endcap strip detectors cover a longitudinal range of 120 cm $< |z| <$ 280 cm.

In total there are 7,888 single-sided and 4,032 double-sided silicon strip modules built into the detector covering an area of about 210 m^2.

1.2.2 Calorimeters

Further outside but still within the magnetic field of the solenoid, the calorimeters are situated. They basically use a total-absorption method to measure the energy and direction of particles. A particle passing through the calorimeters will interact with the calorimeter material, producing particle showers. These showers will lead to ionisation and excitation in the calorimeter, which can be measured and used to determine the energy and direction of the particle.

There are two layers of calorimeters each with its own special purpose. The inner layer is designed to detect electrons, positrons and photons whereas the outer layer is designed to detect hadrons.

Electromagnetic Calorimeter (ECAL)

The electromagnetic calorimeter lies just outside of the tracking system. Here, electrons, positrons and photons produce electromagnetic showers as a result of interactions with the electrons in the calorimeter material.

[5] Typically only a resolution of about 43 μm is expected for a pixel width of 150 μm. This improvement is the result of purposefully not compensating for the large Lorentz angle which leads to significant charge sharing.

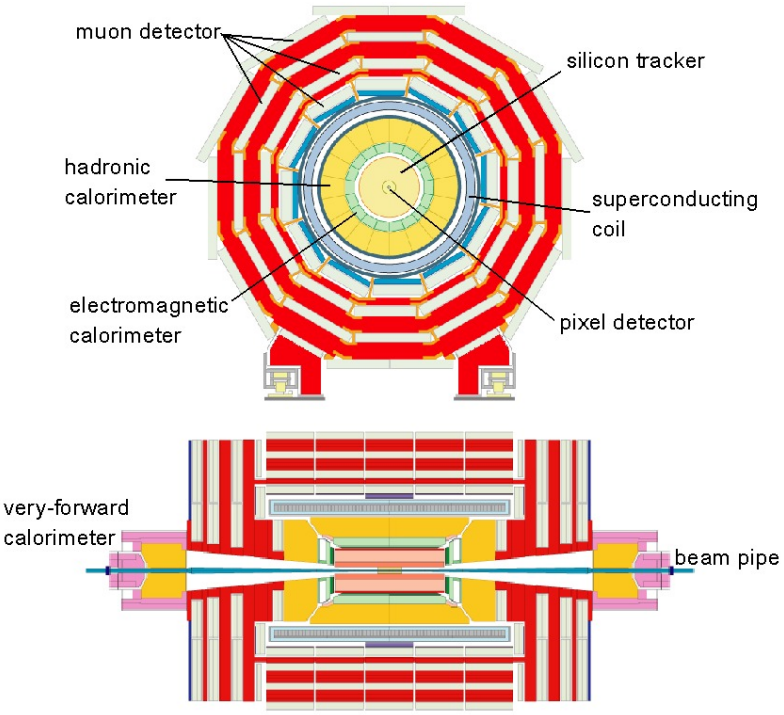

Figure 1.3: *Schematic cuts through the CMS detector across and along the beam pipe [11].*

The electromagnetic calorimeter is composed of 75,000 lead tungstate ($PbWO_4$) crystals, of which 60,000 can be found in the barrel region and 15,000 in the endcaps. $PbWO_4$ was chosen because of its high density of 8.28 g/cm^3 and its large average number of electrons per atom. This results in a short radiation length of 0.89 cm for electrons, positrons and photons and thus in a compact calorimeter.

With its high energy resolution, the ECAL was especially designed to measure the mass in the benchmark decay of the Higgs boson into two photons with a very large accuracy.

Hadronic Calorimeter (HCAL)

Hadrons traverse the electromagnetic calorimeter largely without hadronic interactions. They are then stopped in the hadronic calorimeter where they interact with the nuclei of typically heavy material, and produce hadronic showers of pions, kaons, nucleons and fragments of nuclei.

The hadronic calorimeter consists of 50 mm thick copper absorbers interleaved with 4 mm thick scintillator tiles, which are read out with wavelength shifting fibres.

To provide a good resolution for missing energy, there has to be a calorimeter coverage up to a pseudo-rapidity of $|\eta| = 5$. Therefore, Very Forward Calorimeters (VCALs) are installed 6 m downstream of the endcap calorimeters. They also provide a means for tagging very forward jets. The VCALs are designed similar to the HCAL except that in these calorimeters quartz fibers are used as active medium embedded in a copper absorber matrix.

1.2.3 Muon System

The muon system is the outermost part of the detector, located outside the solenoid. It consists of the iron return yoke interleaved with four layers of detectors for triggering and position measurement. Muons arriving here have already deposited about 3.5 GeV in the inner layers, whereas most other particles (besides weakly interacting particles like muons or neutrinos) have already been absorbed in the calorimeters.

Resistive Plate Chambers (RPC) are used for triggering purposes in both, the barrel and endcaps, since they have an excellent time resolution, covering a pseudo-rapidity range of $|\eta| \leq 2.1$. In the barrel region, Drift Tubes (DT) provide the position measurements, whereas in the endcap regions Cathode Strip Chambers (CSCs) are used for this task, covering a pseudo-rapidity range of $|\eta| \leq 1.3$ and $1.3 \leq |\eta| \leq 2.4$, respectively. A resolution of 150 μm in z-direction and 100 μm in r and Φ directions can be achieved for the barrel region.

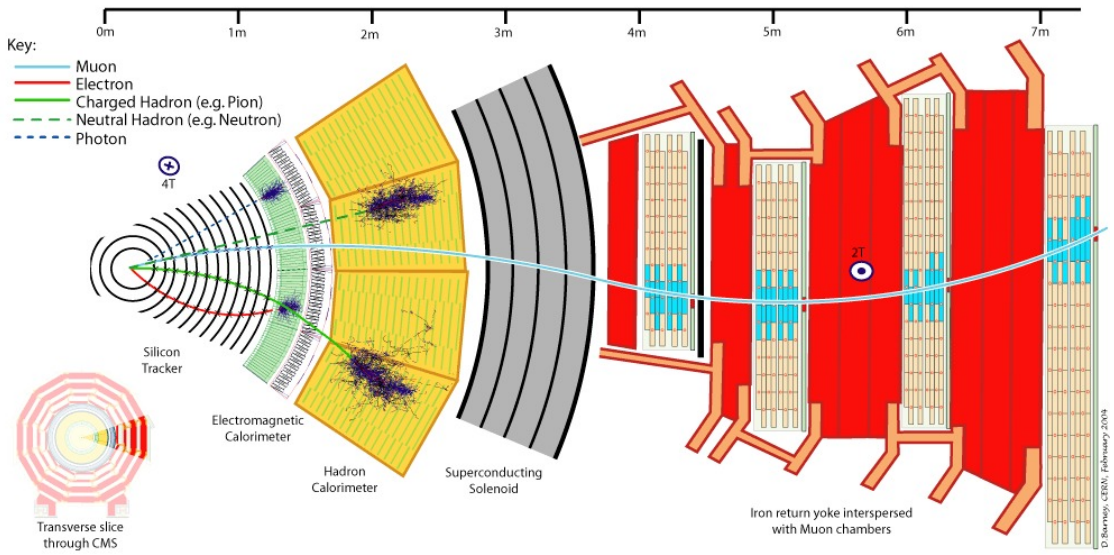

Figure 1.4: *Profile of the CMS detector with various particle tracks: charged particles have a curved trajectory whereas neutral particles move in a straight line; electrons, positrons and photons deposit their energy in the ECAL, hadrons in the HCAL and muons are detected in the muon system [11].*

1.2.4 Trigger and Data Acquisition

At the LHC, the proton bunches will cross each other with a rate of 40 Mhz. Each crossing will yield about 17.5 proton-proton collisions, resulting in $7 \cdot 10^8$ interactions per second. Every bunch crossing generates roughly 1.5 MB of data, leading to 60 TB per second. This enormous data rate has to be reduced to feasible numbers for the mass storage systems and offline computing facilities. For this purpose a 3-level trigger system was devised, the CMS Trigger and Data Acquisition System (TriDAS). It is designed to select events at a maximum rate of $\mathcal{O}(10^2)$ Hz.

At first, the level-1 trigger reduces the event rate by a factor of 400 to about 100 kHz, deciding for each event in about 2 μs if it is interesting or not. To achieve this task, it is completely implemented in hardware and only uses the data from the muon systems' RPCs and from the coarsely segmented calorimeters. Meanwhile, the full event data remains in the memory buffers.

If the event gets accepted it is then passed to the High Level Trigger (HLT), which is deployed as an online processing farm, consisting of fully programmable processors. The HLT can be divided into a level-2 trigger and a level-3 trigger. The former trigger is using the full calorimeter and muon system information and the latter is also utilising tracker information for triggering purposes. Together they decide in roughly 1 s if the event is to be stored and later used for analysis or discarded, resulting in a final event rate of $\mathcal{O}(10^2)$ Hz.

An overview of the data flux and its reduction by the trigger from the CMS detector to the mass storage systems can be seen in figure 1.5.

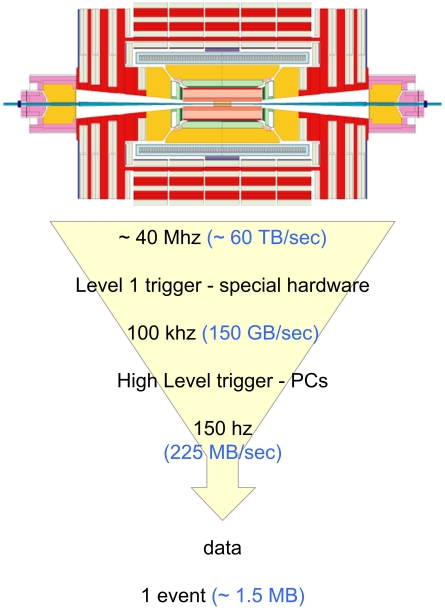

Figure 1.5: *Overview of the CMS Trigger and Data Acquisition System. The 60 TB/s of raw data is filtered by the Level-1 trigger to a rate of 150 GB/s. The High Lever Trigger system then reduces the data further to about 225 MB/s, which is feasible for the storage systems [11].*

Physics of the Higgs Boson

Once the Large Hadron Collider will commence its work, the continuous search for the Higgs boson will reach a new climax, as it is expected that the LHC will either verify or disprove its existence. This thesis takes part in the search for this last unseen particle of the Standard Model of Particle Physics (SM). Therefore, an introduction to the Standard Model, describing the Higgs boson, and the theory of the Higgs boson itself will be given here. Subsequently, the possibilities of Higgs production at the LHC and its decay are described, in particular the challenges of the $t\bar{t}H$ channel, studied in this thesis, are elaborated.

2.1 The Standard Model of Particle Physics

The Standard Model of Particle Physics is the currently accepted theory that describes all matter along with three of its four interactions. It further predicts the existence of a new and yet undiscovered particle, the Higgs boson, which is discussed later. According to the SM, matter can be divided into two classes, fermions, half-integer spin particles, following the Fermi-Dirac distribution and bosons, integer-spin particles, following the Bose-Einstein distribution. The fundamental spin-$\frac{1}{2}$ fermions can be further divided into two families of quarks and leptons, with three generations each, according to their quantum numbers (see table 2.1).

The mass eigenstates of the different quark flavours have turned out not to be the flavour-eigenstates participating in the weak interaction introduced below[1]. Therefore, certain quantum numbers descriptive for a quark generation are subject to change under the weak interaction. Thus, the observable mass eigenstates of the quarks can be transformed into each other.

Baryons are massive particles consisting of two (mesons) or three quarks (hadrons)[2]. The former is composed of a quark and an anti-quark and the latter consists of three quarks or anti-quarks. Each baryon carries an integer electrical charge and a white colour charge. Colour charge is defined in analogy to the colour theory. It comes in three basic flavours: red, green and blue as well as the respective anti-colours. White is the product of a colour and its associated anti-colour or all three colours or anti-colours combined. Colour charge was first introduced as a quantum number to fulfil the Pauli principle for hadrons with three otherwise identical quarks, which have actually been observed.

[1] The eigenstates can be transformed into each other by the Cabibbo-Kobayashi-Maskawa (CKM) matrix.

[2] There are also proposed more exotic compositions like Pentaquarks, consisting of five quarks.

	generation			electrical	colour	interaction			
	1^{st}	2^{nd}	3^{rd}	charge	charge	strong	em	weak	grav.
quarks	u	c	t	2/3	r, g, b, $\bar{r}$, $\bar{g}$, $\bar{b}$	x	x	x	x
	d	s	b	-1/3	r, g, b, $\bar{r}$, $\bar{g}$, $\bar{b}$	x	x	x	x
leptons	e	μ	τ	-1	-	-	x	x	x
	ν_e	ν_μ	ν_τ	0	-	-	-	x	x

Table 2.1: *The two families of fundamental spin 1/2 fermions, divided into three generations, along with their charges and interactions. In the SM, electrically neutral neutrinos are massless and for each fermion there exists an anti-fermion with inverted quantum numbers but otherwise identical properties.*

Each fermion is subject to the weak interaction, fermions with an electrical charge also experience the electro-magnetic interaction and colour charged fermions are influenced by the strong interaction, too. The fourth fundamental interaction, gravity is not covered by the SM, but due to its relative weakness (a factor of 10^{-40} compared to the strong interaction), it generally can be ignored in theoretical studies.

These interactions are mediated by gauge bosons listed in table 2.2. The gauge bosons of the weak interaction ($W^{\pm}$ and Z^0) are massive, whereas all the other bosons (gluons, gravitons and photons) are massless. Gluons carry a colour charge and are therefore themselves subject to the strong interaction they are mediating. Photons on the other hand don't carry an electrical charge and are thus unaffected by the electro-magnetic interaction. Furthermore, $W^{\pm}$ and Z^0 bosons interact weakly with each other and the electrically charged weak gauge bosons also interact with the photon.

Further information about the Standard Model of Particle Physics can be found in [12, 13].

interaction	gauge boson	electric charge	colour charge	relative strength
strong	gluon g	0	r, g, b, $\bar{r}$, $\bar{g}$, $\bar{b}$	1
electromagnetic	photon γ	0	-	1/137
weak	$W^{\pm}$	± 1	-	10^{-14}
	Z^0	0	-	
gravitation	graviton G	0	-	10^{-40}

Table 2.2: *The three interactions described by the SM along with their spin 1 gauge bosons as well as the gravitation with its yet unseen spin 2 mediator. In addition, the charges of the gauge bosons and the relative strength of the interactions is shown.*

2.2 Theory of the Higgs Boson

2.2.1 Higgs Boson

Around 1964, the Higgs boson was first proposed by the English physicist Peter Higgs. It is a massive scalar elementary particle, predicted to exist within the Standard Model, but until now undetected. The Higgs boson is associated with a field. This Higgs field consists of two neutral and two charged component fields – the Higgs boson is the quantum of one of the neutral fields.

The Higgs field permeates every place in the universe and has a non-zero vacuum expectation value of 246 GeV/c^2. This property gives mass to every elementary particle. In particular, the non-zero vacuum expectation value spontaneously breaks the electroweak gauge symmetry, a phenomenon known as the Higgs mechanism described below. This is the only known mechanism capable of giving mass to the gauge bosons that is also compatible with gauge theories.

The Higgs boson's mass is not predicted by the Standard Model, in fact it is rather one of its free parameters. Right now there exists an experimental lower bound for the Higgs boson's mass of 114.4 GeV/c^2 at 2σ confidence, obtained at the LEP [14].

2.2.2 Higgs Mechanism

The Higgs mechanism explains how elementary particles obtain their mass. This is done by the coupling of a gauge boson to a scalar field. To understand this, an exemplary model for spontaneous symmetry-breaking is explained first. For this purpose a complex scalar field H, representing the Higgs field, is introduced. It is then associated with a non-negative potential energy of the form

$$V = (|H|^2 - v^2)^2 \tag{2.1}$$

having continuous minima at $|H|^2 = v^2$ (fig. 2.1) [15].

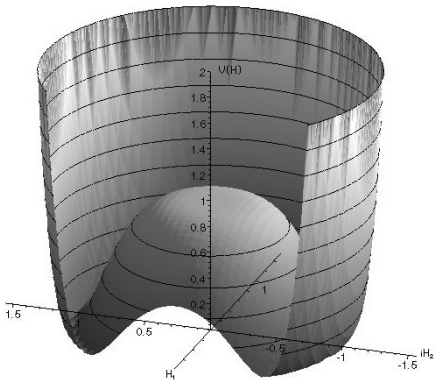

Figure 2.1: *Example for a Higgs potential like function:* $V(H) = (|H|^2 - 1)^2$.

The point $H = 0$ is symmetric with respect to the U(1) symmetry, that changes the complex phase of H. But this point is energetically unstable. The Higgs field will eventually settle in a position of minimal potential energy at $H = v \cdot e^{i\phi}$ with arbitrary ϕ phase. This induces an asymmetry on the vacuum, in the sense that the ground state is not invariant under the U(1) symmetry.

The spontaneous symmetry-breaking model predicts a massless scalar particle, which is the quantum excitation along the direction of ϕ, the so-called Nambu-Goldstone boson. There is no potential energy cost to move around the bottom of the circular valley, so the energy of such a particle is pure kinetic energy, which in quantum field theory implies that its mass is zero. But no massless scalar particles have been detected yet.

A similar problem in the non-Abelian gauge theory was the existence of massless gauge bosons, which, apart from the photon, also have not been detected. Higgs combined the gauge theory with a spontaneous symmetry-breaking model solving both problems: when coupling the scalar field to the gauge theory, the massless ϕ mode of the Higgs combines with the gauge boson to form a massive gauge boson.

In contrast to the U(1) symmetric model explained above, the weak interaction in the SM requires a SU(2) symmetry. This leads to a two-dimensional complex Higgs field. Two of its constituting fields are charged and become the $W\pm$ gauge bosons with the approach described above. One of the neutral fields is then identified as the Z^0 boson and the remaining neutral field becomes the Higgs boson. This theory is called the "Weinberg-Salam-Model" [16]. It is the mechanism that gives mass to the gauge bosons of the weak interaction and to the Higgs boson itself.

The Yukawa mechanism defines the couplings of the Higgs field to the fermions which provides mass to all quarks and leptons in the Standard Model. The coupling constants of this interaction are linearly related to the fermion masses but not provided by theory itself and thus have to be determined experimentally.

The Higgs mechanism was first incorporated into modern particle physics by Steven Weinberg and is now an essential part of the Standard Model. Its experimental discovery or falsification on the other hand is still due and will probably happen in the coming years at the LHC.

2.3 The Search for the Higgs Boson

The foremost goal physicists hope to accomplish at the LHC is to detect the Higgs boson and verify the theory of electroweak symmetry breaking. To manage this task it is essential to know what one is looking for. Therefore, the most important Higgs production processes at the LHC and its decay processes are explained in this thesis, with special focus on the $t\bar{t}H$ channel and its associated background channels.

The SM predicts that the couplings of the Higgs boson are proportional to the masses of the involved particles. This fact is of great relevance for all Higgs production and decay processes.

2.3.1 Higgs Production at the LHC

Hadron colliders provide several ways to produce Higgs bosons. The four most important ones are presented below and an overview of the cross-sections is shown in figure 2.2.

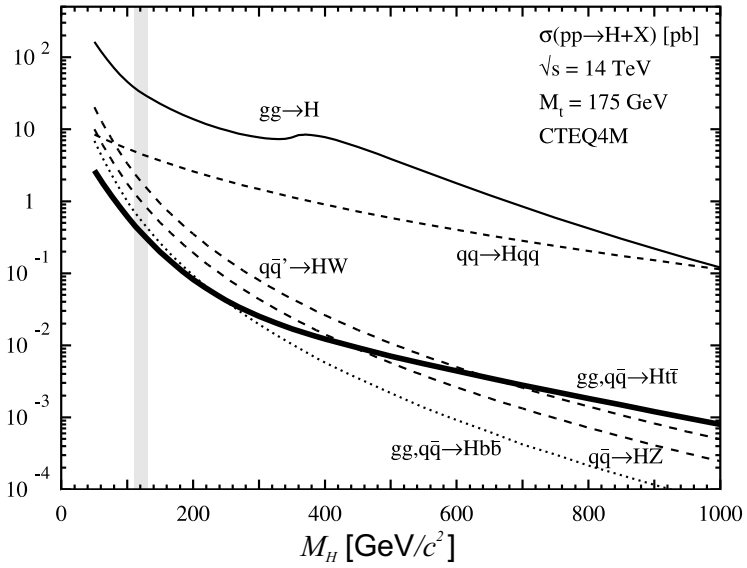

Figure 2.2: *Cross-sections of the main Higgs production processes at the LHC according to the Standard Model. The thick line marks the process of associated Higgs production with* $t\bar{t}$*, analysed in this thesis; the grey area highlights the studied Higgs mass of 120* GeV/c^2 *[17].*

Gluon-Gluon Fusion

Gluon-gluon fusion is the most dominant Higgs production process over the whole Higgs mass range (fig. 2.3). It is mediated by a quark loop (most often by a top quark loop due to its high mass) since the colour charged gluons cannot couple directly to the colourless Higgs boson and because the gluons are massless. This process is suppressed due to the heavy quark loop but this is compensated by the high gluon luminosity of the LHC.

Searches in this channel may suffer from the presence of many background processes. So, only few final states can be used effectively like $H \rightarrow \gamma\gamma$ or $H \rightarrow 4l$.

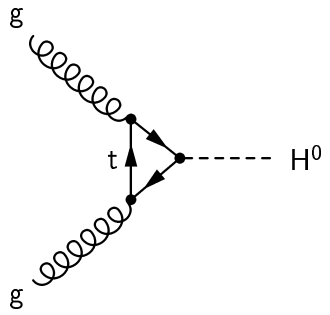

Figure 2.3: *Leading order Feynman graph for Higgs production via gluon-gluon fusion, mediated through a top quark loop. This is the dominant production process over the whole Higgs mass range.*

Weak Gauge Boson Fusion

The fusion of two Z^0 or $W^{\pm}$ bosons is the second largest Higgs production process at the LHC (fig. 2.4). Two quarks each radiate a weak gauge boson that fuse together to form a Higgs boson with the final state quarks leaving in forward and backward directions. In the case of Z^0 fusion quark flavours are conserved, whereas with $W^{\pm}$ bosons quark flavours change.
After the initial detection of the Higgs boson, this process can be further exploited to measure charge and parity (CP) properties and properties of the Higgs coupling.

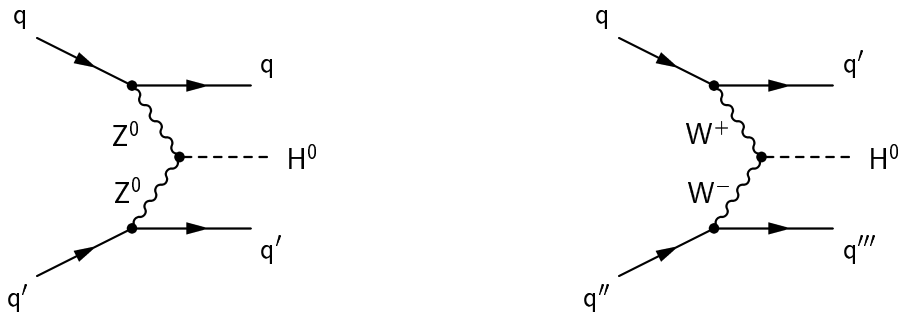

Figure 2.4: *Higgs production via a weak gauge boson fusion process at leading order. Fusion processes with charged bosons change the involved quarks, whereas uncharged boson fusions leave the quark flavours unchanged.*

Higgs Radiation

The next frequent Higgs production process is an associated process where a weak gauge boson is always produced together with the Higgs boson (fig. 2.5). Two quarks annihilate to an off-shell Z^0 or $W^{\pm}$ that then radiates the Higgs boson. For the Z^0 production a quark and its anti-quark

is needed which results in a suppression of this process, which would otherwise be higher than the Higgs production through a $W^{\pm}$ boson (because of the higher Z^0 mass).
This channel can easily be distinguished from its background channels through the bosons' decay particles and furthermore it is for measurements of the corresponding Yukawa couplings.

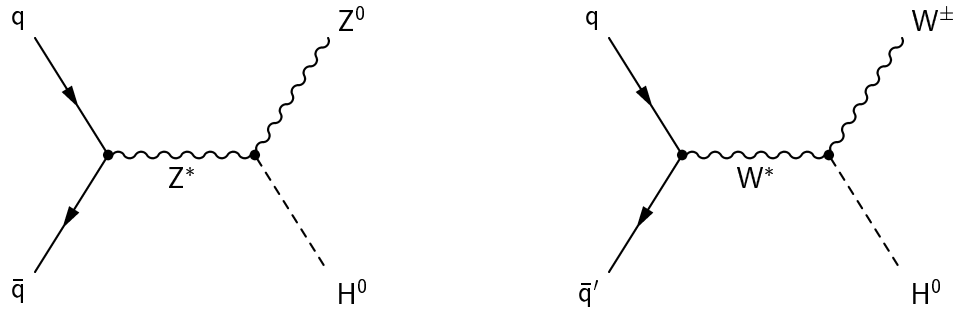

Figure 2.5: *Higgs radiation process for Higgs production. The Higgs boson is radiated by a weak gauge boson that was produced off-shell.*

Associated Higgs Production with a Top Quark Pair

The Higgs production in association with a top and anti-top can mainly occur via two processes. In the first process two gluons decay into $t\bar{t}$ pairs and a top from one decay tree together with an anti-top from the other decay tree then annihilate in the Higgs boson. In the second process, a gluon that was produced through quark-quark or gluon-gluon annihilation decays into a top and an anti-top and one of these then radiates the Higgs boson (fig. 2.6).
This channel has a rather small cross-section but is nonetheless very attractive due to its remarkable signature, especially in the lower Higgs mass ranges. Here the Higgs boson predominantly decays into a $b\bar{b}$ pair which together with the two W bosons and b quarks from the top decays shows a high tagging potential .

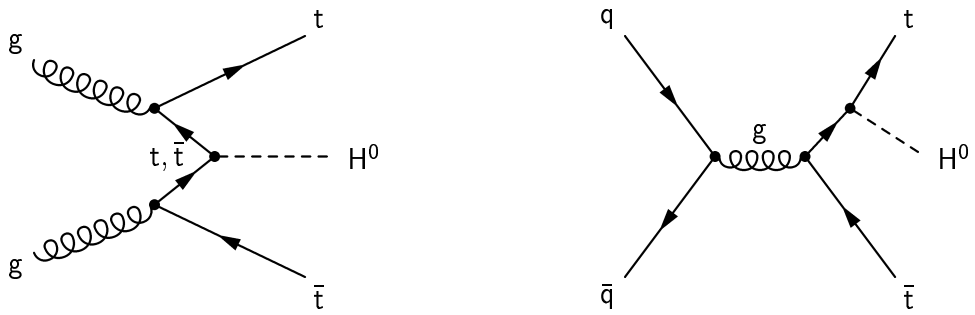

Figure 2.6: *Leading order Feynman graphs for the Higgs production with an associated top quark pair in the final state. On the left, a top-top fusion process is shown and on the right-hand side a gluon decay is displayed.*

2.3.2 Higgs Decay Modes

The Higgs decay mode is highly dependent on the Higgs mass. As already mentioned, it preferably decays into the heaviest particles kinematically possible. Therefore, different analysis approaches have to be used, depending on the assumed Higgs mass to be studied. The branching ratios of the various decay modes are shown in figure 2.7.

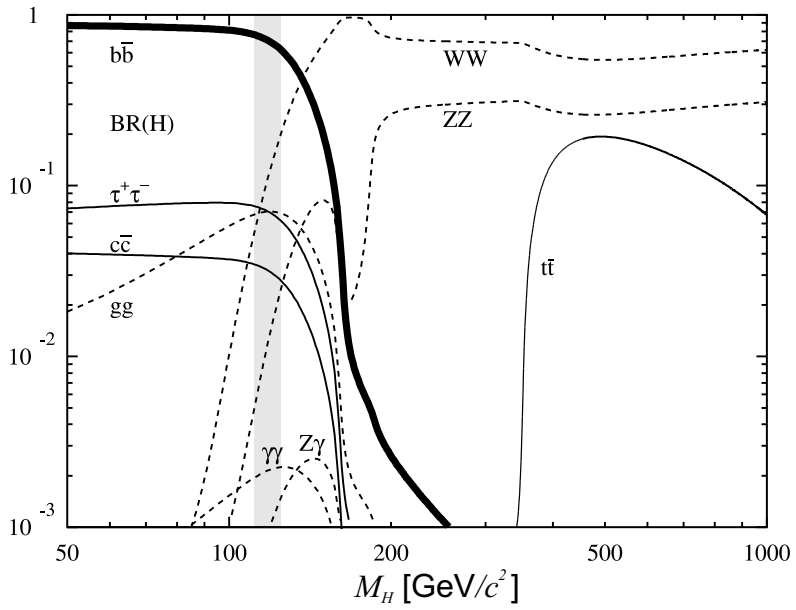

Figure 2.7: *Branching ratios of the main decay channels of the Higgs boson in the Standard Model. The thick line marks the decay process* $H \rightarrow b\bar{b}$ *analysed in this thesis; the grey area highlights the studied Higgs mass of* 120 GeV/c^2 *[18].*

Higgs Decay into Massive Particles

For a Higgs mass below about 150 GeV/c^2, decays into two fermions will dominate, with $H \rightarrow b\bar{b}$ being the most probable one due to the high b mass (fig. 2.8). For higher Higgs masses, decays into two $W^{\pm}$ or two Z^0 bosons become possible. At first, one boson will be produced off-shell[3] and, when the kinematical threshold is reached, on-shell as well . The $t\bar{t}$ final state is suppressed until the decay into two real top quarks becomes kinematically possible. But this process will remain less important for higher Higgs masses compared to both boson decay modes.

[3] An off-shell particle is a virtual particle with a mass far away from the actual resonance mass of the on-shell or real particle. It is only part of the quantum probability calculations in the Feynman diagrams and cannot be measured experimentally. Processes with off-shell particles have a smaller cross-section than the same process with an on-shell particle, but the latter might not always be kinematically possible.

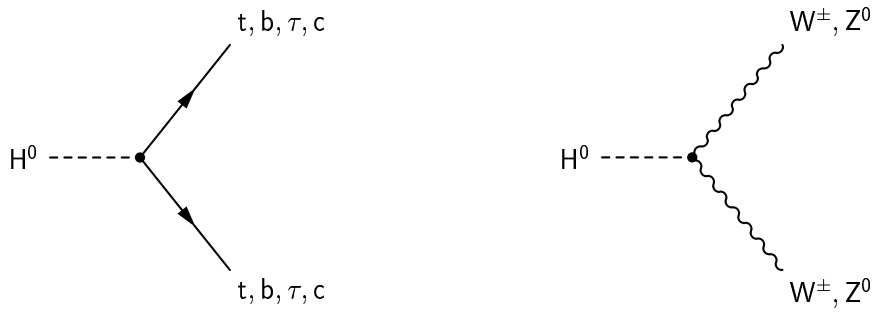

Figure 2.8: *Decay process of the Higgs boson into two massive leptons or bosons.*

Higgs Decay into Massless Particles

In addition, the Higgs boson can also decay into two massless gluons or photons (fig. 2.9). In the case of a gluon-gluon final state, this decay has to be mediated by a quark loop (predominantly with a top quark loop, due to its high mass), since the colourless Higgs boson cannot couple directly to the gluons that have a colour charge. Accordingly, the photon-photon final state is mediated by an electrically charged particle loop.

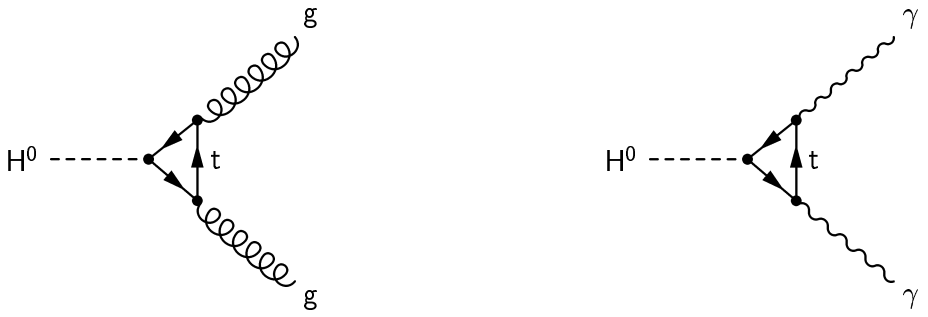

Figure 2.9: *Leading order Feynman graphs for the decay of the Higgs boson into massless gluons or photons mediated by a quark or charged particle loop, respectively.*

2.3.3 The $\mathrm{t\bar{t}H}$ Channel

The channels that are promising for an early discovery of the Higgs boson at the LHC are $H \rightarrow \gamma\gamma$ up to a Higgs mass of 140 GeV/c^2, $H \rightarrow 4l$ for Higgs masses between 130 and 700 GeV/c^2 and $H \rightarrow jjll$, $H \rightarrow ll\nu\nu$ or $H \rightarrow jjl\nu$ at Higgs masses of 0.5 to 1.0 TeV/c^2. In the mass range just above the current experimental limit of 114.4 GeV/c^2, the channel of associated Higgs production $t\bar{t}H$, as shown in figure 2.6, has also the potential to support the Higgs discovery. This channel suffers from a lower production rate compared to the other mentioned channels, but otherwise it features a clear signature due to its two top quarks, that can be detected easily.

The two top quarks produced in association with the Higgs boson almost exclusively decay into a W boson and a b quark. Each W boson then decays further, either hadronically into two light quarks or leptonically into a lepton and the associated neutrino. Depending on the decay mode of the W bosons, the final state of the $t\bar{t}H$ channel is categorised as hadronic ($WW \rightarrow 4j$), leptonic ($WW \rightarrow l\nu l'\nu'$) or semi-leptonic ($WW \rightarrow jjl\nu$). This thesis focuses on the semi-leptonic final state with its four b jets, two light quark jets, one muon (used for triggering) and one neutrino (fig. 2.10).

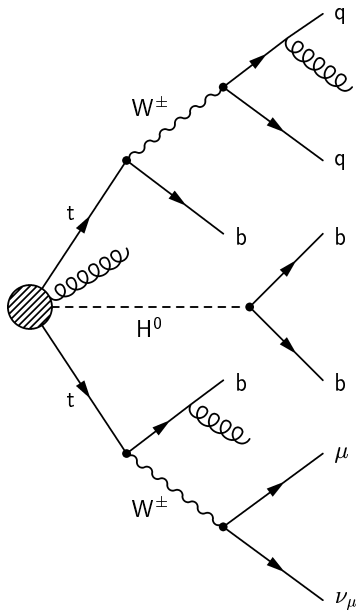

Figure 2.10: *Schematic view of the final state topology of the* $t\bar{t}H$ *channel analysed in this thesis.*

The semi-leptonic final state was chosen because it offered the best overall compromise of all three possible final states. It features a high branching ratio (see table 2.3), less obstructive background processes than the fully hadronic final-state and contains only a single lepton, simplifying triggering and missing energy calculations with regard to its associated neutrino.

Process	Branching Ratio
$W \rightarrow jj$	0.68
$W \rightarrow l\nu$	0.32
$WW \rightarrow 4j$	0.46
$WW \rightarrow jjl\nu$	0.44
$WW \rightarrow l\nu l'\nu'$	0.10

Table 2.3: *Selected branching ratios of the W and WW decay [14].*

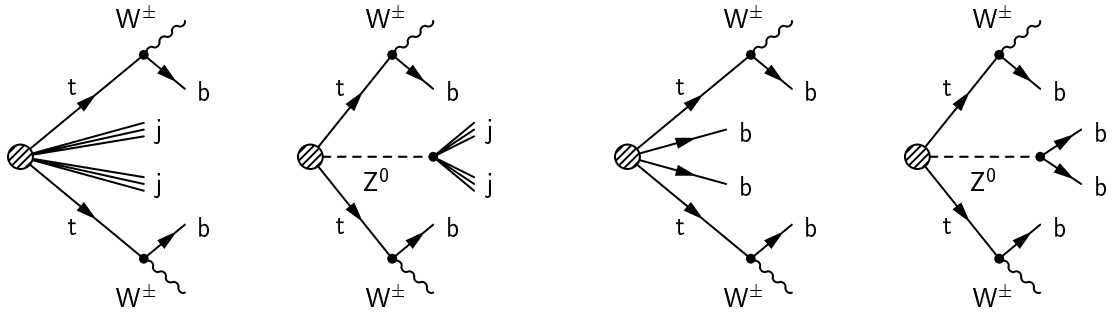

Figure 2.11: *Schematic view of the main background channels along with their sub-channels sporting b jets.*

The main background channels for the $t\bar{t}H$ channel will be $t\bar{t}jj$, where two top quarks and two quark or gluon jets are produced in the proton-proton collision and $t\bar{t}Z$ with $Z \rightarrow jj$, where a Z boson produced in association with the two tops decays into two quark jets. Both channels can be largely reduced through b-tagging algorithms[4], but their sub-channels with b jets as light quark jets remain irreducible (fig. 2.11). In table 2.4, the leading order cross-sections for these channels are listed, as well as the expected number of events after one year of data taking at low luminosity at the LHC. It is evident that the $t\bar{t}jj$ (reducible through b-tagging by a factor of 10^2 to 10^3 according to [19]) and the $t\bar{t}b\bar{b}$ (irreducible) channels will be the dominant background channels that have to be handled. The $t\bar{t}Z$ channel will only slightly contribute to the background due to its low production rate and the wide decay spectra of the Z boson.

Final state	**LO cross-section [pb]**	**number of events**
$t\bar{t}H$	0.577	11,534
$t\bar{t}Z$	0.646	12,920
$t\bar{t}b\bar{b}$	3.28	65,600
$t\bar{t}jj$	507	10,140,000

Table 2.4: *Leading order cross-sections for the $t\bar{t}H$ channel and its most prominent background channels along with the resulting events after one year of data taking at low luminosity (20 fb^{-1}) at the LHC [19].*

2.3.4 Analysis Challenges

The $t\bar{t}H$ channel features a complex topology with four b jets, two light quark jets, an isolated muon, missing energy and possible additional jets from Initial State Radiation (ISR) and Final State Radiation (FSR). This entails high requirements on all levels of the detection and reconstruction chain. Jet resolutions need to be good in order to allow a clear identification of the top quarks, the b-tagging efficiency has to be high to yield a sufficient number of signal events, and the mistagging-rate must be low, so that the immense $t\bar{t}jj$ background can be suppressed.

[4]The aim of b-tagging algorithms is to identify jets in an event containing a b or $\bar{b}$ quark.

The complex final state leads to non-trivial ambiguities in the assignment of the observed detector objects to the original partons of the process that have to be resolved during the analysis. At least, 24 different combinations are possible (assuming perfect jet identification): there are 12 ways to distribute the four b jets to the two top quarks and the Higgs ((6-2)!/2, since the two b jets from Higgs decay as well as the two light quarks are indistinguishable). In addition, there are almost always two different solutions for the longitudinal component of the neutrino's momentum that has to be taken into account[5]. With finite jet resolutions, imperfect b-tagging and the presence of additional jets, the complexity of the final state increases and makes it even more difficult to correctly identify the final state particles.

The $t\bar{t}b\bar{b}$ background channel features the same distribution of final state particles with very similar kinematics as the $t\bar{t}H$ channel, making the identification of a background event difficult. Small differences in the available observables have to be detected and exploited extensively to separate these two channels and to get a clearer view on the Higgs boson.

Another problem arising in the detector is the pile-up of different events. Because of the high proton density in each bunch several collisions can happen per bunch crossing and due to the high bunch cross rate an event from a previous bunch crossing can still be in the detector, when the next bunch crossing happens. This leads to the detection of additional jets which are not coming from the currently triggered event and to the measurement of wrong values as several jets deposit their energy in the same detector element. Sometimes the effect of multiple collisions in one bunch crossing is also referred to as minimum bias. The term pile-up then only applies to collisions in consecutive bunch crossings happening in too quick succession.

[5] In proton-proton collisions, only the transverse component of the neutrino's momentum can be determined accurately by a missing energy measurement; a W mass constraint can then be used to obtain the (generally two) analytical solutions for the longitudinal component.

Neural Networks

A common task in high energy physics analyses is to separate experimental data into certain classes. For example interesting events (signal) have to be divided from unwanted events (background). One way to do this is applying cuts[1] on certain variables of the events. This usually yields acceptable results, but with an increased number of cuts, the number of signal events that do not pass the cut increases, too. Furthermore additional information inherent to the correlations between variables is also wasted. To handle this problem and to improve the overall separation, the whole information inherent to an event can be transferred into one single variable. This variable is then used to classify an event as signal or background. Methods for the necessary transformation step are for example likelihood methods, or more sophisticated, artificial neural networks (ANNs).

The human brain is able to solve complex classification functions with ease. This extraordinary ability has been widely researched by scientists of various fields. In the last decades great efforts were taken to find good models for this design of nature and to adapt it for technical usage. The results of these researches are a variety of artificial neural network models. One expects to built systems based on these models which are able to learn certain facts guided by a teacher or on their own. This accumulated knowledge is then to be applied in different situations, generalising from the learned facts and coming to correct conclusions even with incomplete or faulty information. In addition such systems provide an ideal playground for parallel processing models.

It is important to note that artificial neural networks represent a whole family of models, not just a single technique. Each form is optimised for a specific task, analogous to the functional specifity associated with different regions of the human brain. This thesis is based on the application of artificial neural networks. Therefore an extensive overview of this topic is given in the following sections, focusing on feed-forward networks, since the neural network packages used for this thesis are based on this model. In addition a short overview of other neural network models is given in the last section.

Further information about neural networks can be found in [20] - [23].

[1] The act of rejecting an event if the value of a specific variable of the event does not meet a certain threshold is called, applying a cut on this variable.

3.1 Fundamentals

3.1.1 The Biological Model

The research in the field of artificial neural networks has been greatly inspired and influenced by our knowledge of biological nervous systems. Their basic component is the neuron. This nerve cell receives electro-chemical stimuli from other neurons and, in response, generates electrical signals of its own that are transmitted to other nerve cells. 10^{10} to 10^{12} of these cells form a network, called the brain.

A neuron is highly asymmetric in shape (fig. 3.1). Its central part is the soma or cellular body containing the nucleus and providing the link between the cellular extensions called dendrites and axons. Each nerve cell has got many dendrites that are the main information sources of the neuron. In contrast it has got only a single extension, called the axon, to carry away nerve signals to other neurons. It grows extremely long and undergoes extensive branching to reach as many other nerve cells as possible. At the end of each branch there is a small swelling that connects the axon to a dendrite of a different neuron. This swelling is called a synapse. Thus a highly interconnected network of nerve cells is formed.

Neural activity is defined by an internal electrical potential which is influenced by the activities of the nerve cells connected through the dendrites. If this potential reaches a certain threshold, the neuron "fires" and a series of electrical stimuli are sent through the axon to the synapses. Here, neuro-transmitters, a kind of chemical messengers, are poured out and exhibit or inhibit the following nerve cell. This information transfer between the neurons is mainly regulated at the synapses. These are able to grow and shrink; new synapses can form and old ones can disappear. It is assumed that this behaviour is the actual basis for the learning process of our brain.

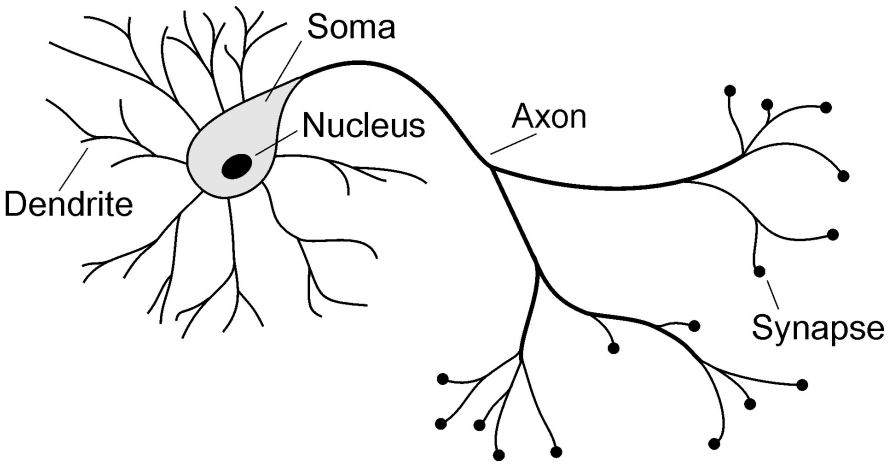

Figure 3.1: *Schematic view of a biological nerve cell.*

3.1.2 The Artificial Neuron

In accordance to the biological neuron, an artificial neuron is generally devised as shown in (fig. 3.2). This was first realised by McCulloch and Pitts in 1943 [24]. A more biological approach to artificial neurons can be found for example in [25, 26].

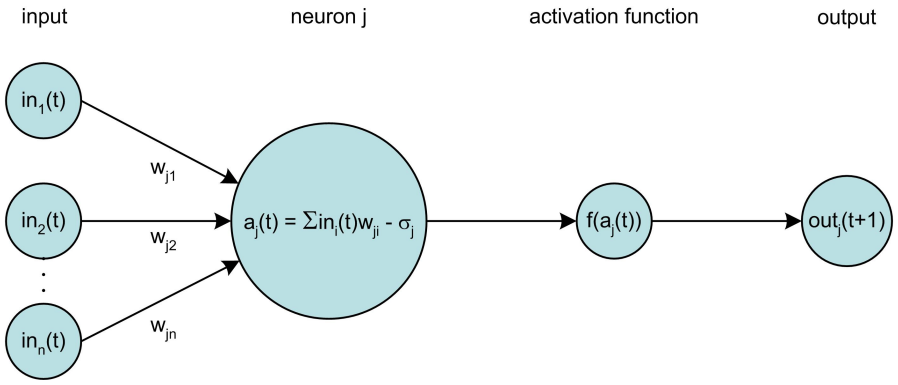

Figure 3.2: *Schematic view of an artificial neuron.*

A neuron j has multiple inputs $in_{1..n}(t)$ coming from other neurons. These values get summed up to $a_j(t)$ in the neuron according to the weights w_{ji} of the inputs. The final neural output $out_j(t+1)$ of the neuron j is obtained by applying an activation function $f(a_j(t))$ to the summed inputs a_j. McCulloch's and Pitts' model used binary values for the input variables and a Heaviside activation function.

The final equation modelling an artificial neuron is as follows:

$$out_j(t+1) = f\left(\sum_{i=1}^{n} in_i(t)w_{ij} - \sigma_j\right) \tag{3.1}$$

The term σ_j is called bias. It is used to control the threshold of the neuron. This gets more vivid, when using the Heaviside-function $\Theta(x)$ as an example for the activation function. Here the bias directly controls at which sum the neural output will switch from 0 to 1, i.e. when the neuron will "fire". The actual value of the bias can also be learned by the neural network, too. Therefore each neuron j gets an additional input with a fixed input value of -1 and a weight σ_j equal to the value of the bias.

Today, current neural network approaches utilise a continuous activation function that is differentiable and maps $(-\infty, \infty) \to [0, 1]$. For this purpose, the sigmoid function

$$s(x) = \frac{1}{1 + e^{-cx}} \tag{3.2}$$

is often used, but other functions like tanh(x) are also employed. This is the primary feature of modern neural networks. The described activation function enables the neural network to learn non-linear relations between different input variables.

3.1.3 Training Methods

The training method is the most important part of a neural network. It defines how a network will learn and what it will be capable of later. Training methods can be classified into two principal categories: supervised and self-organised learning. They will be explained below.

Supervised Learning

Supervised learning methods require a teacher that knows what the target outputs of the network should be for a given set of training input patterns. The training then follows simple steps, which are done for every input pattern of the training and repeated as often as needed:

1. An input pattern is presented to the network.
2. The network output is calculated.
3. The network weights are adapted according to the difference between the calculated output and the target output.

Adjustments of the weights can happen at several occasions. One usually distinguishes between the following two options:

- **Batch Mode:**
  All training patterns are presented to the network before the weights are adjusted. This is useful to take advantage of the full training sample before altering the network. It gets applied to achieve a better adaptation of the neural network to the training patterns.
- **Online Learning:**
  The network weights are already adjusted after several training patterns have been presented to the network. This is usually done if the number of training patterns is too large to keep all the data quickly accessible or if the speed of the training is important.

The neural network models discussed in this chapter all require supervised learning and use the online learning method, if not mentioned otherwise.

Self-Organised Learning

In contrast to the supervised learning method, self-organised learning or unsupervised learning does not need a teacher. It tries to cluster similar input patterns and to maximise the distance in output space between diverse input patterns. This learning method is usually applied when there is no a-priori knowledge of the network output or if it would be too costly to generate. Prominent representatives of this method are Kohonen networks, explained in section 3.5.2.

3.2 Feed-Forward Networks

3.2.1 Topology

Neural network topologies exist in great quantities. The simplest structures are called feed-forward networks (fig. 3.3). They only feature information transport into one direction from the inputs to the outputs. Here, neurons are usually arranged in layers and connections only exist from one layer to the next. One distinguishes between the input layer, the output layer and possible hidden layers in-between. It is common convention not to include the input layer in the overall layer count. The inputs of the network are generally also referred to as neurons even though they play no active role in the information processing.

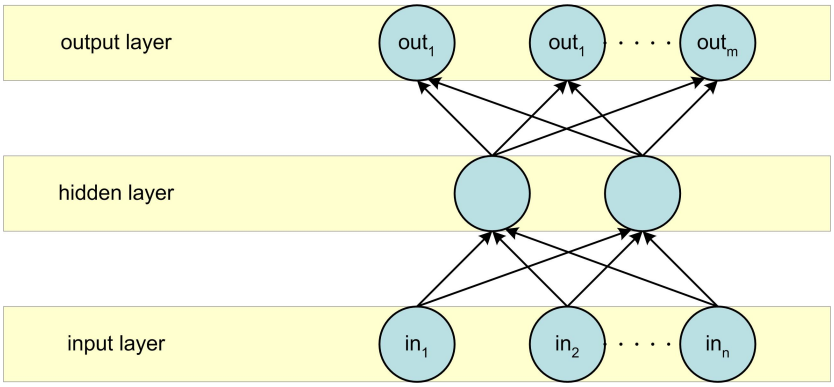

Figure 3.3: *Structure of a feed-forward network, including an input, one hidden and an output layer.*

Other topologies include recurrent and fully connected networks (fig. 3.4). A network is called recurrent if cross, auto and backward connections exist between the neurons. A fully connected network has got connections from each neuron to each other neuron.

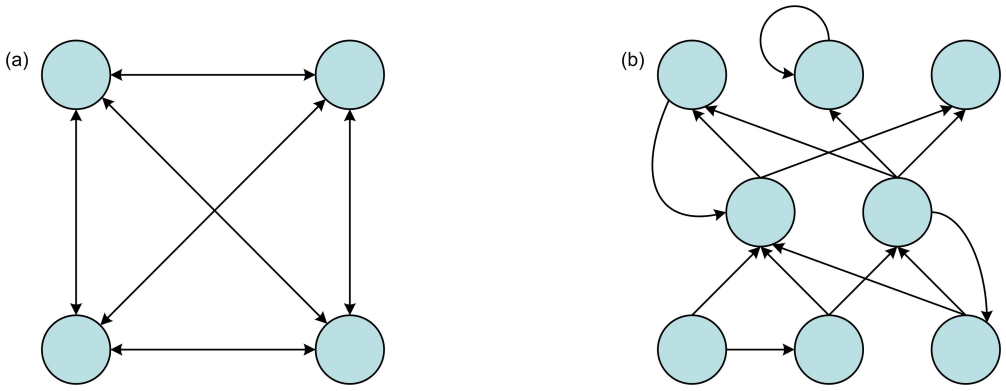

Figure 3.4: *Different network topologies. (a) fully connected, (b) recurrent.*

3.2.2 The Perceptron

The perceptron was first devised in 1957 by Frank Rosenblatt [27]. It is the simplest type of a feed-forward network, consisting only of an input and an output layer (fig. 3.5). Each input neuron is linked to each output neuron by a weighted connection.

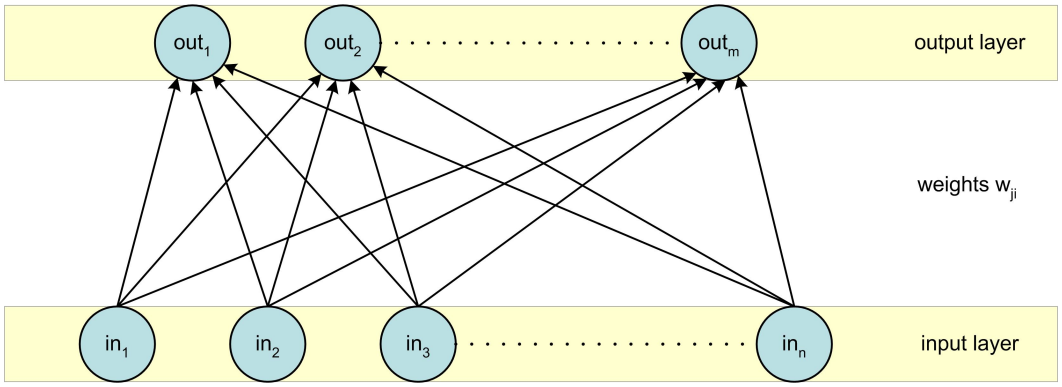

Figure 3.5: *A simple perceptron, consisting of an input layer with n inputs and an output layer with m outputs. Every input neuron is connected to every output neuron.*

Every output $\text{out}_{1..m}$ of the perceptron is an explicit function of the inputs $\mathbf{in} = (\text{in}_1, ..., \text{in}_n)^T$, that can be calculated straightforward after propagating the input values through the network:

$$\text{out}_j = f\left(\sum_{i=1}^{n+1} \text{in}_i \; w_{ji}\right) \qquad j = 1, ..., m \tag{3.3}$$

The bias mentioned above is included in the sum of formula 3.3 as an additional weight $w_{j(n+1)}$ for each neuron j with a fixed input value of $\text{in}_{n+1} = -1$.

Gradient Descent

The training of the perceptron is done with a supervised learning method. Therefore a set of N training samples comprised of input patterns $\mathbf{in}^q$ and associated target output patterns $\mathbf{out}^q$ is needed. These sample input patterns are presented to the network and the weights are adjusted according to the target and network output.

In order to measure the quality of a neural network training an error quantity $E(\mathbf{W})$ is defined as function of the network weights. The goal of a network training is to minimise this error function by altering the network weights, so that target and network outputs converge.

A commonly used error function is the sum-of-squares error, defined as:

$$E(\mathbf{W}) = \frac{1}{2}\sum_{q=1}^{N}\sum_{j=1}^{m}\left(out_j^q - true_j^q\right)^2 = \sum_{q=1}^{N} E^q(\mathbf{W}) \tag{3.4}$$

Here the squared differences between the target outputs $true_j^q$ and the calculated outputs out_j^q for each input pattern q get summed up. The factor $1/2$ is introduced to eliminate duplicate summands.

If $E(\mathbf{W})$ is differentiable with regard to the network weights w_{ji} various gradient-based techniques can be applied to obtain a minimum of $E(\mathbf{W})$. This will typically yield only a local minimum but certain training techniques and a careful choice of the starting values of the weights help to reach a local minimum not far from the global minimum.

The best understood of these numerical optimisation techniques is the gradient descent algorithm. It suggests to change each weight w_{ji} of the neural network in the opposite direction of the gradient of the error function $E(\mathbf{W})$ at the position of the current network weights $\mathbf{W}$:

$$\Delta w_{ji} = -\eta\frac{\partial E(\mathbf{W})}{\partial w_{ji}} = \sum_{q=1}^{N}\Delta w_{ji}^q \tag{3.5}$$

The parameter η is called learning rate and indicates how fast the network weights are changed. It should be chosen small in order not to miss a possible minimum. But it should also be chosen not too small for a faster training. Typically it is not a constant and gets reduced as the network training approaches a final minimum.

If the activation function f(x) is differentiable, equation 3.5 can be further rewritten as:

$$\Delta w_{ji}^q = -\eta\ \delta_j^q\ in_i^q \tag{3.6}$$

with:

$$\begin{aligned}\delta_j^q &= (out_j^q - true_j^q)\ f'(a_j^q)\\ a_j^q &= \sum_{i=1}^{n+1} in_i^q\ w_{ji}\end{aligned}$$

Equation 3.8 defines an update rule for the perceptron's weights that minimises the error function $E(\mathbf{W})$. It is referred to as delta rule [24]. Since it is only dependent on one input pattern and the output of one neuron, it is easy to implement. The calculations get particularly simple if the sigmoid function s(x) of equation 3.2 with parameter $c = 1$ is chosen as activation function, since $s'(x) = s(x)\cdot(1 - s(x))$.

Network Interpretation

Because of the simple structure of the perceptron, its functionality can be easily visualised. Each network output function out_j describes a hyperplane in input space, separating it into two halves. The actual position of that hyperplane – not its orientation – can be varied further by the bias. Thus input patterns can be separated into two classes. Figure 3.6 gives an example of a neural network simulating a logical AND-function.

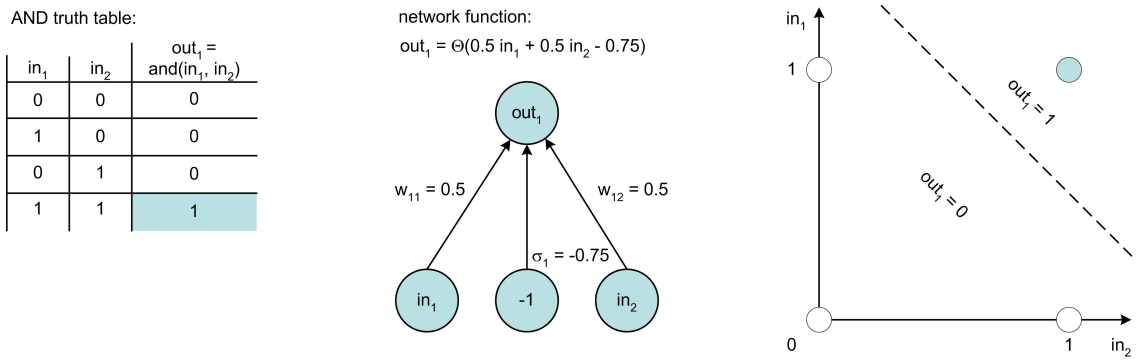

Figure 3.6: *Logical AND function. The truth table of the AND function is shown on the left. A neural network able to simulate this function is displayed in the middle using neurons according to McCulloch and Pitts. On the right, the input space* $\{in_1, in_2\}$ *is plotted with the associated outputs* out_i*. The dashed line symbolises the hyperplane the neural network introduces, separating the areas with* $out_1 = 1$ *from the areas with* $out_1 = 0$*.*

Limits of the Perceptron

As described in the previous section the perceptron is only capable of learning problems that are linear separable. A simple challenge as the logical XOR function (fig. 3.7) cannot be modelled by this approach. This deficiency led to a drastic decrease in neural network researches in the 1970s before more sophisticated models got developed.

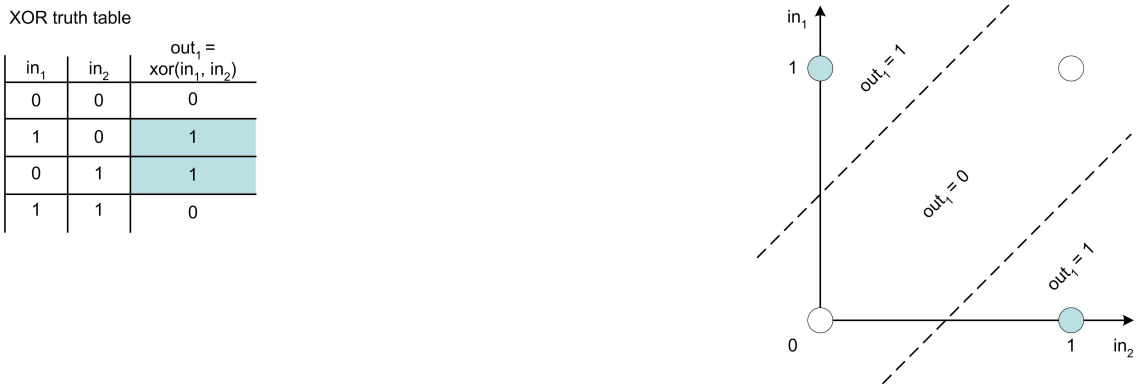

Figure 3.7: *Logical XOR function. The truth table of the XOR function is shown on the left. On the right the input space* $\{in_1, in_2\}$ *is plotted with the associated outputs* out_i*. There exists no single hyperplane that can separate the areas with* $out_1 = 1$ *from the areas with* $out_1 = 0$*.*

3.2.3 Multilayer Networks

In order to overcome the limitations of single layer neural networks like the perceptron this concept got expanded. A logical step is to introduce another layer of neurons, thus effectively linking several perceptrons together to form a multi-layer perceptron (MLP) that surpasses the classification capabilities of its components (fig. 3.8). Further layers can be added accordingly to the neural network. In each case the computations for each neuron stay the same, even though the output neurons are no longer direct functions of the input variables in_{i1}.

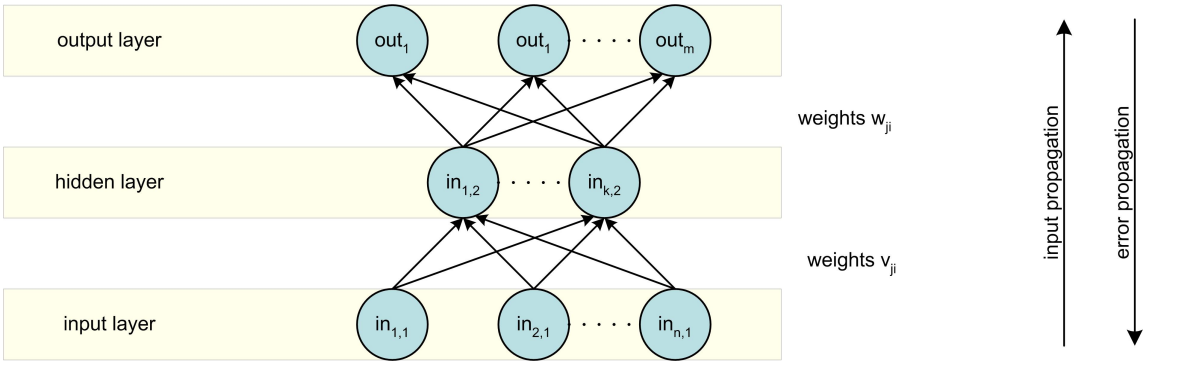

Figure 3.8: *A multi-layer perceptron with n inputs and m outputs, featuring one hidden layer with k neurons. Input patterns traverse the neural network from the bottom to the top and get processed thereby whereas errors are handed backwards for the training.*

The hidden layer of a MLP transforms an input pattern $\{in_{11}, ..., in_{n1}\}$ into another representation $\{in_{12}, ..., in_{k2}\}$ (not necessarily injective). Here the output layer can construct a hyperplane that separates the classes (fig. 3.9). Another interpretation of the functionality of a MLP takes advantage of the MLP's buildup of several perceptrons. It states that each perceptron forms one hyperplane in input space to separate the classes.

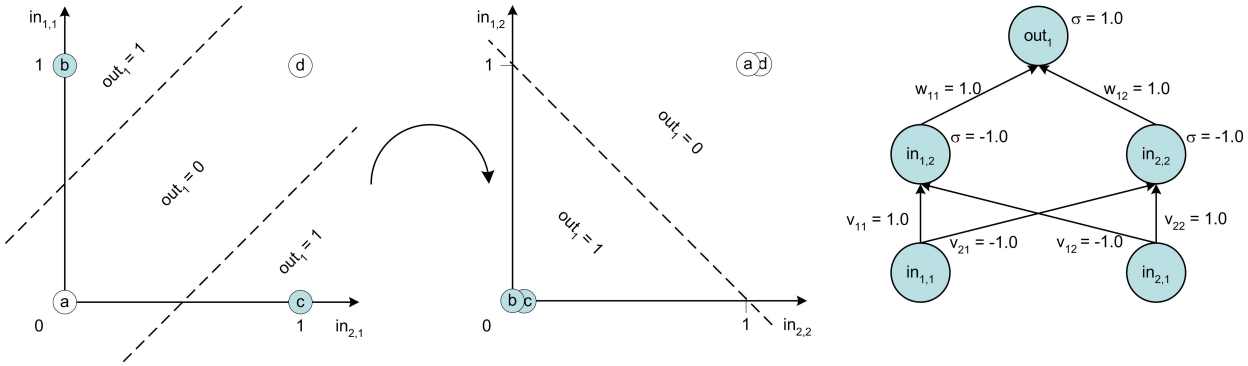

Figure 3.9: *Logical XOR function. On the left, the input space $\{in_{11}, in_{21}\}$ is shown along with the associated output regions. It gets transformed by the hidden layer into $\{in_{12}, in_{22}\}$ as depicted in the middle. Now a hyperplane can be found to separate the two classes. On the right, the responsible two-layer neural network is presented; the bias inputs are only shown implicitly.*

Backpropagation

Multi-layer neural networks cannot be trained by the simple delta-rule introduced for the perceptron, since the values of the target output of the hidden neurons are not known. If an network output deviates from its target value, it is impossible to tell which neuron in the hidden layer was responsible for this error and thus which weight should be changed.

An extension to the delta-rule, also based on the gradient descent method is the backpropagation algorithm. It was independently developed by several scientists, of whom Paul Werbos was the first in 1974 [28] but backpropagation has not become widely known until a decade later. This algorithm is divided into two steps:

- An input pattern is presented to the network and associated network output is computed
- The errors are backpropagated from the output layer to the input layer

The principal approach stays the same as in equation 3.5. Only now the derivative $\frac{\delta E(\mathbf{W})}{\delta w_{ji}}$ has to be calculated using the chain rule. This leads to the following results:

- for the output layer:

$$\Delta w_{ji}^q = -\eta \; \delta_{j2}^q \; in_{i2}^q \tag{3.7}$$

  with:

$$\begin{aligned} \delta_{j2}^q &= (out_j^q - true_j^q) \; f'(a_j^q) \\ a_j^q &= \sum_{i=1}^{k} in_{i2}^q \; w_{ji} \end{aligned}$$

- for the hidden layer:

$$\Delta v_{ji}^q = -\eta \; \delta_{j1}^q \; in_{i1}^q \tag{3.8}$$

  with:

$$\begin{aligned} \delta_{j1}^q &= f'(a_j^q) \cdot \sum_{i=1}^{m} w_{ij} \; \delta_{j1}^q \\ a_j^q &= \sum_{i=1}^{n} in_{i1}^q \; v_{ji} \end{aligned}$$

  where v_{ji} stand for weights from the input layer in_{i1} to the hidden layer in_{i2} and w_{ji} denote the weights from the hidden layer to the output layer out_i.

The actual weight update rule stays the same in both cases, only the δ-function is different for the output layer and the hidden layer. These equations can be expanded accordingly for additional hidden layers.

3.3 Training Optimisations

Simple training algorithms like the backpropagation algorithm introduced above deliver good training results. But there are many approaches to accelerate the training speed of a network and to improve the overall network quality after the training. Some of these optimisations are introduced in this section.

3.3.1 Regularisation

Oscillation of the error function $E[\mathbf{W}]$ around a minimum can be a problem in the training of the neural network. This can be suppressed by introducing a penalty term to the error function:

$$E(\mathbf{W}) \rightarrow E(\mathbf{W}) + \tau \cdot P(\mathbf{W}) \tag{3.9}$$

The penalty term $P(\mathbf{W})$ is controlled by the free parameter τ. A value of $\tau = 0$ means, that there will be no regularisation. Furthermore the penalty term can be used to enforce certain requirements on the neural network. One potential penalty term is defined as follows:

$$P(\mathbf{W}) = \sum_{i} \frac{w_i^2}{1 + w_i^2} \tag{3.10}$$

The neural network now gets punished for having large weights w_i. Thus minimising this error function leads to a network with overall small weights. Oscillations are also suppressed due to an altered form of the error function. This approach is called weight elimination [29].

3.3.2 Preprocessing

Convergence to a minimum in the error function $\Delta E(\mathbf{W})$ may be improved by preprocessing the input patterns before the training of the neural network. Some methods for preprocessing are listed below. For more information see [30].

- **mean cancellation:**
  The mean of all values in an input pattern should be evenly distributed around zero to avoid a bias and to improve the training speed. For example if all input patterns consist solely of positive values, the next weight update will happen in the same direction for every weight.
- **decorrelation:**
  With uncorrelated input variables the weights of the neural network can be minimised separately instead of the complicated process to minimise all weights at once.
- **covariance equalisation:**
  If the covariances of all input variables are in the same order of magnitude, convergence to a minimum may be reached faster.

3.3.3 Learning Rate

Momentum Term

The learning rate η is a crucial parameter for a quick and successful training. If the error function $E(\mathbf{W})$ features broad minima, η can be chosen large to speed up the training. However steep minima could be missed in this way. In general, if the weight update rule depends too strongly on the learning rate, oscillations around the minima of $E(\mathbf{W})$ can occur. To avoid this behaviour a momentum term is usually added to the weight update rule:

$$\Delta w_{ji}(t+1) = -\eta \frac{\partial E(t)}{\partial w_{ji}} + \alpha \cdot w_{ji}(t) \tag{3.11}$$

This helps to prevent large changes in w_{ji} and to keep the direction of the weight updates on track. The momentum term typically speeds up the convergence and leads to a more efficient training [31].

Adaptive Learning Rate

It is useful to have a high learning rate η at the beginning of the training, leading to quickly made improvements. After the training gets close to a minimum in the error function, η should be reduced to actually reach it. Such a learning rate is called adaptive learning rate.

The learning rate should be increased if $\Delta E(\mathbf{W})$ constantly decreased in the last N steps and vice-versa. If the changes in the error function did not show a clear tendency, the learning rate should be kept constant. This can be modelled by the following equation:

$$\eta(t+1) = \begin{cases} +a\eta(t) & \text{if } \Delta E(\mathbf{W}, n) < 0 \quad \forall\, n = t, ..., t-N+1 \\ -b\eta(t) & \text{if } \Delta E(\mathbf{W}, n) > 0 \quad \forall\, n = t, ..., t-N+1 \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

whereas $\Delta E(\mathbf{W}, n) = E(\mathbf{W}, n) - E(\mathbf{W}, n-1)$ denotes the change in the error function from step (n-1) to step n. The free parameters a, b and N in this equation have to be chosen appropriately.

There are more sophisticated methods that exploit the idea of altering the learning rate like Quickprop [32] or Rprop [33]. These are in general more theoretically founded and offer even better training results.

3.3.4 Other Error Functions

The sum-of-squares error function defined in equation 3.4 represents only one way of calculating the quality of a network. Different error functions can be used to focus the training on certain aspects.

For example different norms could be chosen instead of the Euclidean distance in the sum-of-squares error function (eqn. 3.13). For large values of R this would favour training patterns

with large errors. However individual runaway patterns could falsify the whole result.

$$E(\mathbf{W}) = \frac{1}{2}\sum_{q=1}^{N}\sum_{j=1}^{m}\left(out_j^q - true_j^q\right)^R \tag{3.13}$$

The relative entropy function (eqn. 3.14) forces the network to learn the hypothesis represented by the output out_j with a probability $\frac{1}{2}(1 + out_j^q)$ that this hypothesis is true [26]. The advantage of this function is that it rapidly diverges if one output saturates. In the same scenario, the sum-of-squares error would just stay constant for a long time.

$$E(\mathbf{W}) = \sum_{q=1}^{N}\sum_{j=1}^{m}\left(\frac{1}{2}(1 + true_j^q)\,\ln\frac{1 + true_j^q}{1 + out_j^q} + \frac{1}{2}(1 - true_j^q)\,\ln\frac{1 - true_j^q}{1 - out_j^q}\right) \tag{3.14}$$

Other error functions are defined differently at the beginning and at the end of the learning procedure (eqn. 3.15). To avoid early local minima, the error function is larger for those output values out_j that are not yet close to the desired target output $true_j$.

$$E(\mathbf{W}) = \sum_{q=1}^{N}\sum_{j=1}^{m}\begin{cases} \gamma(out_j^q - true_j^q)^2 & \text{if } sign(true_j^q) = sign(out_j^q) \\ (out_j^q - true_j^q)^2 & \text{if } sign(true_j^q) = -sign(out_j^q) \end{cases} \tag{3.15}$$

with $0 < \gamma < 1$ chosen appropriately.

3.3.5 Further Optimisations

Skipping of Training Patterns

Training patterns that are already trained well, i.e. that produce a small error function, do not need to be learned further. After the first step in the backpropagation algorithm $E(\mathbf{W})$ is checked against a threshold value c. If the error is smaller than c, the next training pattern is used, otherwise the error backpropagation is done.

Output representation

The actual representation of the output pattern can also complicate the training. For example, a neural network should learn to classify an alphabet A-Z. Now 26 output neurons are introduced, one for each letter. Output $out_j = 1$ indicates, this is letter j and $out_j = 0$ denotes, this is not letter j. The difference in the output pattern between the different classes is not very large (A = 100... vs. B = 010...), therefore they are difficult to be trained well. Either the output representation should be modified using fewer output neurons, for example using a binary representation, or individual networks should be used for each letter.

3.4 Network Design and Interpretation

3.4.1 Design Criteria for Neural Networks

When defining the parameters for a neural network, one is faced with multiple decisions that will all have an impact on the quality of the trained network. Several of these options are now explained in more detail.

Hidden Layer

The number of hidden layers defines the abilities of the neural network, which mathematical functions it will be able to simulate. As shown, a network with no hidden layer can only learn linear separable problems. One hidden layer enables the network to learn arbitrary continuous functions and with two hidden layers, every function can be approximated with a neural network [32, 34].

The number of hidden neurons is another free parameter in the neural network design. If too few neurons are chosen the network will not be able to learn all characteristics of the training input patterns. On the other hand, if too many hidden neurons are chosen, the network might get overtrained. In the worst case it will learn every training pattern by heart and not be able to correctly classify any other input pattern (fig. 3.10).

Experience has shown that a good value for the number of hidden neurons lies between half the number of input neurons and double that number. Growing and pruning algorithms exist to alter the number of hidden neurons during the training, alleviating this choice [26, 35].

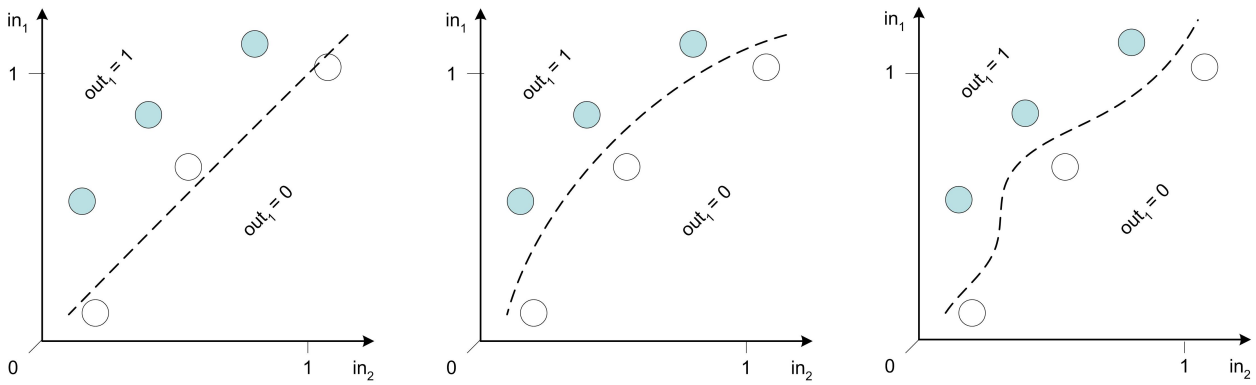

Figure 3.10: *Network input space* $\{\mathrm{in}_1, \mathrm{in}_2\}$ *with training patterns and the network separation line. On the left, the network was not able to learn all aspects of the training patterns and thus some patterns are classified wrong. In the middle, the network has learned to generalise between the two classes and a good separation line is drawn. On the right, the network has learned each training pattern by heart, leading to a poor division of the two class regions.*

Initial Weights

Even though the initial weights of a neural network are chosen randomly, several conditions should be taken into account. Most important, the weights should not be chosen equal. If it is still done, all neurons will behave in the same way forever. At the start of the training they all have the same connections, thus they will receive the same inputs, yield the same output and generate the same error, leading to the same weight updates. Basically a neural network with only one neuron per layer is generated by this means.

The weight modification Δw_{ji} is directly proportional to the derivative of the activation function $f'(a_j^q)$. Therefore it is advantageous to start with small weights and distribute them evenly around zero. Thus a_j^q will be initialised around zero, too, where the gradient of $f(a_j^q)$ is at its maximum for typical activation functions. This prevents an initial bias and leads to larger weight updates making the initial training faster.

Training Input Patterns

The choice of the training input patterns is especially crucial since the network will receive all its knowledge from these patterns. They have to be chosen carefully so that input patterns are taken of all classes that have to be distinguished. Special regard has to be given to the complex regions in input space, where different classes meet. Here, the input patterns will be quite similar and therefore difficult to separate. Furthermore the training patterns have to be distributed equally over all classes so that every class can be learned equally well. If not enough patterns are available, this can be simulated by weighting the individual patterns differently.

Generally the training patterns are divided into a training sample and a test sample, which both exhibit the same features as described above. In this case, the training sample is used for the network training and the test sample to survey the training process. If the neural network is trained too much, it could get overtrained, specialised in the separation of the training patterns and therefore losing its ability generalise. This behaviour can be restrained by using test samples (fig. 3.11).

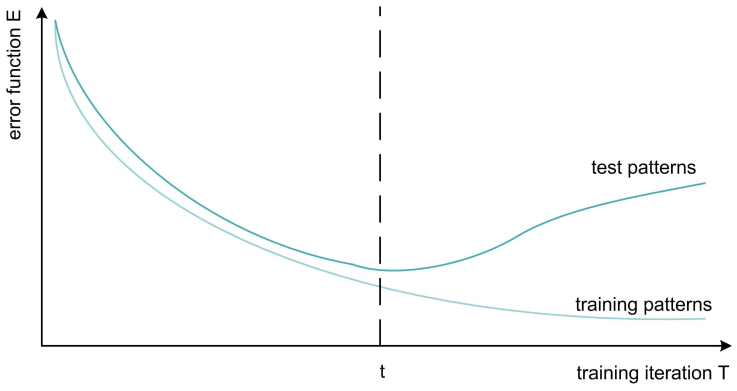

Figure 3.11: *Progression of the network error function for training and test patterns. At some iteration t the network begins to learn the training patterns by heart and the results on the test patterns begins to deteriorate.*

3.4.2 Interpretation of the Neural Network Output

In order to evaluate the quality of a neural network, one has to examine the ability of the trained network to separate signal events from background events. For a quantitative analysis of the quality, the variables purity P and efficiency E are defined:

$$\mathrm{P(out)} = \frac{\text{number of signal events with network output } > \text{ out}}{\text{number of events with network ouput } > \text{ out}} \tag{3.16}$$

$$\mathrm{E(out)} = \frac{\text{number of events with network output } > \text{ out}}{\text{number of events}} \tag{3.17}$$

The parameter out is a threshold for the interpretation of the network output. If the output is larger than c, the corresponding input pattern is regard as signal event and as background event otherwise.

Purity is a measure for how many of the classified events actually belong to that class. A high signal purity indicates that most events considered to be signal events are truly signal events. Efficiency denotes how many signal events of the whole sample got classified at all. A low signal efficiency shows that most input patterns produce a network output below the threshold c. For representative purposes the network quality is usually visualised as shown in figure 3.12.

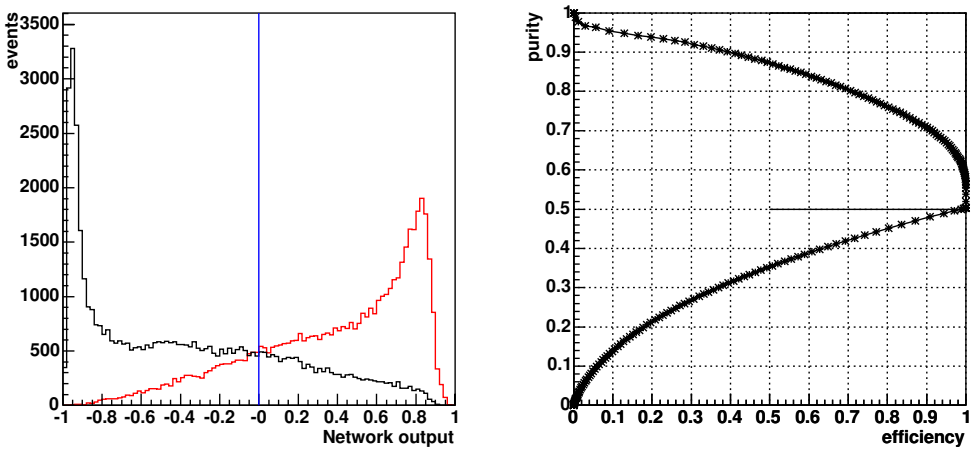

Figure 3.12: *On the left, the number of analysed events, divided into signal (red) and background (black) is plotted versus the network output. On the right, the purity of the signal is plotted versus the efficiency of the signal. Each point on this curve is a possible working point for the neural network. The nearer this point is to the top-right corner, the better the network will work.*

The network output scaled to [0, 1] is generally interpreted as a probability value. This assumption is only true, if the network output c for each input pattern is about equal to the associated signal purity P(c) (fig. 3.13). In this case, the neural network is trained optimally [36]. This can be shown in the following way: the mean contribution to the error function E for the

input patterns with the network output out can be described as

$$E = P(out) \cdot (1 - out)^2 + (1 - P(out)) \cdot (0 - out)^2 \tag{3.18}$$

The first term describes the contribution of the signal events with a purity P(out) and a target network output of 1. The second one represents the contribution of the background events which have a purity of (1 - P(out)) and a target output value of 0. The network training tries to minimise the error function, thus if E is minimal, the neural network is trained optimally. This leads to:

$$\frac{dE}{dout} = 0 \rightarrow P(out) = out \tag{3.19}$$

This makes the purity P(out) a linear function of the network output for an optimally trained network.

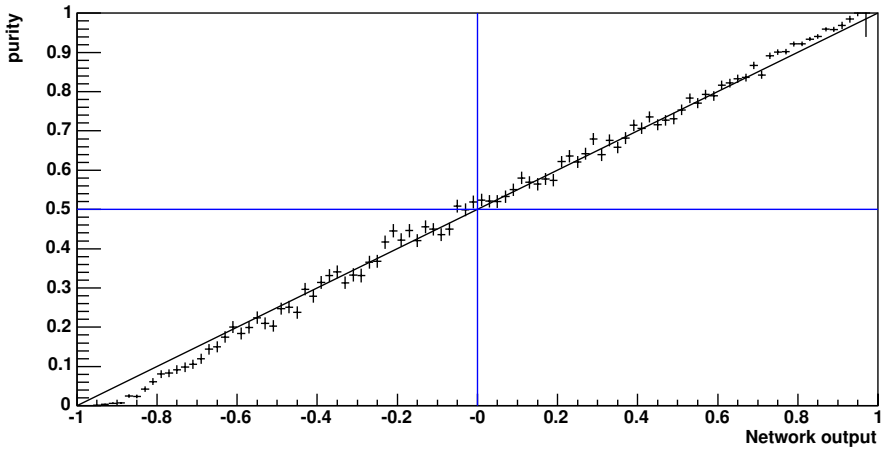

Figure 3.13: *The signal purity plotted against the network output. The better the points lie on the first bisector, the better the network training is regarding the interpretability of the network output as a probability.*

3.5 Other Neural Network Models

As mentioned in the introduction, the term neural network encompasses not only one single technique but a whole range of models. Not all of these models are predestined for the usage in this physical analysis context just as not every part of our brain is suited for every task. In order to give a broader view of the subject of neural networks, three more kinds of these networks are introduced. The radial basis function (RBF) network is similar to the neural network described above, only the hidden neurons are handled differently, using a RBF. Self-organising or Kohonen maps (SOM) pose a completely different learning approach as they try to classify the input patterns on their own. Finally Hopfield nets are an example of a network architecture with recurrent connections.

3.5.1 Radial Basis Function Networks

Radial Basis Function (RBF) networks [37, 38] are in general two-layer feed-forward networks, similar to the MLP described above. But in contrast to them, the hidden neurons in RBF networks feature radial basis functions as activation functions. Thus these neurons only produce a large output for input patterns that are close to their centre. Typically a Gaussian function is chosen as RBF:

$$\Phi_j(\mathrm{in}) = e^{-\frac{||\mathrm{in}-\mu_j||^2}{2\sigma_j^2}} \tag{3.20}$$

with in being an input pattern and μ_j, σ_j being parameters of neuron j similar to weights.

An idea behind using radial basis functions is to get a more visual representation of what the network has actually learned. Each neuron j in the hidden layer represents an area in input space with its centre μ_j and and variance σ_j. One or more of these neurons then represent a class. The output layer detects which hidden neurons were activated by the input pattern and thus which class the pattern belongs to.

However, this approach requires a good coverage of the input space by radial basis functions. A common solution is to associate each input pattern with a hidden neuron and then to apply shrinking techniques to reduce the size of the hidden layer. This is needed to avoid overfitting and reducing the computation requirements in the output layer.

Various training methods exist for RBF networks. In general one has to distinguish between the training of the hidden layer and the output layer. The latter represents in principle a simple perceptron and can be trained accordingly. On the other hand the hidden layer requires new techniques for adapting the centres, variances and even the number of hidden neurons. Here a modified delta rule or even a geometric approach is used [39, 32].

3.5.2 Kohonen Maps

Kohonen or self-organising maps (SOM) [40] are a neural network structure that uses unsupervised learning. It is based on a single layer of m neurons, organised in a grid structure (fig 3.14).

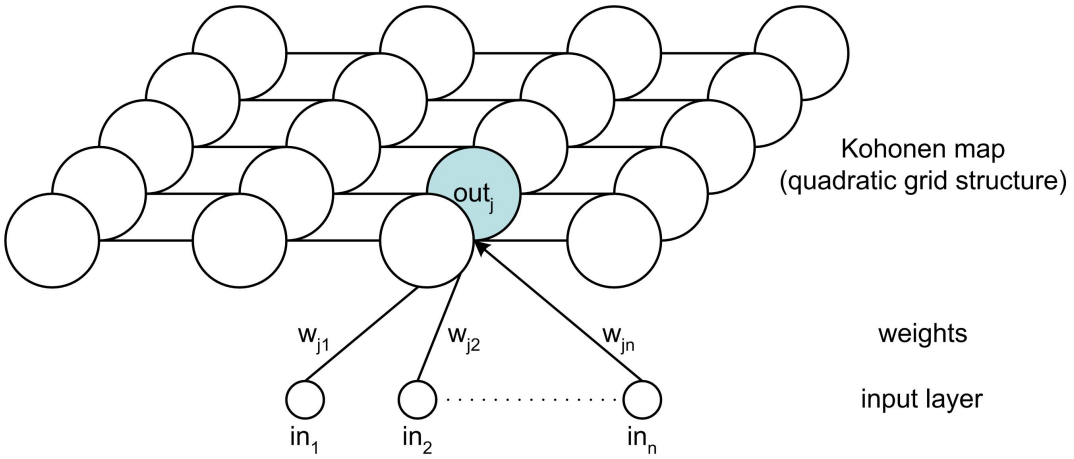

Figure 3.14: *Kohonen map with a quadratic grid structure. The inputs and weights for one neuron are shown.*

The neurons are only connected to the n inputs in_i, between them, only neighbourhood relationships exist. These relations induced by the grid structure are of special importance for the network training and interpretation. Furthermore each neuron has a weight vector **w** of the same dimension as the input patterns. There are several ways to compute the neuron outputs. One possibility is to calculate the inverse Euclidian distance:

$$out_j = \left(\sum_{i=1}^{n} (in_i - w_{ji})^2 \right)^{-1} \tag{3.21}$$

Training Algorithm

At first the n-dimensional weight vectors $\mathbf{w}_j = \{w_{j1}, ..., w_{jn}\}$ get initialised randomly for each neuron j. The training follows these steps:

1. An input pattern $\mathbf{in} = \{in_1, ..., in_n\}$ gets selected randomly and presented to the network.

2. The similarity of each neuron j to the input pattern is calculated by the Euclidian distance:

$$E = \sum_{i=1}^{n} (in_i - w_{ji})^2 \tag{3.22}$$

   The neuron with the smallest distance to the input pattern is then selected, labelled neuron k.

3. The neurons i in the neighbourhood of the best matching neuron k get updated by shifting their weights $\mathbf{w}_i$ closer to the input pattern **in** according to:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta \cdot \phi(i,k) \cdot (\mathbf{in} - \mathbf{w}_i) \tag{3.23}$$

with η being the learning rate and $\phi(i,k)$ being a distance function on the grid structure, also called neighbourhood kernel. It is a measure for the strength of the neighbourhood couplings between neuron i and neuron k.

The training is done several times for all training patterns until the results are sufficient.

Interpretation

Kohonen maps are especially good for visualising classes in a high-dimensional input space. They learn to map the n-dimensional input space to a lower-dimensional structure, most often a two-dimensional grid. The neuron's weights directly characterise the input pattern, it will be most sensitive for. Because of the training method, neurons which are sensitive to similar input patterns are clustered together. Thus the trained Kohonen map gives a good impression of the existing classes and their distribution in input space.

When a new input pattern is presented to the map, those neurons will produce the highest output values, whose weights are most similar to the input pattern. It can be classified according to the region of the Kohonen map that produced the highest network output. But as the training is unsupervised, there is no a-priori knowledge of which neurons represent which class. Therefore a tuning has to be done after the training of the network. Input patterns with a known classification are presented to the network and the responsive regions are noted in order to determine which areas of the Kohonen map describe which class.

Variations

The neighbourhood relations can be modelled with various grid structures. Commonly used are one-dimensional chains or two-dimensional grids. Quadratic grids are easier to compute whereas hexagonal ones yield better results [41].

Depending on the grid structure and the desired neighbourhood relations, the distance function $\phi(i,k)$ can be chosen in many ways, typically a function that is fast to calculate like a gauss distribution or a cylinder distribution:

$$\phi(i,k)_{\text{cylinder}} = \Theta(c - ||\mathbf{w}_i - \mathbf{w}_k||) \tag{3.24}$$

with parameter c defining the neighbourhood radius.

Preprocessing in the form of decorrelation and normalisation of the input variables in_j is also a useful extension of this network structure since the Kohonen map is sensitive to input variables from vastly different regions and correlations between them.

3.5.3 Hopfield Networks

Hopfield networks [42] are a kind of feed-back neural networks. In contrast to the feed-forward networks, a feed-back network also features backward connections, so that there is no explicit information flow from the input neurons to the output neurons. When an input pattern is presented to the network, it is updated until a stable state is reached or until a certain number of update steps are made.

Hopfield networks consist of only one layer of neurons (fig. 3.15). These act as input and output neurons at the same time. Each of them is connected with each other but not with itself. Thus the network exhibits only indirect feedback loops.

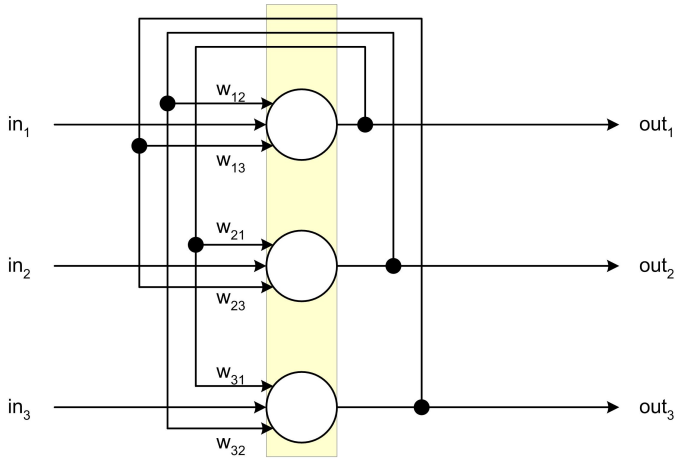

Figure 3.15: *Hopfield network with its single layer of three neurons.*

Hopfield neurons are designed as binary McCulloch-Pitts neurons with a Heaviside activation function. Thus the whole network can take 2^n different input and output patterns. The network weights are usually required to be symmetric, i.e. $w_{ij} = w_{ji}$. This is motivated by the idea of setting up an energy function for the network and analyse it with the methods of statistical mechanics.

The Hopfield network design can be expanded for a continuous activation function and continuous input and output values [43]. The considerations described below also apply in principle.

Weight Update Mechanism

The recursive structure of the Hopfield network provides different possibilities for updating the weights of the neurons.

- **synchronous update:** all neurons change their state at the same time.
- **asynchronous update:** a neuron gets selected randomly and its state gets updated.

The asynchronous update method is usually applied for Hopfield networks since it is easier to implement and especially suited for parallel computers. The synchronous update method on the other hand is better suited for theoretical calculations.

Training

The network training stores N input patterns $\{in_1^i, ..., in_n^i\}$ in the neural network by adjusting the network weights. This can be done in one step by the generalised Hebbs training law:

$$w_{ij} = w_{ji} = \begin{cases} \sum_{k=1}^{N} in_i^k \; in_j^k & i \neq j \\ 0 & i = j \end{cases} \tag{3.25}$$

However, the capacity of the network for storing patterns is limited. It can be approximated by the factor $\frac{N}{n} \approx 0,138$ [44].

Stability

Like in all feed-back systems, the changes in the Hopfield network have to decrease over time in order to reach a stable state. Otherwise the network output might start to oscillate. The Cohen-Grossberg theorem gives a sufficient criteria for the stability of such recurrent networks:

Cohen-Grossberg theorem (1983): *Recurrent neural networks are stable, if the following conditions are fulfilled for all network weights w_{ij}:*

1. $w_{ij} = w_{ji} \; \forall i \neq j$
2. $w_{ii} = 0 \quad \forall j$

This theorem can be proven by introducing an energy function for the network as shown below. It is also important to note that these requirements are not mandatory. A feed-back network can be stable without fulfilling them but has not to be.

Energy Function

The Hopfield network can be described by an energy function:

$$E(t) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ji} \; out_i(t) \; out_j(t) - \sum_{j=1}^{n} in_j \; out_j(t) + \sum_{j=1}^{n} \sigma_j \; out_j(t) \tag{3.26}$$

where the network inputs in_j, weights w_{ji} and thresholds σ_j stay constant during every network update, that is made.

It can be proven that E decreases with each step and does not stay constant until a trained pattern is reached – denoting a minimum in the energy function [42]. This energy function is similar to the Ising model in theoretical physics and thus the same thoughts can be applied.

Simulation and Analysis Tools

When performing analyses in a high energy physics context one usually prefers to use already established frameworks and tools to facilitate the task. These software environments are geared to perform especially well with the huge amount of data arising in such an environment. Most are under continuous development, leading to frequent updates with additional functionalities.

For this analysis as well various existing tools were used directly and indirectly to simulate, store and analyze data. These software packages are introduced in the following sections.

4.1 The CMS Software Framework

Since the LHC and the CMS detector are still under construction, no real data for analysis can be acquired. In order to prepare analyses before the start-up of the collider, a large software framework has been compiled, capable to simulate every step from the particle collisions up to the final detector output. It is also used for reconstruction purposes and physics analyses, which use realistic algorithms that will also be used on real data.

The data samples analysed in this thesis have been produced with this CMS software. Therefore, a short overview of its most important components concerning this thesis will be given.

CMKIN, CompHEP and PHYTIA

The physical processes in a particle collision are calculated by Monte Carlo event generators. In the CMS framework these generators are interfaced by the CMS kinematics software package CMKIN [45], which sets up the CMS and LHC specific parameters. It provides a common linkage between the physics event generators (CompHEP, PHYTIA, etc.) and the CMS detector simulation. This interface is based on the HEPEVT (High Energy Physics Event) common block, a current standard structure used in the CMS framework to store particle information about one event, that will be replaced in the future by the object-oriented HepMC (High Energy Physics Monte Carlo) standard.

CompHEP [46] is a package made for automatic calculations of elementary particle decay and collision properties in the lowest order of perturbation theory. For this purpose it offers several physical models for calculations. The package can be divided into a symbolic part, mainly for calculations of the squared matrix elements of the decay processes, and a numerical part for the generation of the hard decay process.

PYTHIA [47] is a Monte Carlo event generator including showering and hadronisation processes. The program is intended to generate complete events in as much detail as possible in leading order perturbation theory. Therefore, several models for various physical aspects are included (soft and hard interactions, initial- and final-state parton showers, etc.). It can utilise events generated by matrix element generators like CompHEP, ALPGEN or Madgraph.

CMSIM and GEANT3

Together the CMSIM [48] and GEANT3 [49] packages are used to describe the passage of particles through the detector. In our context, these are mostly particles originating from a collision.

GEANT3 is a general detector description and simulation tool. Its core functionality is to describe interactions of particles with matter. Thus, it is used to describe the interactions of the incident particles with the detector material as well as secondary processes, providing particle trajectories and energy deposits within the various detector layers.
The CMS simulation package CMSIM is an application of GEANT3. It provides the CMS detector specific descriptions needed for the simulation. In addition, it offers a simple interface to users not familiar with GEANT3 to get access to the full detector simulation.

CMSIM has been recently replaced by the newly developed package OSCAR (Object-oriented Simulation for CMS Analysis and Reconstruction) [50] which interfaces GEANT4 [49], the object-oriented successor of GEANT3. For this thesis data samples still generated with CMSIM were used since they were readily available.

ORCA

The ORCA (Object-oriented Reconstruction for CMS Analysis) [51] framework provides the reconstruction software for further event analysis. It is intended to be used for final detector optimisations, trigger studies, global detector performance evaluations and event reconstruction.

ORCA consists of several subsystems describing detector components such as tracker, calorimeter, etc. and classes describing reconstructed physical objects like vertices and jets. The necessary reconstruction algorithms such as vertex finders or jet finders are also part of this framework. ORCA is based on CARF (CMS Analysis and Reconstruction Framework) [52], a framework designed to prototype reconstruction methods. It enforces the design principles of action-on-demand (objects will only be reconstructed once and only if needed) and event-driven notifications (in response to events, observer classes will be called, that initialise the appropriate actions).

Further Components

The CMS Framework consists of many more components that are not presented here, because of their little importance for this thesis. But some shall be mentioned: FAMOS (a fast detector simulation and reconstruction software) [53], SCRAM (a software configuration management and build tool) [54] and XCMSInstall (a graphical CMS software installation tool) [55].

In the near future, the complete CMS framework is going to be replaced by an improved new framework, called CMSSW, which stands for CMS Software [56]. This framework is then also intended to be applied when the LHC commences its work in 2007.

4.2 ROOT

ROOT [57] is an object-oriented framework written in C++ aimed at solving the data analysis challenges in high-energy physics. It provides an extensive set of classes for analyzing and visualisation purposes such as curve fitting, fourvector handling or histogramming. ROOT was developed in the mid 1990s to succeed the then standard PAW [58] framework which was written in FORTRAN-77 and slowly reached its limits with the ever increasing amounts of data the experiments yield.

A notable feature of ROOT is CINT, its built-in C++ command line interpreter and script processor. This interpreter makes interactive data analysis possible without having to recompile after each change of the code, and thus facilitates rapid prototyping. one can also create a standalone analysis program in C++ by linking against the ROOT libraries which will run faster compared to CINT and can utilise all current C++ features.

Another important part of the ROOT software framework concerning this thesis are the class `TMultiLayerPerceptron` and associated classes. These classes provide a rudimentary implementation of a standard feed-forward neural network with various backpropagation learning algorithms, along with training optimisations like a momentum term and input normalisation. The class `TMLPAnalyzer` was also later integrated into the ROOT framework to measure and visualise the network's performance, thus giving the user a standardised tool for these analysing tasks. Introductory examples for the usage of these classes can be found in the ROOT distribution [59].

4.3 PAX

The Physics Analysis Expert (PAX) [60] is a C++ based toolkit, developed at the Universities of Aachen and Karlsruhe. It introduces a new level of abstraction between the event generator and detector reconstruction software on one side and the physics analysis algorithms on the other side. The actual physical data are saved in an independent PAX container structure, which the analysis code then uses. Thus, a new freedom from the underlying experiment-specific analysis software can be achieved. With analysis code no longer being dependent on the actual data format, it can be easily shared between various research groups and similar analyses can be compared quickly. Within one experiment the code also stays persistent against changes in the event generator and detector reconstruction software.

To achieve this task, PAX depends on interfaces that fill the data from the event generator or detector reconstruction software into PAX structures. Interfaces already exist for many commonly used physics data formats (`PaxTuple` for HEPEVT, `PaxHepMC` for HepMC, etc.) and if needed, new interfaces can be easily implemented due to PAX being published as free software under the GNU Lesser General Public License.

Another main feature of PAX is its ability to deal with multiple event interpretations. Current analyses often have to deal with a lot of particles per event, which leads to ambiguities as depicted in chapter 2.3.4. PAX provides functions to manage and evaluate the possibly huge combinatorics.

PAX Class Structure

The main unit of PAX is the `PaxEventInterpret` class (see fig. 4.1), a general container for high energy physics events. It comprises further containers for the physical objects of the event like vertices (`PaxVertex`), fourvectors (`PaxFourVector`) or collisions (`PaxCollision`) as well as additional values needed in the course of the analysis. The `PaxFourVector` class either inherits from the `HepLorentzVector` class provided by the CLHEP [61] package or from the `TLorentzVector` class of ROOT. Thus basic fourvector functionality is obtained and a connection to an existing analysis software is made by extending the CLHEP or ROOT functionality, respectively.

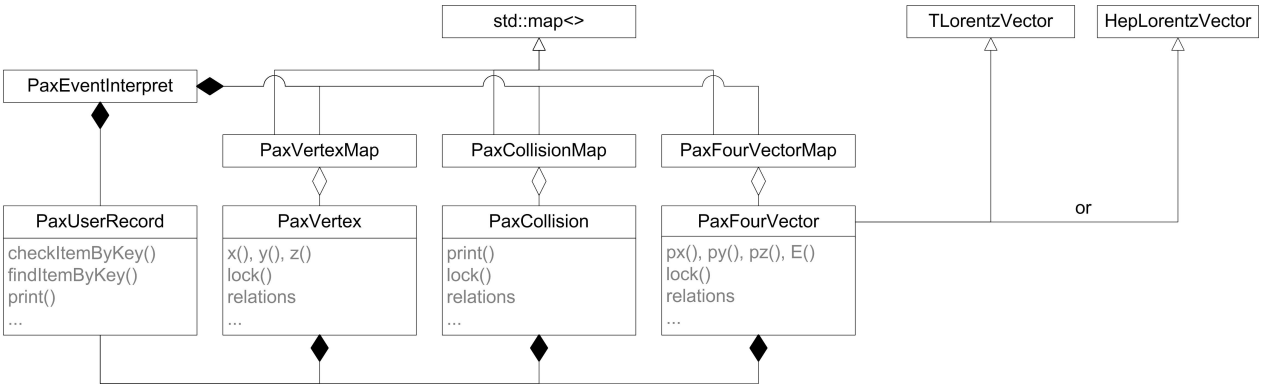

Figure 4.1: *The PaxEventInterpret class represents the basic unit in PAX. It contains physical objects like fourvectors (PaxFourVector), vertices (PaxVertex), collisions (PaxCollisions) and additional values (PaxUserRecord).*

Relations between physical objects in an event can be manifold. Most often one has to deal with many-to-many relations that are not easy to handle. For this purpose the class `PaxRelationManager` (see fig. 4.2) was devised to deal with the relations that can arise within one `PaxEventInterpret`. This class construct utilises the *mediator* design pattern [62] to achieve its task. It attends to the connections between fourvectors, vertices, etc. constructing the whole decay tree of the event. One application of these relations is the locking mechanism included in PAX. It allows excluding certain objects and their entire decay tree from the analysis.

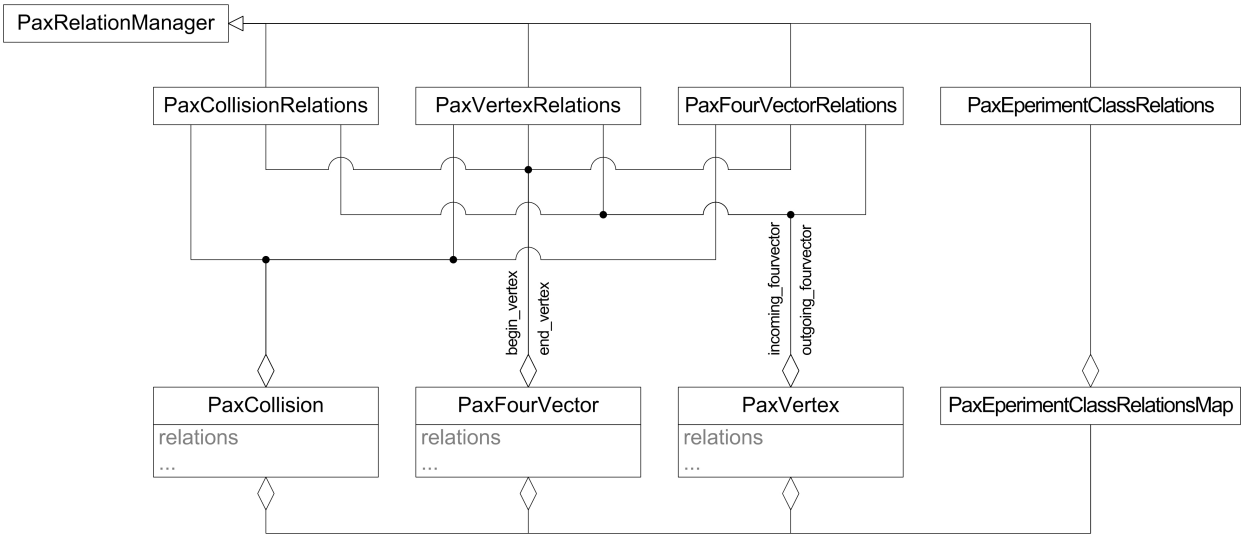

Figure 4.2: *The PaxRelationsManager controls the relations between the physical objects in an event, providing additional information like the decay tree of the event and additional functionality like the locking mechanism.*

PAX Persistency

The PAX framework also provides a functionality to store `PaxEventInterpret` objects to disk. These objects can then later be analysed with a stand-alone PAX-based analysis. Thus, it is possible to store all relevant objects like fourvectors or vertices and the relations between them, retaining the complete decay trees and event histories.

Further information about the PAX libraries including documentation, source code and example analyses is available at the official PAX website located at CERN [63].

4.4 NeuroBayes®

NeuroBayes® [64] has been developed at the University of Karlsruhe by Prof. Dr. Michael Feindt et al. It is now distributed by $<$ Phi $-$ T $>$ [65], a spin-off company also located in Karlsruhe, that makes its functionality available outside of physical contexts, e.g. for optimisations of car insurance conditions.

NeuroBayes® represents an advanced neural network technology utilising modern and sophisticated algorithms. It is based on a three-layer feed-forward network with a backpropagation algorithm for the training. This network is accompanied by complex preprocessing routines and significance control mechanisms that considerably enhance the network's performance. NeuroBayes® utilises Bayes' theorem[1] to take a priori knowledge into account. This is advantageous since the

[1] $P(\text{theory}|\text{data}) = \frac{P(\text{data}|\text{theory}) \cdot P(\text{theory})}{P(\text{data})}$: experimentally measured data is interpreted in context of the theory

network output will thus never lie outside of the training range, i.e. unphysical results will not arise. In addition, the neural network output can be interpreted as Bayesian a-posteriori probability when using this approach. For this, signal and background patterns have to be trained with target values of 1 and 0, respectively.

Structure

The NeuroBayes® package is based on two components, the Teacher and the Expert. The former is used to train the neural network. At first, (pre-)processing parameters have to be adjusted for the actual training. Then, the Teacher receives all training patterns along with their associated target value(s). After they have been preprocessed, the network is trained with these data samples. Then, the resulting network description is stored in a file, the Expertise, along with an additional file containing information about the training process and its results. The Expertise can be read subsequently by the Expert component of NeuroBayes® to set up the neural network, which is used to analyse new data.

Preprocessing

One of the main features of NeuroBayes® is its extensive preprocessing capability. With its sophisticated routines, an automated and robust preprocessing of the input data becomes possible. It transforms the input data into another representation better suited for network training without losing information inherent to the data samples. This step is especially important, as the search for the global minimum can usually be accelerated and the final results improved. NeuroBayes® supports many different preprocessing options that are further described in [64]. They can be separated into two groups:

- **General Preprocessing:**
  Several preprocessing methods are executed on all input data if not defined otherwise. At first, the input variables are equalised and scaled to a range of [-1, 1], so that extreme input values have no excessive influence on the training. In a second step, these flat distributions are transformed into a normalised Gaussian distribution.
  The input variables are correlated to the target value but also to each other, which complicates network training. Therefore, they are rotated in the parameter space they span until they are linear independent to each other.

- **Individual preprocessing:**
  Specific preprocessing settings can be defined for each individual input variable to further exploit the user's knowledge of these variables and improve the training. Thus, a special treatment of variables with discrete values or a δ-function becomes possible. Also regularised fits with spline functions can be applied to the distributions of the input variables.

Pruning

Another notable feature of NeuroBayes® is its ability to alter and thus improve its predefined network structure. This optimisation process happens through two different kinds of pruning methods implemented in NeuroBayes®:

- **Input variable pruning:**
  Before the actual training begins, NeuroBayes® excludes certain input variables from the training. The decision which variables are to be pruned in this way is based on the statistical significance of the input variables in relation to the training target. This behaviour is entirely controlled by the user, who defines the significance threshold to be used by the network preprocessor, if any.

- **Connection pruning:**
  During the training phase NeuroBayes® monitors the performance of each network connection and can eliminate those connections which have become irrelevant for the network output by setting their weight to exactly zero. This alters the network structure and lowers the number of free parameters, leading to an easier training in the following training iterations.

Usage

NeuroBayes® has originally been implemented in FORTRAN and further development still continues in this language. To open up its possibilities to a larger number of users, several interfaces have been devised, e.g. for C++ with ROOT. The C++ interface is now commonly used by the CDF and CMS working groups at the University of Karlsruhe.

But some limitations coming from its FORTRAN origin have to be kept in mind. For example, only one instance of the Teacher component may run at the same time because of its internal structure, whereas several Experts can run simultaneously. The relevant C++ classes have been implemented accordingly.

An abstract wrapper class for neural network usage in general has been developed in the scope of this thesis. Two concrete classes have been derived for existing neural network packages of which one is a dedicated NeuroBayes® wrapper and the other interfaces the neural network included in the ROOT package. These classes are described in more detail in appendix A.

More information about NeuroBayes® can be found in [64].

4.5 Software Versions

The various software versions used in the scope of this thesis are presented in table 4.1. This includes production software for the generation of the data samples used, as well as analysis software for the subsequent processing of this data. Fundamental programs like the underlying Linux distribution, the C++ compiler, etc. are not listed.

Software package	**version**
compHEP	41.10
PYTHIA	6.215
CMSIM	133
ORCA	7.6.1 and 8.7.4
PAX	2.00.10
ROOT	4.02/00
NeuroBayes	1.3 (internal version)

Table 4.1: *Software versions used for simulation and analysis within the scope of this thesis.*

Jet Pairing

The CMS detector will observe the particles produced in proton-proton collisions at the LHC. But this is only part of the information needed for analysing such events. It is also necessary to find the relations between the detected particle jets, to know which of them decayed from the same mother particle and essentially to rebuild the entire decay tree of the detected event. In order to achieve this, the detected jets have to be successfully paired with the expected particles of the decay. This jet matching can be an easy task or quite difficult, depending on the character of the observed event.

In this thesis, the decay channel $t\bar{t}H$ and one of its background channels ($t\bar{t}b\bar{b}$), as depicted in chapter 2.3.3, are analysed. These channels have a complex topology for jet matching, consisting of four b quarks, two light quarks and a lepton with an associated neutrino. In addition, there are usually additional partons detected, coming from initial state radiation, final state radiation or not from the current collision at all. This leads to a very large number of possible pairing combinations of which the correct one has to be singled out.

There is already an existing study on jet matching for these channels based on a likelihood method by Alexander Schmidt [1]. This study has been used as a basis for the approach with neural networks presented here.

5.1 Data Samples

For this thesis two data sets were provided by Alexander Schmidt, one for the signal process $t\bar{t}H$ and one for the background process $t\bar{t}b\bar{b}$. The events were produced as described below, using the software packages and versions described in chapter 4.

At first the hard decay process of two protons was simulated with CompHEP. For the signal samples the Higgs mass was set to $m_{Higgs} = 120$ GeV and no restrictions in phase space were made. For the background process the following constraints were required for the two additional b jets not coming from top decays: $P_t > 15$ GeV, $|\eta| < 3$ and $\Delta R(b_1, b_2) > 0.3$. In both cases, the top mass was set to $m_{top} = 175$ GeV and the bottom quark mass to $m_{bottom} = 4.62$ GeV. The PDF model used was CTEQ4l. The compHEP output was then interfaced to PYTHIA for simulating parton showering and fragmentation to produce a full generator event. Here, only events with a muon in the final state were selected. Next, the interaction of the generated partons with the detector material was simulated with CMSIM and GEANT3. After this detector

simulation, ORCA was used to calculate the response of the detector electronics to the energy deposit in the detector by the partons. The reconstruction of physical objects like tracks and jets was then also done by ORCA although with a different version, leading to a reconstructed event. Finally these events were filled into `PaxEventInterprets`. Here, two versions of each event are stored, one containing the information after the event generation with PYTHIA and one containing the event information after the full reconstruction with ORCA. In addition both event interpretations include further information for identifying the various jets. In case of the reconstructed data this could not be fully done for each event. Here the reconstructed jets had to be matched with the ones in the generator event, which becomes impossible if the observed jet is headed along the beam line or too close to another one. Thus only some reconstructed events have the true information from the generator level needed for the network training. These are called good matched events.

Some size information of the signal and background event samples are shown in table 5.1.

	Signal Samples	**Background Samples**
All Events	86907	83292
Good matched Events	16601	13869

Table 5.1: *Event sample sizes for the events used in this thesis. Good matched events are those reconstructed events for which the information needed for jet identification could be retrieved completely.*

5.2 Ambiguities in the Final State

The events to be examined by the jet pairing neural network exhibit a complex topology with four b jets, two light quark jets, one muon and missing energy in the final state. Two of the b jets and the light quark jets come from top decays and the other b jets stem from the Higgs decay in case of a signal event. To determine the correct association of these jets to the original partons of the process, every possible combination has to be evaluated by the neural network. Altogether there are $6! = 180$ possible pairings that have to be analysed. To reduce this number it is at first assumed that a perfect identification exists for the b jets, reducing the number of combinations to $4! \cdot 2! = 48$. This can be further reduced by a factor of two, since the b jets from the Higgs boson are indistinguishable and once more by a factor of two, since the two light quark jets from the hadronically decaying W boson are indistinguishable, too. This leads to a final number of 12 possible jet matchings that have to be tested.

Since an event also contains a neutrino represented by the missing energy, the number of pairings is increased again by a factor of about two, because the missing energy cannot be reconstructed unambiguously, as explained in chapter 2. Two possibilities usually remain that have to be taken into account.

An event normally contains more jets than the ones described above. These additional jets are the result of initial state radiation, final state radiation or caused by the pile-up of events in the detector. A realistic analysis has to take them into account, too, increasing the combinatorics by a large factor, especially if b jets cannot be identified perfectly. The analysis presented here uses only the partons inherent to the clean decay tree as shown in chapter 2.3.3 and assumes that a perfect identification of b jets exists, unless otherwise noted.

5.3 Description of the Analysis Methods

An analysis with a neural network can be divided into two phases. At first there has to be a training phase, in which the network gets tuned to the problem, followed by an application phase, in which new data is passed to the network for analysis. Both steps include several similar proceedings that are fully discussed in the training phase.

Training of the Network

The analysis loops over all events intended for the training. These get passed to the actual training method, where all possible jet combinations, according to the restrictions mentioned in the last section are created. For each pairing the whole decay tree is reconstructed and the input variables for the network training are calculated. Subsequently, these input patterns are passed to the neural network in an arbitrary order. Depending on the number of possible reconstructions of the missing energy for each combination, there are about eleven times more wrong jet combinations than there are correct ones. To compensate this inequality, varying weights are applied to the input patterns for the training: $\text{weight}_{\text{sig}} = 11$ and $\text{weight}_{\text{bkg}} = 1$.

After all events have been processed by the training method and the neural network has been filled with all input patterns for each jet matching, the actual network training is initiated.

Application of the Network

Like for the training, each event designated for investigation is passed to the application method where all jet combinations and subsequently all input patterns are generated. For each event every input pattern is then passed to the neural network and an output value in the range of [0, 1] is generated. This value describes how good each combination is in relation to the truth, higher values indicating a better jet matching.

All combinations for each event are sorted according to their associated network output, whereas the combination with the best output is chosen as correct. During this step, further information is produced including the impact of restrictions on the minimal network output for the best jet matching, on which will be referred later in section 5.5.1. In addition the actual jet matching that produced the best network output is recorded for each event.

5.4 Analysis of Generator Events

The aim of this analysis is to train a neural network that will state whether a certain pairing of jets is the correct one or not. It is first performed on generator events since these exhibit clearer signals. Therefore discriminating variables are easier to be determined. In the following, correct combinations are also named signal patterns and wrong pairings are called background patterns.

Variables

In this early phase of the analysis, five variables were used to differentiate correct jet combinations from false pairings. These variables, used as input pattern for the neural network, are the same as for the likelihood approach in [1]. To understand the meaning of them, the physical values they are based on, are introduced at first:

- **geometry**:
  The geometry of an event including the axis and angles is visualised in fig. 5.1. In CMS the direction of the z axis is defined by the direction of the beam line.
- **rapidity r**: $r = \frac{1}{2}\ln\left(\frac{E+p_z}{E-p_z}\right) = \tan^{-1}\left(\frac{v}{c}\right)$
  The rapidity r is a kinematic variable of a particle, with c being the velocity of light and E, p_z and v describing the particle's energy, momentum in z direction and velocity.
- **pseudo-rapidity** η: $\eta = -\ln(\tan\Theta/2)$
  The pseudo-rapidity η is a good approximation of the true rapidity r for large particle momenta ($p^2 \gg m^2$). This is useful since η can still be measured, even if the mass and momentum of the particle is unknown.
- **distance in phase space ΔR**: $\Delta R = \sqrt{\Delta\eta^2 + \Delta\Phi^2}$
  The direction of an outgoing particle is defined by η and ϕ. Particles travelling in the same direction lie near to each other in the $\eta - \phi$ space. Therefore ΔR is a good measurement for the actual distance of two jets.

Some of these physical values are not used until in the next chapter, but they are presented here already for the sake of completeness. The input variables for the neural network are listed below. Their distributions for reconstructed events can be seen in appendix B:

1. **$t_{Whadronic}$ mass**:
   Mass of the top quark with a hadronically decaying W boson daughter.
2. **$t_{Wleptonic}$ mass**:
   Mass of the top quark with a leptonically decaying W boson daughter.
3. **$W_{hadronic}$ mass**:
   Mass of the hadronically decaying W boson.

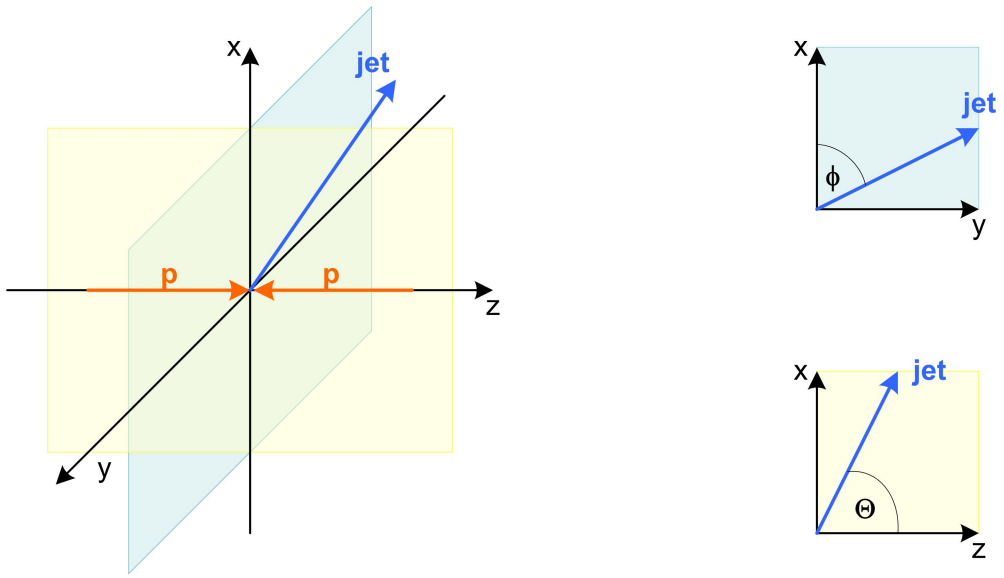

Figure 5.1: *Visualisation of the event geometry with the coordinate axis and the angles Φ and Θ. On the left, a three-dimensional view of the coordinate system is given. The other two images show projections on the xy- and the xz-plane. The z axis is defined by the beam line.*

4. $\mathbf{\Delta R(W_{hadronic},\ b_{Whadronic})}$:
   Distance in phase space between the hadronically decaying W boson and the b quark coming from the same top decay as the W boson.

5. $\mathbf{\Delta R(\mu,\ b_{Wleptonic})}$:
   Distance in phase space between the muon produced by the W boson decay and the b quark coming from the same top decay as the W boson.

The top quarks are reconstructed from jets, for which the correct matching is determined by the neural network. If misidentified jets are used for the reconstruction, it is not likely that the correct top mass will be reconstructed. Therefore these masses are appropriate variables for distinguishing signal and background patterns. The same is true for the mass of the W boson. But since a perfect identification of b jets is assumed, the jets contributing to the W boson are always paired correctly, leading to the same W mass distribution for signal and background events. Therefore, this variable is excluded from the analysis.

When the top quark decays, most of its inner energy is used in the production of the W boson and the b quark. The remaining energy is used for boosting these two decay products back-to-back in the top quark rest frame. The longitudinal momentum of the top quark is transferred to its two daughter partons, leading to a longitudinal boost of the W boson and the b quark that is much larger than their transversal boost. Therefore, the distance in phase space between these two partons is rather small compared to the distance between one of them and an arbitrary parton. $\Delta R(W_{hadronic},\ b_{Whadronic})$ attempts to use this information. The same argumentation holds true for the input variable $\Delta R(\mu,\ b_{Wleptonic})$.

Results

The neural network is trained with 60,000 of the generator events using the true missing energy information from the generator. The evaluation of the quality of the trained network is done twice, using the remaining events, once with the full missing energy information from the generator and once with the reconstructed missing energy[1]. When using the true missing energy information, the results are naturally almost perfect, yielding a correct pairing in 92.7% of all events. The more realistic analysis with areconstructed missing energy yields 71.7% events with a correct jet matching (fig. 5.2).

As can be seen, the signal patterns also exhibit a large amplitude for small network output values. This is easily explained: for each correct pairing about two signal patterns are submitted to the neural network since there are almost always two possibilities for the reconstructed missing energy. Because only one of them is correct – which one cannot be determined – there are essentially false combinations classified as signal patterns in the testing phase leading to this additional peak at small values of the network output.

If the jet matching with the best network output per event is not the correct combination, it is most often a combination in which one b quark from the Higgs decay was associated to a top quark instead and vice-versa. In this case the top quark is most often the top quark with the leptonically decaying daughter W boson. This behaviour has to be verified for reconstructed events and suppressed if possible.

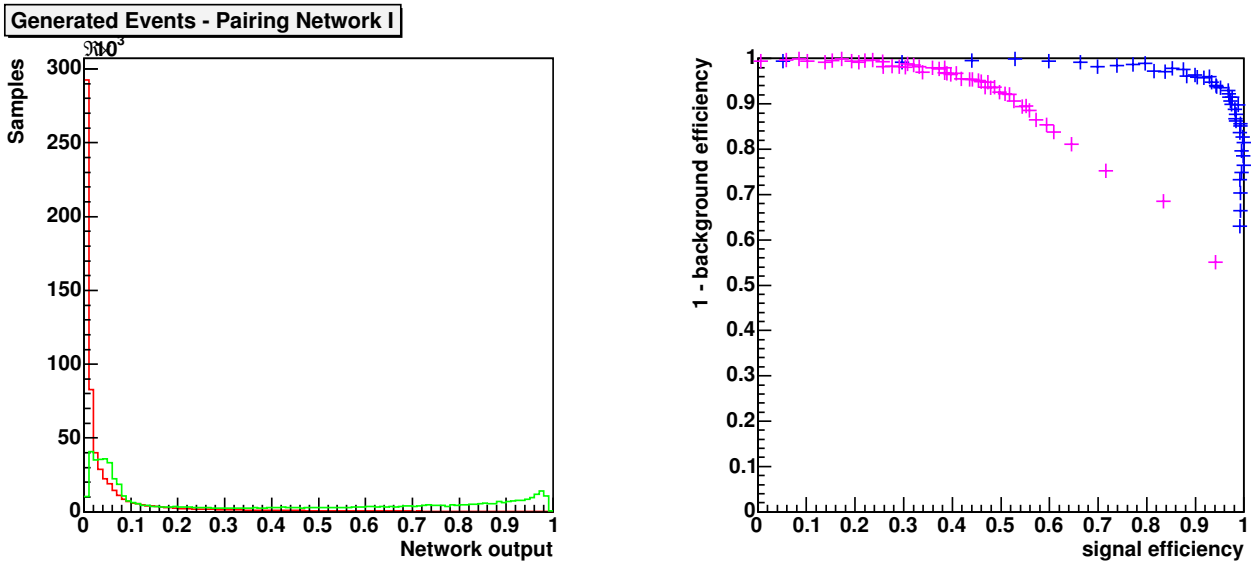

Figure 5.2: *On the left-hand side, the network output using generator signal events with a reconstructed missing energy is shown (correct combinations: green, wrong combinations: red). For visualisation purposes, the number of signal samples is scaled accordingly to the number of background samples. On the right, the efficiency plots can be seen for generator signal events with both, reconstructed (pink) and true missing energy (blue).*

[1]The calculations of the missing energy are done by the class `TtHMEzCalculator` provided by A. Schmidt.

5.5 Analysis of Reconstructed Events

Having achieved adequate results with generator events, the same analysis is applied to reconstructed events. It will naturally deliver worse results, since the whole reconstruction process entails various imprecisions: the detector resolutions are finite, the reconstruction methods are imperfect and the actual event geometry might pose problems, e.g. if jets are headed along the beam pipe. All of these problems lead to less clear distributions of the variables that are used for separating signal from background patterns. Therefore the resulting network performance is worse than for generator events (fig. 5.3).

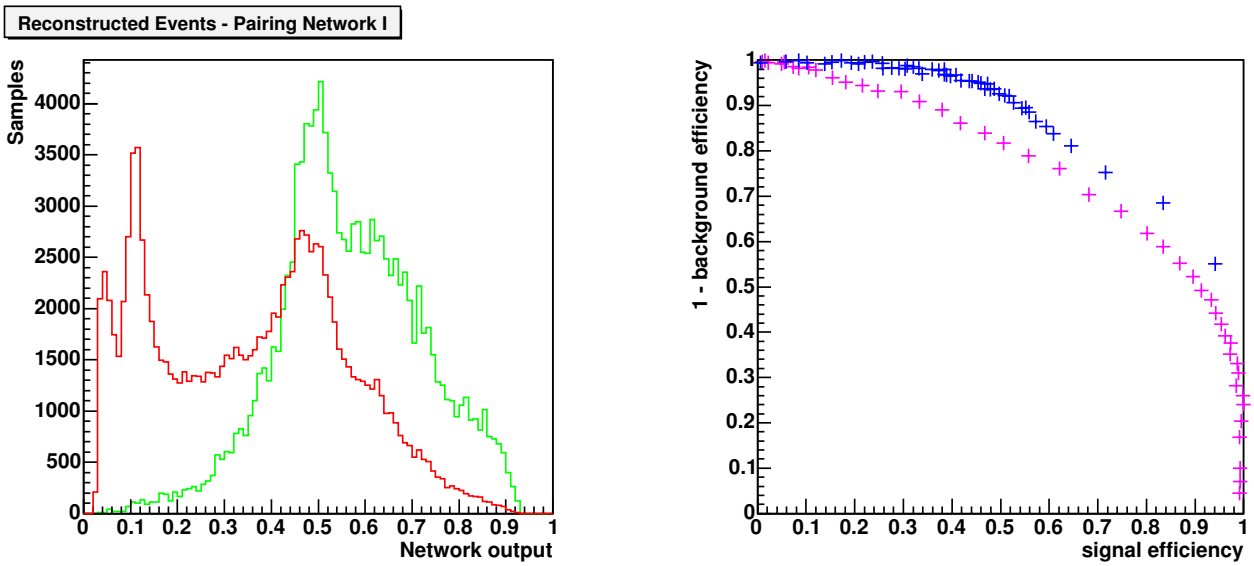

Figure 5.3: *On the left-hand side, the network output using reconstructed signal events is shown (correct combinations: green, wrong combinations: red). For visualisation purposes, the number of signal samples is scaled accordingly to the number of background samples. On the right, the efficiency plots can be seen for reconstructed signal events (pink) and generator signal events with a reconstructed missing energy (blue).*

The reconstructed events that can be used for the training of the network and for testing its performance are limited to good matched events – events for which the true jet identities can be retrieved. Without this information available, there is no criterion for the correctness of a jet combination. Therefore neither an input pattern with a target output value can be created nor the network output can be evaluated with respect to its correctness. Like for generator events, about 70% of the available events are used for training and the remaining for the performance test. A perfect identification of b jets is assumed again, thus only the four b jets have to be paired correctly.

The network output looks good and especially the efficiency plot is not much worse when compared to the network with generator events. Unfortunately the amount of events for which the correct jet combination is found drops to 30.7%. For the remaining events, mostly one of the b jets from a top decay is correctly associated to its mother particle and the other to the Higgs boson, just as with generator events.

5.5.1 Improving the Analysis

In order to improve the results of the pairing network achieved so far, several steps are taken. At first, additional variables are introduced helping the neural network to discriminate signal from background patterns. Taking only events, whose best network output is above a certain threshold is also discussed as another possibility to increase the amount of correctly paired events. At last, network adjustments are tested to improve the analysis further.

Additional Variables

If the correct jet combination is not found by the network, pretty frequently a combination where one b jet is falsely associated to a wrong branch of the $t\bar{t}H$ decay tree is chosen instead – with the hadronically decaying top, the leptonically decaying top and the Higgs each being the origin of one branch. Decreasing the network output of these combinations should lead to an overall improvement in the network performance. The following variables are determined as useful:

1. $\Delta R(b_{higgs1},\ b_{higgs2})$
2. $\Delta R(b_{Wleptonic},\ t_{Whadronic})$
3. $\Delta R(W_{leptonic},\ b_{Whadronic})$
4. $\Delta R(b_{Whadronic},\ t_{Wleptonic})$
5. $\Delta R(W_{hadronic},\ b_{Wleptonic})$
6. $\Delta R(q_1,\ b_{Wleptonic})$
7. $\Delta R(t_{Whadronic},\ t_{Wleptonic})$

The distributions of these seven variables are plotted in appendix B, separated for signal and background patterns.

In contrast to the previous section, two jets from different decay branches are chosen for the ΔR variables. These will only exhibit a small ΔR value if a wrong jet combination is chosen and their constituting partons belong to the same decay branch. For $\Delta R(b_{higgs1},\ b_{higgs2})$ the same explanation as in section 5.4 holds true, although to a lesser extent since the two b jets together are not as heavy compared to their Higgs mother as the W boson and the b quark are compared to the top quark.

Some more variables have been suggested in being useful and included in the neural network for testing their ability to improve the performance of it. But since they contribute only negligible improvements, these five variables are omitted in the further analysis. They are listed below:

1. $|\phi(\mathrm{b_{higgs1}}) - \phi(\mathrm{b_{higgs2}})|$
2. $\Delta\mathrm{R}(\mathrm{t_{Whadronic}}, \mathrm{higgs})$
3. $\Delta\mathrm{R}(\mathrm{t_{Wleptonic}}, \mathrm{higgs})$
4. $\measuredangle(\mathrm{b_{higgs1}}, \mathrm{b_{higgs2}})$
5. $|\cos(\Theta_{\mathrm{higgs}})|$

The denotation of the individual partons is visualised in fig. 5.4 for a signal event. There are essentially three different branches in the $\mathrm{t\bar{t}H}$ decay tree. One originating from each top quark and the third starting at the Higgs boson. The partons in each branch are subscripted according to the defining property of this branch – either a hadronically or leptonically decaying W boson or a Higgs boson. In case of a background event the two b quarks labelled $\mathrm{b_{higgs1}}$ and $\mathrm{b_{higgs2}}$ do not come from a Higgs decay but directly from the proton-proton collision. The jets $\mathrm{q_1}$ and $\mathrm{q_1}$ as well as $\mathrm{b_{higgs1}}$ and $\mathrm{b_{higgs2}}$ are sorted in ascending order according to their masses.

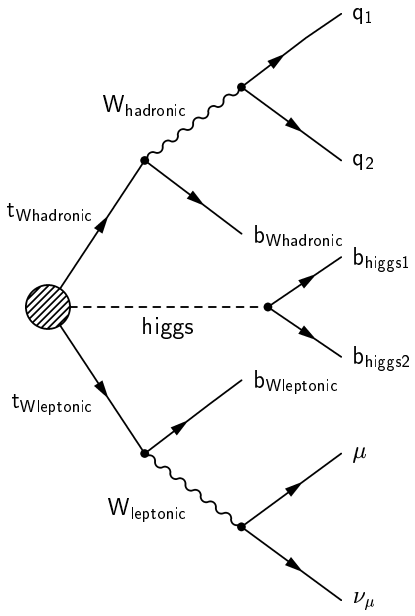

Figure 5.4: *Denotation of the partons in the* $\mathrm{t\bar{t}H}$ *decay channel.*

Application of Cuts

The network output is an estimation for the correctness of the analysed jet pairing. Therefore only events whose best jet combination has a network output above a certain threshold are considered. In table 5.2 the resulting percentages of correct pairings are listed as well as the remaining fraction of the overall events.

Threshold	0.0	0.5	0.6	0.7	0.8	0.85
Correct Pairings	30.7%	31.3%	32.7%	37.2%	45.2%	49.3%
Efficiency	100.0%	95.3%	81.2%	52.3%	24.5%	12.2%

Table 5.2: *Observing only events whose best combination has a better network output than the threshold. The percentage of correct pairings for all considered events and the overall percentage of events considered is shown.*

As can be seen, the percentage of correct pairings increases slightly with a higher threshold but the efficiency drops by a huge amount since more events get rejected. Only about 10,000 $t\bar{t}H$ events will be detected at CMS in the low-luminosity phase each year [19]. Therefore reducing them by 75% for a small improvement of only 15% in the jet pairing by using a threshold of 0.8 is not recommended.

Further Possibilities

The training of the neural network can also be improved by altering the network preprocessing and the manner in which the input variables are presented to the network. These are essentially only technical measures in the application of the neural network and thus have no physical background, but they might help in obtaining more correctly paired events.

At first, strongly correlated variables have been systematically scheduled for decorrelation instead of just trusting the automatical decorrelation routines. This has already led to a better network performance.

Furthermore the mass variables have been presented twice to the network with different preprocessing options once altering the input by using a Gaussian transformation (preprocessing flag 12) and once using a regularised spline fit on the mass variables (preprocessing flag 14). Since this has not produced noticeable improvements, only one preprocessing will be used in the further analysis.

Finally three different possibilities for weighing the signal and background patterns have been checked – not weighing them at all, weighing signal patterns with 11 and background patterns with 1 or weighing signal patterns with 1 and background patterns with 11^{-1}. Here the second alternative yields the best results and is used subsequently.

Results

The presented modifications lead to an improvement of about 3% in the ability of the network to find the correct jet combination for each event – 33.4% are paired correctly. The results of the improved network are shown in fig. 5.5 and table 5.3. It can be seen that the efficiency stays about the same but the network output looks much better with a clearer separation between the maximum of the signal and background patterns.

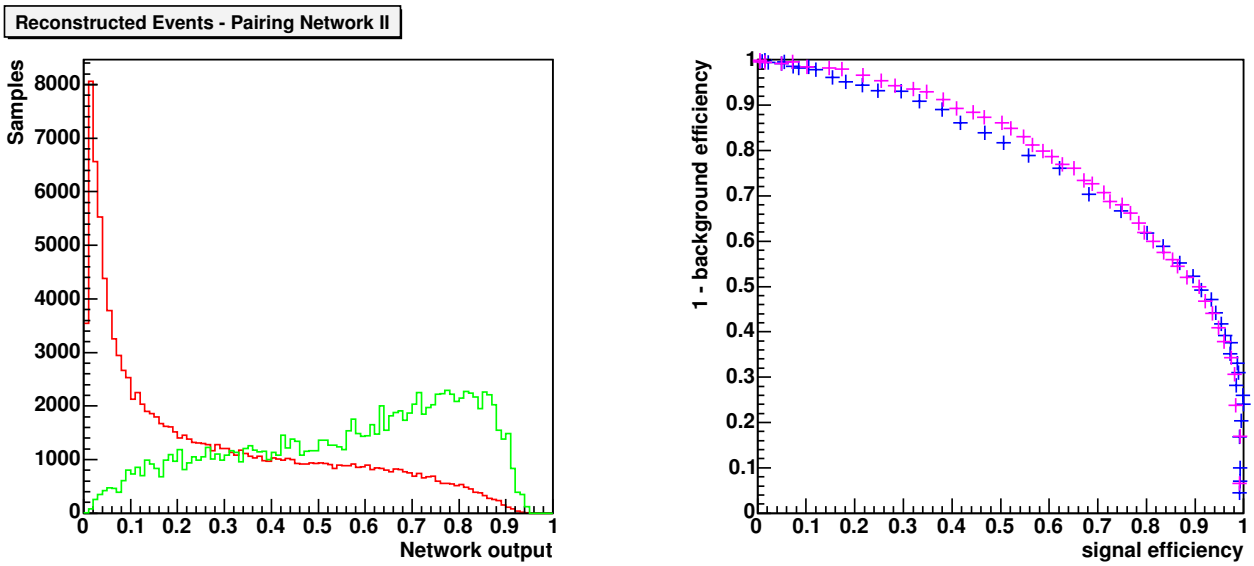

Figure 5.5: *Performance of the improved network on signal events. On the left, the network output is shown. On the right, its efficiency is shown (pink) as well as the efficiency of the previous network (blue).*

Threshold	0.0	0.5	0.6	0.7	0.8	0.85
Correct Pairings	33.4%	33.8%	34.4%	35.8%	40.2%	45.5%
Efficiency	100.0%	96.0%	90.9%	77.8%	48.8%	26.6%

Table 5.3: *Results of applying a threshold on the output of the improved neural network using signal events.*

5.5.2 Analysis of Background Events

The neural network has also to deliver good results when confronted with background events. There are two desired behaviours, either the network output is low for every jet combination of the background events or the pairing efficiency is high. In the first case an event can be easily rejected, reducing the number of background events in this stage already. In the second case, the correct jet matching is often found, facilitating the classification task of the background suppression network that receives this information subsequently.

Modifications to the Analysis

As already mentioned above, background events lack the two b jets coming from a Higgs decay. Therefore two other b jets not coming from top decays are chosen instead. For compensating the Higgs boson, all calculations are applied to this $b\bar{b}$ system in the background samples instead.

Results

Like for signal events, only good matched events can be used. 70% of them have been used in the training and the remaining ones are taken for testing purposes with a perfect b jet identification assumed. The network output and the associated efficiency plot can be seen in fig. 5.6.

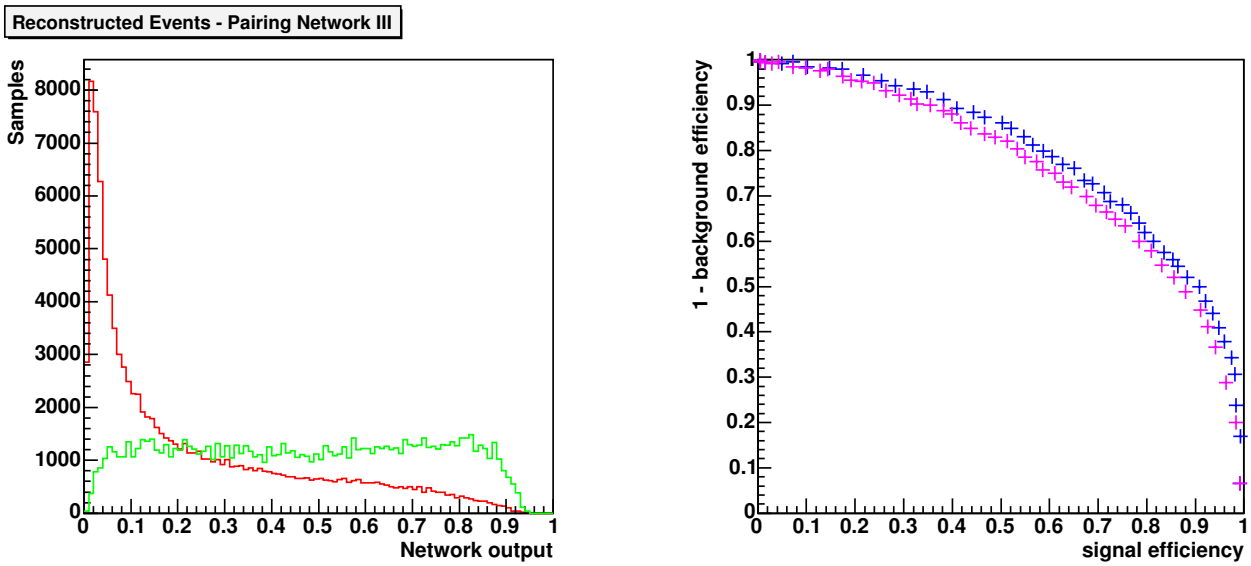

Figure 5.6: *On the left, the final network output using reconstructed background events is shown (correct combinations: green, wrong combinations: red). On the right, the efficiency plots can be seen for reconstructed background events (pink) and for reconstructed signal events (blue).*

The application of the neural network on background events leads to a similar distribution of the network output in the same range as for signal events. Thus, the pairing network can not be used for rejecting background events based on the network output. The efficiency plots are also almost identical for signal and background events, leading to a similar number of correctly paired events. In fact, the correct jet combination is found for 31.6% of the background events compared to 33.4% of the signal events.

Application of Cuts

By using a threshold value on the network output for accepting events, this result can be further improved and the number of correctly paired events increased. But this again happens at the cost of the overall number of events (see table 5.4):

Threshold	0.0	0.5	0.6	0.7	0.8	0.85
Correct Pairings	31.6%	32.4%	32.8%	34.1%	37.2%	38.8%
Efficiency	100.0%	90.5%	81.7%	66.0%	38.4%	21.4%

Table 5.4: *Observing only events whose best combination has a better network output than the threshold. The percentage of correct pairings for all events considered and the overall percentage of events considered is shown.*

The background events behave similar to signal events with increasing the threshold value. Considering these numbers, the recommended approach for the further analysis is to take the best jet matching the neural network delivers for each event as correct combination and continue the analysis with all events. As mentioned above, large thresholds to improve the jet matching would yield an insufficient number of remaining events.

After several years of data taking at CMS this approach could be reconsidered taking the large luminosity and thus the available number of signal events into account. If this will be done, it is important to verify that the rejection of events does not lead to biased remaining events. To check whether this is happening, distributions of certain variables can be compared before and after events have been discarded, in order to determine whether their shapes have changed. Here, the invariant Higgs mass and the invariant $b\bar{b}$ mass are examined for signal and background events, respectively (fig. 5.7). The Higgs mass is chosen since the main purpose of CMS is to determine its value. If its distribution would get shifted, the resulting reconstructed Higgs mass would not represent the true Higgs mass. The $b\bar{b}$ mass is also inspected since it is the equivalent of the Higgs mass for background events. Its shape should not change either for the same reasons as the Higgs mass.

As shown in these figures, a threshold for accepting events does not alter the shape of the Higgs and $b\bar{b}$ mass. A comparison of the generator and the reconstructed Higgs mass after the pairing has been done, shows that there will be no clear peak. This results due to the fact that more than half of the events have been paired incorrectly, which leads to a smearing of the Higgs mass towards values. In case of a wrong jet matching most often one of the b jets from the top decays is used for reconstructing the Higgs boson instead of its more energetic daughter b jet, resulting in a smaller Higgs mass.

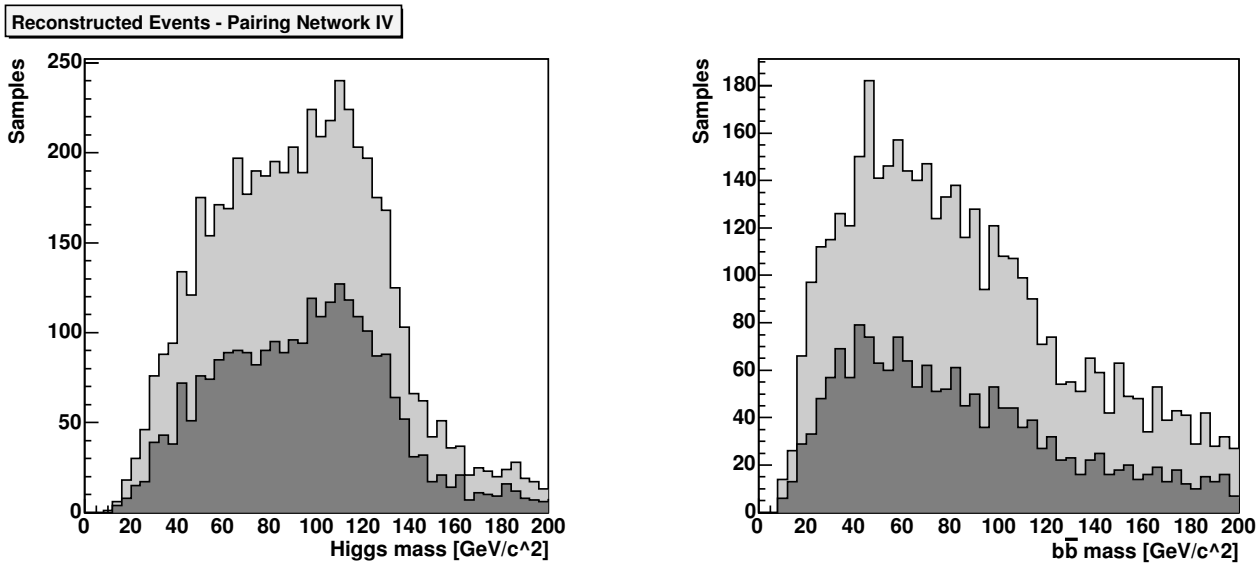

Figure 5.7: *Distributions of the Higgs mass (left) and of the* $b\bar{b}$ *mass (right), calculated with reconstructed events, before (light) and after (dark) rejecting events with a best network output for the jet pairing below 0.8.*

5.6 Comparison of Neural Network Packages

The relevant analyses in this thesis are all done with the neural network package NeuroBayes® by $<$ Phi $-$ T $>$. In addition the performance of the `TMultiLayerPerceptron` neural network class of ROOT is also evaluated as a free alternative to the commercial NeuroBayes® package. A common interface for both neural networks has been developed in the scope of this thesis and is explained in appendix A.

Both neural networks are in principal multilayer feed-forward networks with a backpropagation learning algorithm. The major difference is that NeuroBayes® deploys massive preprocessing to simplify the input patterns and intelligent pruning for improving the network structure. On the other hand, the ROOT network just normalises the input variables. Thus one expects that NeuroBayes® will perform better on strongly correlated or otherwise suboptimal input data compared to the `TMultiLayerPerceptron` class.

For this comparison both neural networks receive the same input patterns and the same settings for training and testing. Since `TMultiLayerPerceptron` does not have the same preprocessing capabilities as NeuroBayes®, these settings are only applied to the latter network. The network comparison is done on reconstructed events instead of generator events since they are more important for the analysis.

As shown in the comparison diagram (fig. 5.8) NeuroBayes® is able to separate signal from background patterns. However the separation is not perfect. On the other hand, the `TMultiLayerPerceptron` completely fails at this task. Consulting the correlation matrix of the input variables in appendix B explains this result quite well, since most of them are strongly correlated, a condition that `TMultiLayerPerceptron` apparently cannot handle well.

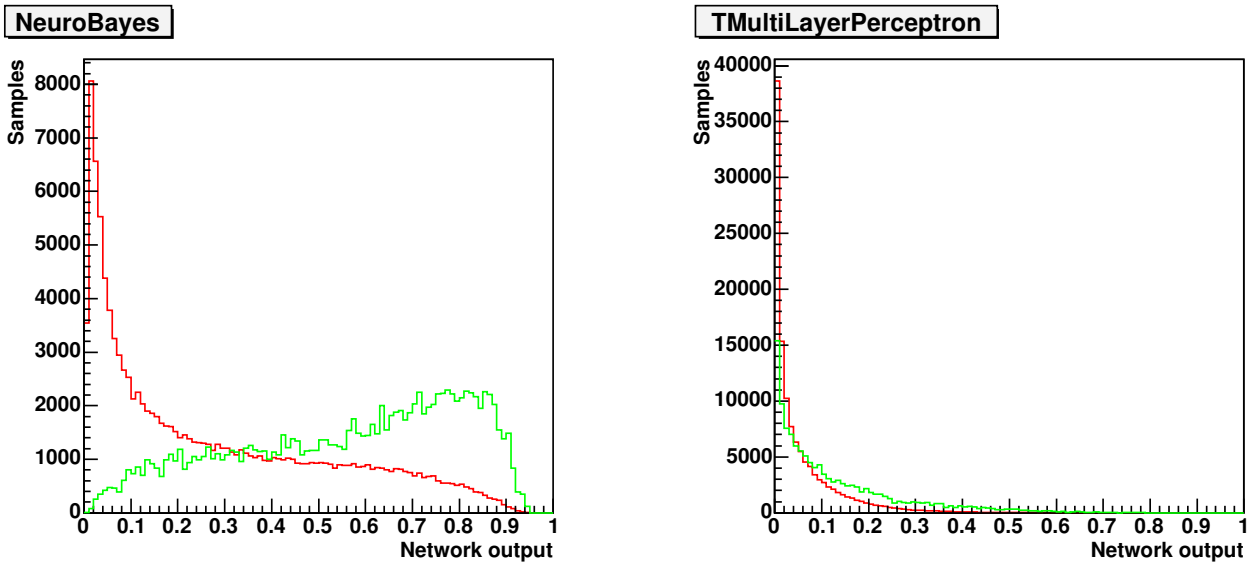

Figure 5.8: *The output of the network for the pairing analysis is shown with signal samples in green and background samples in red. The number of signal samples has been scaled to the number of background samples. On the left, the results of $NeuroBayes$® are shown. On the right, the TMultiLayerPerceptron output is displayed.*

5.7 Comparison to the Likelihood Analysis

The likelihood analysis implemented by Alexander Schmidt [1] delivers about the same results as the analysis of the pairing problem with a neural network – 35% compared to 33%. However his analysis considers all jets in an event whereas the analysis with the neural network is limited to only match the four interesting b jets correctly, the other ones are not considered.

A quick feasibility study at the end of the diploma thesis showed that the current neural network is unable to handle all jets. The sheer number of possible combinations leads to uniformly distributed network output values instead of a clear maximum for the correct jet matching, since many combinations are almost correct. Most frequently, the correct combination is found within the top five ones, but using all of them would decrease the performance again.

The deficit of the neural network could be compensated by using the likelihood output as one input for the network along with further input variables. Thus the neural network can exploit the good separation capabilities of the likelihood analysis and further improve it by using the information inherent to the additional variables. This approach is a future task and not part of this diploma thesis. Until then it is proposed to use the output of the likelihood analysis for the pairing task. This proposal is supported by the fact that the free neural network class `TMultiLayerPerceptron` included in ROOT performs even worse than the commercial NeuroBayes® package, as shown in the previous section.

Background Suppression

Only few events generated in a proton-proton collision at the LHC are actually of interest. After an initial preselection by the trigger system (see chapter 1.2.4), it is the task of the respective analysis software to separate these signal events from all unwanted background events creating a clearer data sample for the further analysis.

This thesis focuses on the $t\bar{t}H$ signal channel, which has several prominent background channels like $t\bar{t}b\bar{b}$, $t\bar{t}jj$ and $t\bar{t}Z$. All of them feature similar final states like the signal channel and therefore cannot be suppressed easily. In particular, the background channel $t\bar{t}b\bar{b}$ is studied in this thesis. This poses a challenge for traditional analysis software, since it has the exact same final state partons with similar kinematics as the signal channel. Thus, a neural network is used to take advantage of as much information as possible to best separate the $t\bar{t}H$ signal events from the $t\bar{t}b\bar{b}$ background.

The signal and background data sets used for the analysis in this chapter are the same as described and used in the previous chapter for the jet pairing task. The same nomenclature is applied for denoting the individual jets of an event.

6.1 Description of the Analysis Methods

The analysis for the background suppression is again divided into two phases, the training and the performance test of the neural network. In both phases, signal and background events are passed to the appropriate training or testing method in an arbitrary order. Here, the whole decay tree is reconstructed from the final state particles in each event. At this stage, the true particle identity from the generator is used. Subsequently, the necessary variables for the neural network are calculated and the input pattern is passed to the network, either for training or for evaluation.

The same number of signal and background patterns is passed to the neural network. Therefore the same weight is applied to both. Another approach would be to pass a number of signal and background patterns according to their expected real number in the experiment to the neural network and weight them accordingly (e.g. $\text{weight}_{\text{sig}} = 1$, $\text{weight}_{\text{bkg}} = \frac{\text{signal events}}{\text{background events}}$). This approach is not attempted in this thesis.

The $t\bar{t}b\bar{b}$ background events do not contain a Higgs boson decaying in two b quarks. Therefore the two jets with the highest b probability[1] not coming from a top decay are used instead as substitute for this Higgs decay chain. They form a system called $b\bar{b}$. In contrast to generator events, where exactly two additional jets exist, usually more than two jets can be found in reconstructed events.

6.2 Analysis of Generator Events

A first study of the $t\bar{t}H$ channel at CMS was made in [2]. Here, it was stated that the $t\bar{t}b\bar{b}$ background channel will pose the most problems for the analysis. The other two channels can be effectively suppressed by the preselection described inside this note. There are several variables mentioned that could turn out to be useful in discriminating the $t\bar{t}b\bar{b}$ background from the $t\bar{t}H$ signal. They are used as input variables for a first performance check of the neural network with generator events.

The Higgs boson in the signal event is considered to be coming from the decay of a virtual t^*. As explained in chapter 2.3.1, this is only true for one of the two possible Higgs production processes for this channel, with the other process being a top-top fusion. Since both processes are inseparable, it is not clear how much contribution comes from which one. Furthermore it can not always be determined which of the two reconstructed top quarks has to be associated to the Higgs boson in order to reconstruct the virtual t^*. The top quark, that is nearest in angle to the $b\bar{b}$ named t_{near} is used for this purpose.

The $b\bar{b}$ in the background events generally comes from the decay of a virtual gluon as opposed to the Higgs boson in a signal event. These b jets are not strongly associated to the top quarks of the decay, since any intermediate coloured particle could have radiated the gluon. Because of the stronger association of the Higgs boson to one of the top quarks and the fact that the spin of the Higgs and the gluon is different (spin-0 and spin-1, respectively), the angular and momentum distributions of the t^* and the $b\bar{b}$ might be exploited to discriminate signal from background events.

The variables chosen in [2] due to their described physical properties are listed below. Plots of their distributions for reconstructed events can be found in appendix B:

1. **t^* mass**:
   Invariant mass of the virtual t^* that radiates the Higgs boson in case of a signal event. It is generated by combining the $b\bar{b}$ with the reconstructed top quark t_{near} nearest in angle.

2. $\measuredangle(\mathbf{b\bar{b}}, \mathbf{t_{near}})$:
   The actual angle between the $b\bar{b}$ system and the t_{near} quark.

[1] The b probability is a value generated by the CMS reconstruction software that indicates the probability of a jet containing a b quark. It is a logarithmic variable and higher values denote a greater probability.

3. $\mathbf{\Delta r(b\bar{b}, t_{near})}$:
   Rapidity difference between the $b\bar{b}$ system and the associated top quark.

4. $\mathbf{\Delta R(b\bar{b}, t_{near})_1}$:
   Distance in phase space using pseudo-rapidity for the calculation.

5. $\mathbf{\Delta R(b\bar{b}, t_{near})_2}$:
   Distance in phase space using rapidity for the calculation.

6. $\mathbf{\Theta(b, b\bar{b})}$:
   The angle Θ between the b quark and the $b\bar{b}$ direction in the $b\bar{b}$ restframe.

The second to fifth variables are all similar and attempt to exploit the information inherent to the nature of the Higgs radiation from the top quark. The angle Θ tries to utilise the different angular distributions of the Higgs boson and the virtual gluon, i.e. of the mother particle of the $b\bar{b}$ in a signal or background event. It is only listed for the sake of completeness since this variable does not contribute to a better signal to background discrimination as stated in [2]. Its distribution shows no distinct difference for signal and background events and it does not have got relevant correlations to the other variables. Therefore it is omitted from the neural network.

The training is performed on 60,000 signal and 60,000 background events. The resulting separation performance of the background suppression neural network using variables 1 to 5 calculated from generator events as input pattern is shown in fig. 6.1 below.

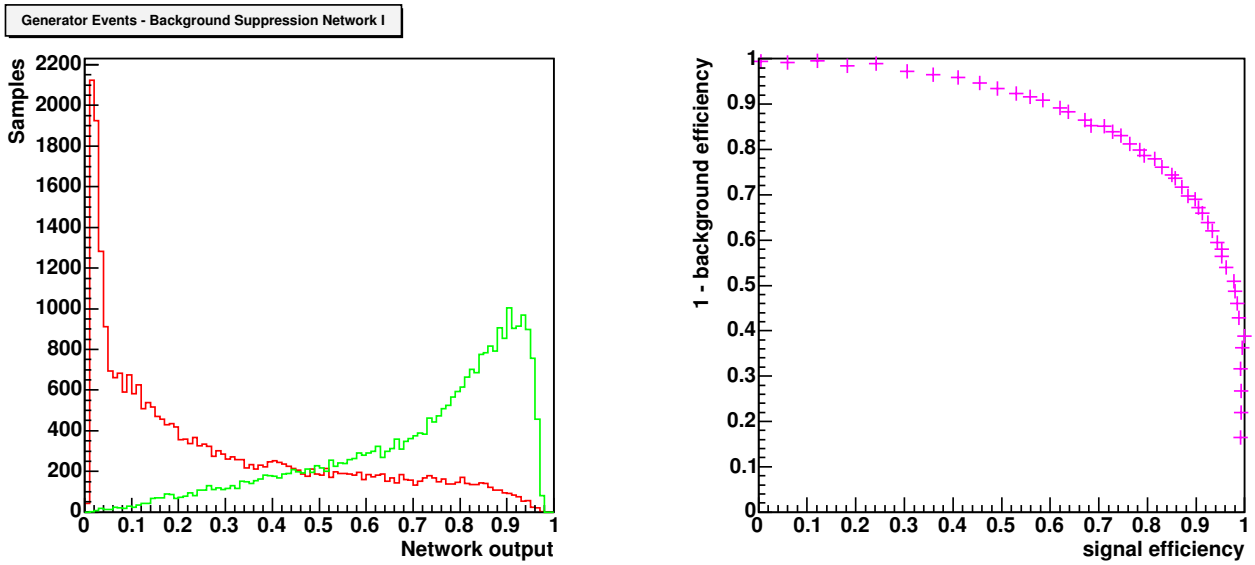

Figure 6.1: *On the left-hand side, the network output using generator events is shown (signal events: green, background events: red). On the right, the efficiency plot for this trained neural network is displayed.*

The invariant mass of the virtual t^* is the primary discriminating variable of the network. Its distribution is depicted in fig. 6.2 to the right. This variable depends on the mass of the Higgs boson since the $b\bar{b}$ system corresponds to the Higgs boson in case of signal events. This is a problem, since the Higgs mass is to be determined by the analysis and not just a parameter passed to it. If an input variable is dependant on the Higgs mass the network will learn this mass as defined by the event generator and will be more sensitive to such events, even if the real mass is different. Furthermore the distribution of the invariant t^* mass is disadvantageous for the training of the network. Masses below about 300 GeV/c^2 generally only occur for background events. Therefore, the network will use this input variable as the sole discriminator in this mass range, disregarding other available information. Examining the distribution of the invariant $b\bar{b}$ mass for background events (fig. 6.3), it is evident, that its shape is shifted to higher masses in the direction of the Higgs mass used by the event generator if the the t^* mass is used as input variable. This bias on the background events remaining after all events with a network output of less than 0.8 have been rejected is an unwanted behaviour. Therefore the invariant t^* mass is not a good input variable for the neural network.

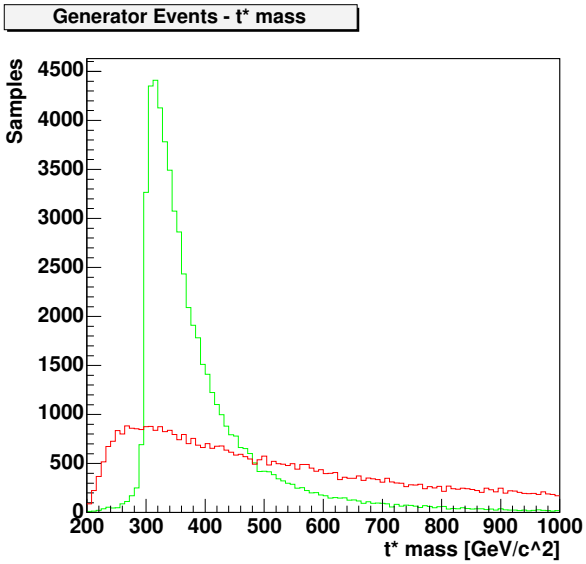

Figure 6.2: *Distribution of the invariant* t^* *mass (signal: green, background: red).*

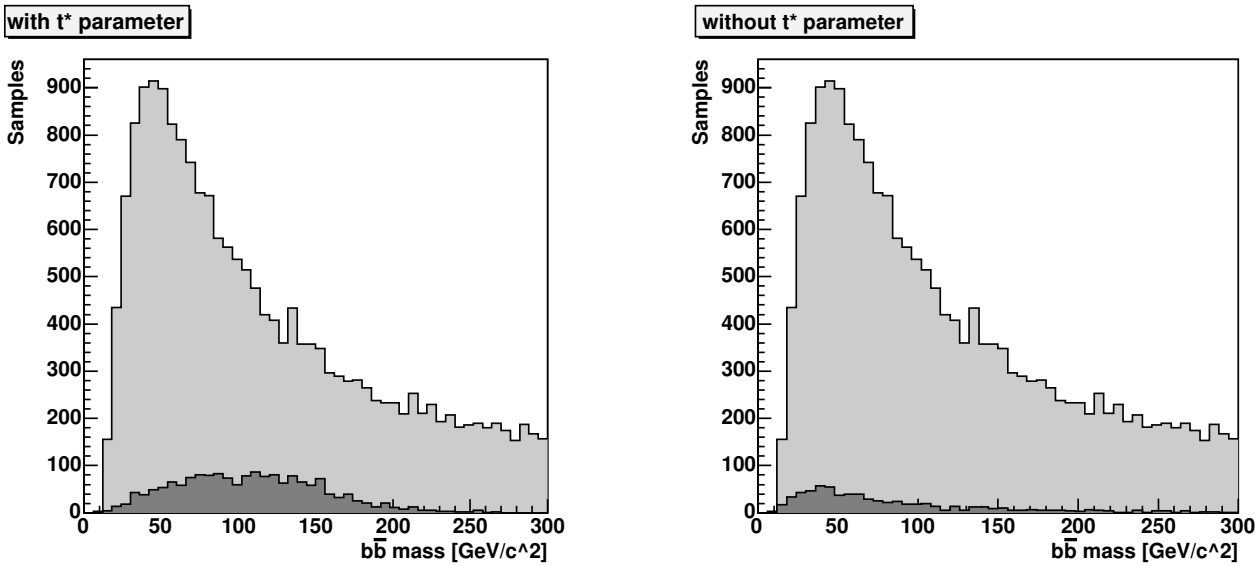

Figure 6.3: *On the left, the distribution of the* $b\bar{b}$ *mass for background events is shown, using a network that was trained with the* t^*mass *variable. The light area comprises all background events and the dark area only events with a network output greater than 0.8. On the right, the same is shown for a network without the* t^* *variable.*

The performance of the neural network trained with the remaining four variables is shown in fig. 6.4. As expected its ability to separate signal from background events is worse than before. However, the resulting network performance is still good.

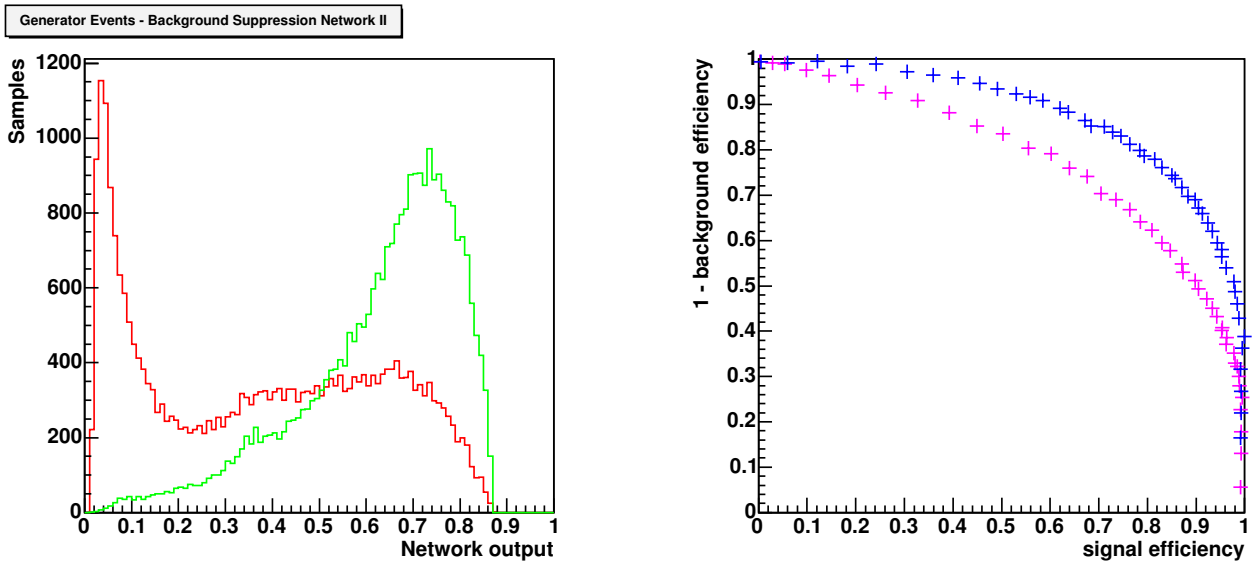

Figure 6.4: *On the left-hand side, the network output using only four input variables is displayed (signal events: green, background events: red). On the right, the efficiency plot for the current neural network (pink) and for the network with the invariant mass of the* t^* *(blue) is shown.*

6.3 Analysis of Reconstructed Events

After a set of variables producing a good separation on generator events has been chosen for the network, the analysis has to be tested on more realistic data before continuing. For this purpose, the reconstructed events are used for the same analysis as before. Because the true jet identities are still needed only good matched reconstructed events can be used for this purpose. The training is performed on the good matched fraction of the events used in the previous section. The resulting performance of the neural network is depicted in fig. 6.5 on the next page.

As can be seen, the network performance is roughly the same as for generator events even though the actual distribution of the network output appears to be much worse. This result is slightly unexpected, as a more pronounced decrease in the performance like for the jet pairing network should have happened. The reason for this behaviour could be the fact that the events for training and testing are already biased since only good matched events are taken. Unfortunately at this stage there is no possibility to verify or falsify this assumption by not using good matched events, since the neural network needs to know the identity of the particles for its training.

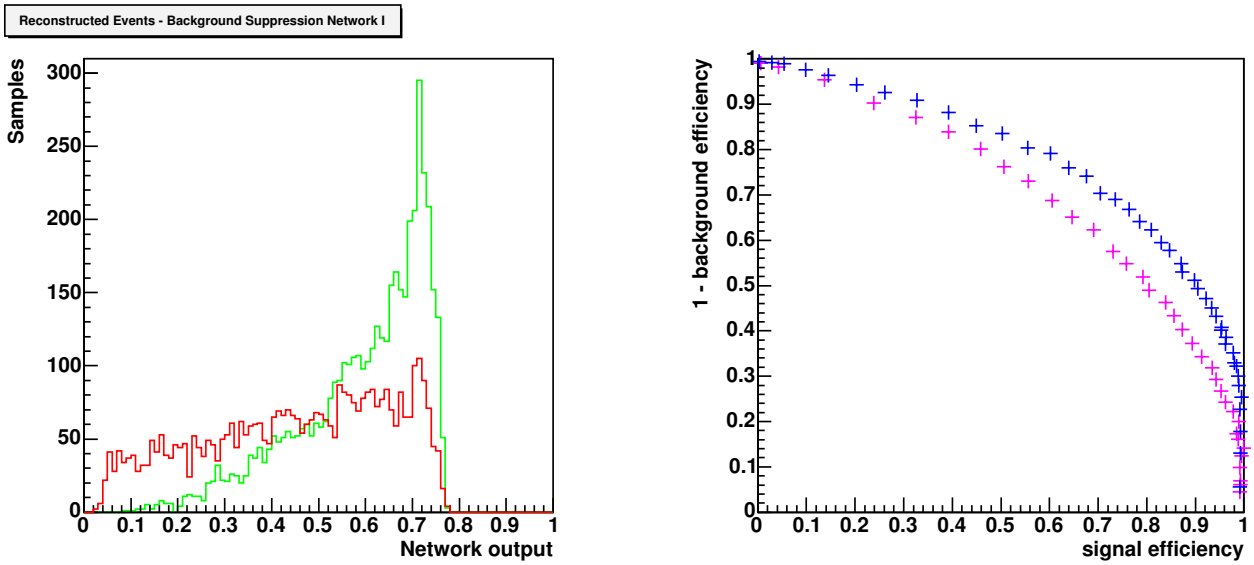

Figure 6.5: *On the left, the network output using reconstructed events is shown (signal events: green, background events: red). On the right-hand side, the efficiency plot using reconstructed events (pink) and generator events (blue) can be seen.*

6.3.1 Improving the Analysis

In order to further improve the analysis, additional variables are added to the neural network to increase its discriminating capabilities. The following three variables seem to be the most promising ones:

1. $\Delta R(b_{higgs1},\ b_{higgs2})$
2. $|\eta(b_{higgs1})|$
3. $|\eta(b_{higgs2})|$

They try to further utilise the information inherent to the differences in the production of the $b\bar{b}$ system for signal and background events. The distance in phase space ΔR between the two b jets of the $b\bar{b}$ system is generally smaller for signal events than for background events, i.e. the Higgs decay products leave the interaction in closer vicinity. The η distribution is uniform for the two background b jets with a peak in the very forward direction, whereas the daughter jets of the Higgs boson rather travel transversal to the beam line.

These three variables have been primarily chosen because their distributions exhibit noticeable differences for signal and background events as described above, thus an improvement in the performance of the neural network can be expected. Their distributions are shown in appendix B, separated for signal and background events.

In addition, some technical improvements similar to those mentioned in chapter 5.5.1 have been applied to the neural network for the final analysis run on reconstructed events.

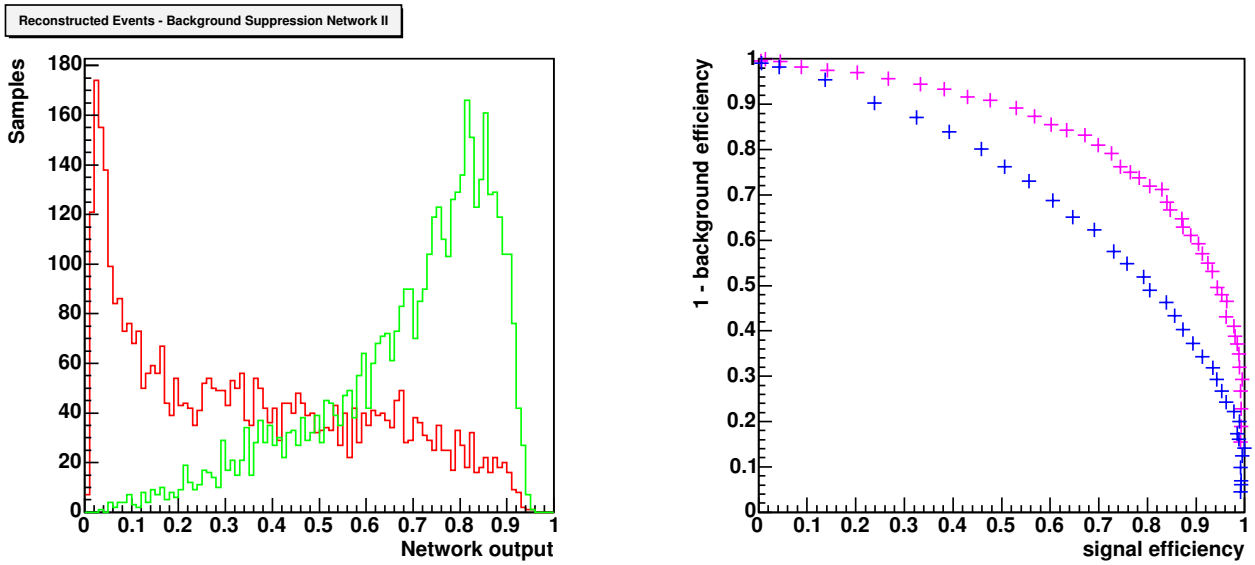

Figure 6.6: *On the left, the network output using reconstructed events is shown (signal: green, background: red). On the right, the efficiency plot for the improved network (pink) and for the previous one (blue) are presented.*

Results

The network performance has been increased by a large amount due to introducing the afore mentioned improvements (fig. 6.6). Taking a signal efficiency of 0.9 for example leads to a corresponding background efficiency of 0.4, decreasing the number of background events by more than 50%. The signal to square root of background ratio after one year of data taking at CMS in the low-luminosity phase yields to:

$$\frac{N_S}{\sqrt{N_B}} \approx 11.7 \text{ instead of } \approx 9.2 \text{ previously.}$$

The working point for the following analysis was chosen at a signal efficiency of 0.9 which is directly correlated to rejecting all events with a network output of less than 0.4. The reconstructed Higgs mass of the remaining events (signal as well as background) is plotted in fig. 6.7 along with its true distribution from all signal events used. The plots have been scaled to the same number of events for an easier comparison.

Here, the Higgs mass is reconstructed quite well. The peak is at the same mass value for both distributions, only the reconstructed Higgs mass plot features more entries at lower values,

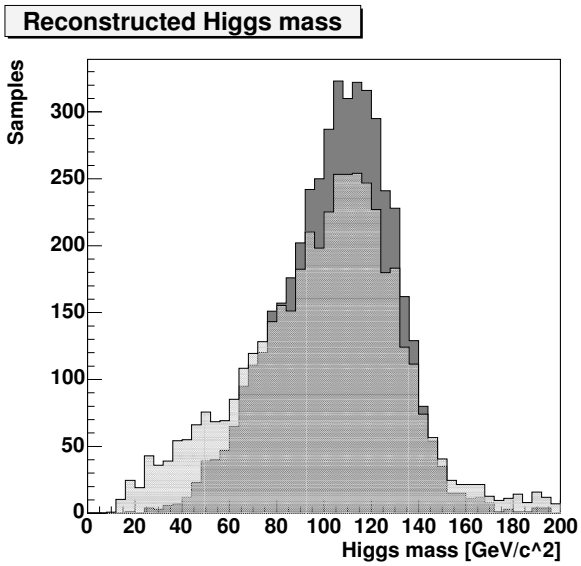

Figure 6.7: *Distribution of the Higgs mass (simulated: dark, reconstructed: light).*

caused by the remaining background events. Using a Gaussian fitting function to approximate the presented curves yields a Higgs mass of $m_{Higgs} = 100.0 \pm 0.5$ GeV/c^2 with a standard deviation of $\sigma_{Higgs} = 30.7 \pm 0.4$ GeV/c^2 compared to the reconstructed values $m_{Higgs} = 105.1 \pm 0.4$ GeV/c^2 and $\sigma_{Higgs} = 23.0 \pm 0.3$ GeV/c^2 in the signal events after the detector simulation. The mean of this Higgs mass does not equal $m_{Higgs} = 120$ GeV/c^2, set by the event generator, due to detector and reconstruction effects. For example, not the entire jet energy was measured by the detector hardware or the jet reconstruction software has not taken into account a large enough area of the calorimeter for a correct jet energy measurement.

6.4 Comparison of Neural Network Packages

The pairing neural network presented in the previous chapter has been implemented with the NeuroBayes® package and the `TMultiLayerPerceptron` class of ROOT. Both variants have been trained on the same data and their performances have been compared, resulting in a clear recommendation for NeuroBayes®. The same procedure is repeated for the background suppression network using the same methods and conditions as for the pairing network.

As can be seen in fig. 6.8, both neural network packages are capable of solving this separation task. Due to its more sophisticated methods, NeuroBayes® has some advantages and achieves a slightly better result than the `TMultiLayerPerceptron`. The discrepancy in the networks' performances for the jet matching and the background suppression task are again explained by the correlations of the input variables. Appendix B shows, that the variables of the background suppression network are only weakly correlated. This is an indication that the neural network provided by ROOT might perform similar to the NeuroBayes® package.

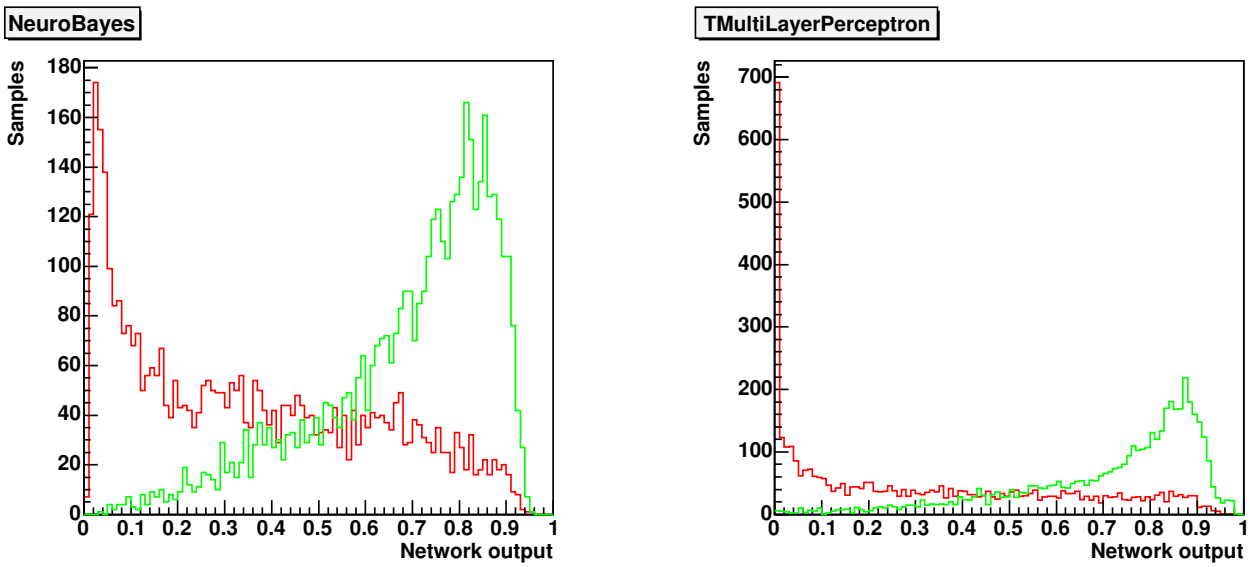

Figure 6.8: *The output of the network for the background suppression analysis is shown with signal samples in green and background samples in red. On the left, the results of $NeuroBayes$® are shown. On the right, the TMultiLayerPerceptron output is displayed.*

6.5 Combination of both Analyses

In the previous chapters, two neural networks have been analysed. The first one is used for identifying the correct jet matching in a given event and the second one separates $t\bar{t}H$ signal events from $t\bar{t}b\bar{b}$ background events. This has been a rather academic task so far, since jet identities supplied by the event generator have been used for the background suppression task. In order to arrive at a more realistic analysis, both networks are now connected, resulting in the analysis chain described below:

1. Event samples are produced by a Monte Carlo generator.
2. These are passed through the CMS reconstruction software and enriched with additional information like the b probability.
3. The jet pairing network and the background suppression network are both trained with the reconstructed data using the jet identities supplied by the event generator.
4. A new event is passed to the background suppression network for classification.
5. The jet pairing network is applied to deliver an estimation of the jet identities of the event.
6. The determined pairing is used by the background suppression network to decide whether an event is a $t\bar{t}H$ signal or a $t\bar{t}b\bar{b}$ background.

The first three steps illustrate the training phase of the neural networks followed by the combination of jet pairing and background suppression in steps four to six, leading to the results presented here.

Results

The analysis chain as described above has been implemented and trained with the good matched fraction of 60,000 signal and 60,000 background events as before. For evaluating the network performance, an independent sample of 8,766 good matched events was used.

The network output of the combined analysis is shown in fig. 6.9. As expected, the results turn out worse since an imperfect jet matching was applied. For receiving a signal efficiency of 0.9 a remaining fraction of 66% of background events has to be accepted. In this case, the signal to square root of background ratio after one year of data taking is 8.4 compared to 10.8 before.

Nevertheless, a working point at a signal efficiency of 0.9 was chosen again in order to reconstruct the invariant mass of the Higgs boson. The resulting mass distribution is shown in fig. 6.10. As described earlier in chapter 5.5.1, the imperfect jet matching done by the pairing network yields a Higgs mass that is shifted to lower values. The same effect occurs here delivering a Higgs mass of $m_{Higgs} = 94.7 \pm 0.8$ GeV/c^2 with an associated standard deviation of $\sigma_{Higgs} = 36.3 \pm 0.5$ GeV/c^2, by using a Gaussian approximation.

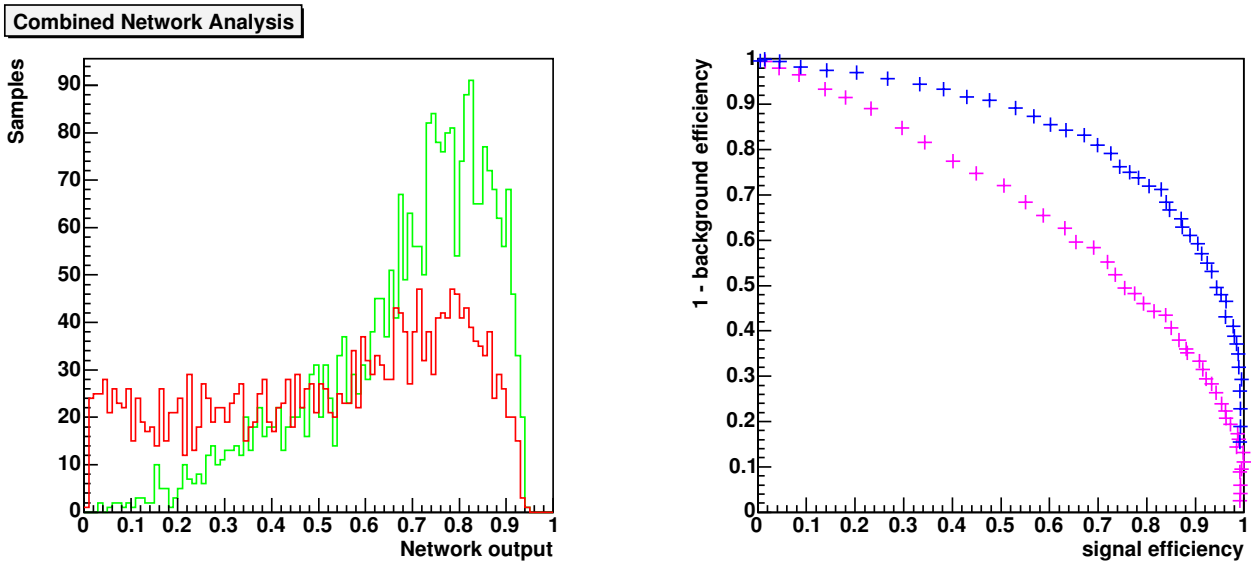

Figure 6.9: *On the left, the network output for the combined analysis is shown (signal events: green, background events: red). On the right-hand side, the corresponding efficiency plot (pink) and for the analysis using jet identities provided by the generator (blue) is shown.*

It is interesting to note, that this distribution still has a peak at the same value like the Higgs mass distribution for the signal events even though it is superposed by an additional peak of the same order of magnitude at about 60 GeV/c^2, caused by the incorrectly paired b jets. Also shown as a dashed line in fig. 6.10 is the distribution of the Higgs mass for signal events after the jet matching. It represents the best possible reconstruction that the network analysis (shown in light grey) can achieve. In fact both distributions are similar, therefore the background suppression has been very successful.

Problems

While performing this combined analysis, some problems arose that had to be dealt with and that have an impact on the previous analyses. The most important problem was the existence of physical unreasonable reconstructed jets with $E < p_z$ for several jet matchings. In such a case, e.g. the rapidity cannot be calculated. The cause of this behaviour is a wrong jet combination paired with a wrong missing energy calculation. This problem has not been detected while studying the pairing network since none of its parameters is sensitive to this energy constraint. The perfect jet matching used for the

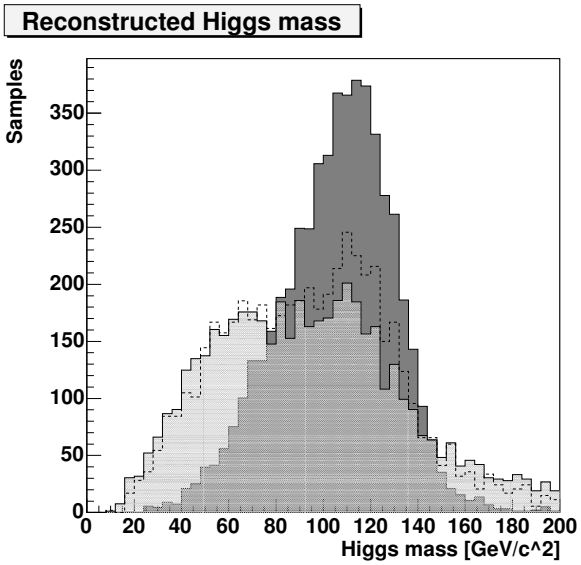

Figure 6.10: *Distributions of the Higgs mass (simulated: dark, after jet matching: dashed, reconstructed: light).*

background suppression network resulted in not detecting this either. To compensate for this behaviour, all such jet pairings are considered as being wrong. This leads to a decrease in the number of useable events by about 50%. The appropriate changes have retrospectively been applied to the other analyses and the results presented in this thesis already reflect them.

Another alternative to deal with this problem is using NeuroBayes®' ability to handle input variables that are not defined for all input patterns. This would allow to also use events for which the rapidity cannot be calculated due to their energy being smaller than the momentum in z direction for all jet matchings. Jet pairings would not automatically be declared wrong because of this energy constraint and the efficiency decrease of 50% could be avoided. This approach seems promising but has yet to be studied.

Conclusion and Outlook

Large physical experiments like CMS at CERN yield a lot of complex data that have to be inspected thoroughly. Powerful analysis tools and methods are required for this task. Neural networks are one possible solution, providing the means for such a data challenge. The purpose of this thesis was to research the potential of these techniques in the field of CMS analyses and to apply them on one concrete problem.

The $t\bar{t}H$ channel with its associated background channel $t\bar{t}b\bar{b}$ was chosen as target of this study. Previous analyses had already identified the difficulties that this channel possesses. In this work, the performance of neural networks was probed with it. In particular the NeuroBayes® package and the `TMultiLayerPerceptron` class of ROOT have been examined.

At first, a network was developed to correctly match the final state particles to the actual partons of the process. This is an important task, since before starting any further analysis the identities of the particles that are examined have to be well known. An existing analysis [1] has been transferred for the usage of neural networks and expanded. The resulting network yields the correct jet pairing for 33.4% of all signal events and for 31.6% of all background events. This is about the same performance as in the previously mentioned analysis. The neural network approach has still to be expanded to include an imperfect identification of b jets and also to consider all jets contained within an event instead of only the jets corresponding to the final state particles. Interfacing the existing likelihood analysis and exploiting its pairing information for the neural network could be useful, too.

Afterwards, another neural network was designed for the actual physics analysis presented in this thesis. It should separate $t\bar{t}H$ signal events from the $t\bar{t}b\bar{b}$ background. Based on an existing CMS note [2], discriminating variables have been identified and a network has been devised. It yields a good performance by rejecting 60% of the background events while still keeping 90% of the signal. It also offers a good reconstruction of the Higgs mass similar to the signal events after the detector simulation. Another promising approach to the background suppression would be using two different neural networks, one aimed at identifying signal events and the other one for the background events.

Combining both neural networks was the final step in the analysis. The pairing network delivers the supposedly correct jet matching used in the background suppression network. The resulting combined network could still perform its discrimination task, within the given scope, but it got evident, that the imperfect jet identification was the main limiting factor for the analysis. This can be seen in the distribution of the reconstructed Higgs mass in the signal events, which

is already shifted to lower values after the jet pairing. Therefore it is recommended to give full attention to improving the jet matching. The proposed analysis chain is depicted in fig. 7.1.

This thesis has shown that neural networks are a useful and powerful tool for analysing high energy physics data in the CMS context that can match or even surpass more traditional approaches. The possibilities of the two developed neural networks have yet to be exhausted. There are still input variables that have to be investigated and technical means that can be further exploited for a better network training.

In addition a framework has been developed within the scope of this thesis in order to obtain independence from the actual underlying network package. An interface for each package applied in this thesis was created, making analyses using either of them easily comparable. In the future this framework could be expanded to include further neural networks, thus always using the best one appropriate for the current task. Furthermore, additional analysis functionality could be integrated into the framework, endowing weaker networks with more performance.

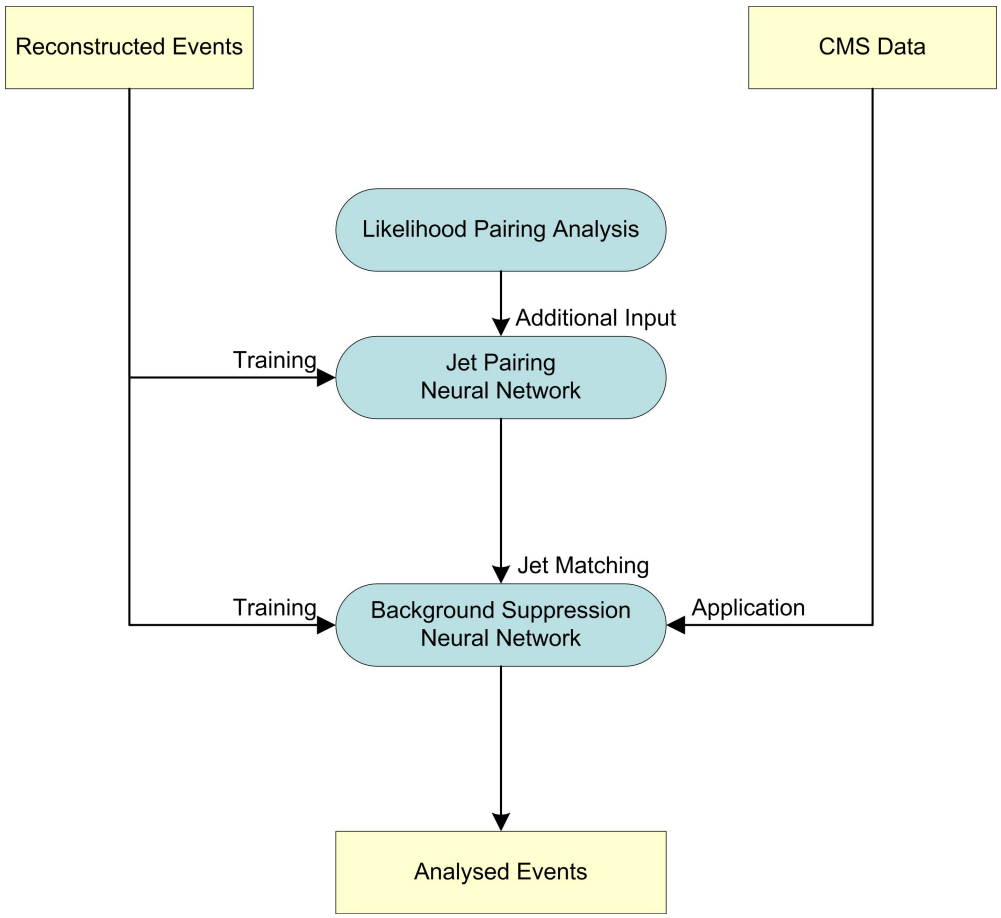

Figure 7.1: *Proposed analysis chain for the suppression of background events. The neural networks for jet matching and background suppression are trained with reconstructed Monte Carlo events as input. The first one also receives additional information from the likelihood pairing analysis. In application, the background suppression network receives real CMS data to analyse and then employs the pairing network to deliver jet identifications for real detector events.*

Framework for Neural Network Packages

There is a great variety of neural network packages, free as well as commercial, e.g. NeuroBayes®, `TMultiLayerPerceptron`, JETNET, etc. . Each offers a different user interface. Transferring an analysis from one of these neural network packages to another one can be quite a challenge because of the diversity of their interfaces. Thus an easy comparison of multiple neural networks for a certain analysis task becomes impracticable at the very least.

To solve this problem, a neural network framework has been devised in the scope of this diploma thesis to easily wrap the functionality of different neural network packages under the same interface, thus unifying the access to the analysis potential of these tools. This framework represents a new layer of abstraction between the actual analysis code and the neural networks, rendering the analysis independent of the underlying neural network package.

Apart from the abstract base class (`PaxNeuralNet`) that defines the principal functionality all network wrappers have to provide, two concrete classes have been implemented. They wrap the functionality provided by the NeuroBayes® package (`PaxNeuroBayesNet`) and the `TMultiLayerPerceptron` class of ROOT (`PaxRootMLP`). Some further improvements, especially for the NeuroBayes® interface are planned but have not be implemented within the scope of this thesis.

The structure of the neural network framework is shown in the UML[1]-diagram in fig. A.1.

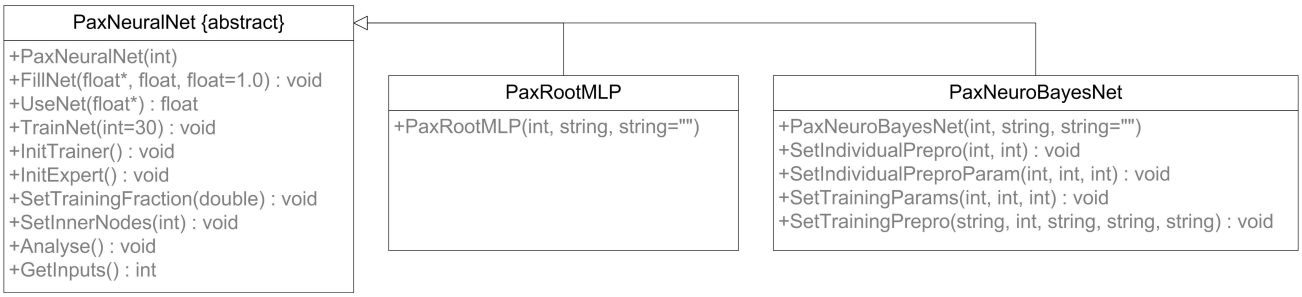

Figure A.1: *UML diagram for the neural network framework. An abstract base class (PaxNeuralNet) gets derived by two concrete classes that wrap the functionality of $NeuroBayes^{®}$ (PaxNeuroBayesNet) and TMultiLayerPerceptron (PaxRootMLP), respectively.*

[1] Unified Modeling Language

A.1 PaxNeuralNet Class

This abstract class acts as the general foundation for all other network classes that will be derived from it. All functionality that has to be provided by every network wrapper is defined here.

Network training and the subsequent application of the trained network using this framework usually takes place in the following steps: At first the desired neural network class is instantiated. For training purposes the trainer is initialised, followed by adjustments to the preprocessing and training parameters, if needed. After this setup step, the input patterns are passed to the network which is in turn trained. Having finished the training the expert mode of the neural network has to be initialised and then new samples can be passed to the network for analysing.
The methods responsible for the outlined procedure are introduced below:

- **PaxNeuralNet(int inputs):**
  A general constructor, receiving the number of network inputs.
- **virtual void InitTrainer():** (abstract)
  This method has to be called before anything concerning network training takes place. Variables get initialised, classes instantiated and files created necessary for training.
- **virtual void FillNet(float* input, float target, float weight=1.0):** (abstract)
  Here input patterns can be passed to the neural network for training. Along with the actual pattern its designated target value and optionally a weight for this event get passed.
- **virtual void TrainNet(int iterations=30):** (abstract)
  This method starts the training process with the specified number of training iterations. At the end of the training a file containing the resulting network should be written to disk. This duty rests with the concrete classes.
- **virtual void InitExpert():** (abstract)
  Before the usage of a neural network this method has to be called. It restores the network description from file and initialises the needed variables.
- **virtual float UseNet(float* input):** (abstract)
  This method uses the trained neural network with the passed input pattern. The network output is then returned by this function, scaled to a region of [0, 1].
- **virtual void SetTrainingFraction(double fraction):** (abstract)
  With this method the fraction of input samples used for training can be defined. The remaining input samples will be used for performance tests during the training phase if the underlying neural network supports this functionality.
- **virtual void SetInnerNodes(int inner):** (abstract)
  Here the number of neural nodes in the hidden layer is defined. Currently only a three-layer network is supported, thus only one hidden layer exists.

- **virtual void Analyse():** (abstract)
  This method starts an analysis of the training and the resulting network. What happens exactly is subject to the concrete derived class.

- **int GetInputs():**
  Returns the number of network inputs defined for this network.

Notice: This interface was designed with three-layer feed-forward neural networks in mind.

A.2 PaxRootMLP Class

This class wraps the functionality of the `TMultiLayerPerceptron` class of the ROOT framework. It retains and implements all methods defined by its base class `PaxNeuralNet` and adds one public method, explained below.

Upon completion of the network training, the network structure is saved in a directory named *results*. This information is loaded when the expert part of the neural network gets initialised. The `Analyse()` method – using the `TMLPAnalyzer` class for its purposes – also stores a ROOT file in the *results* directory.

The network output of the `TMultiLayerPerceptron` neural network is not confined to the range that the training target values set. In order to solve this unwanted behaviour, the neural network of ROOT was modified: the output functions of the output neurons were set to a sigmoid function instead of a linear function. Therefore the `TMultiLayerPerceptron` class and its associated classes had to be extracted from the ROOT package and compiled on their own – with this change built in.

- **PaxRootMLP(int in, string file, string directory=""):**
  This constructor expects the number of input variables the neural network is going to have, the name for the analysis- and network description-file and optionally a directory name, where the *results* directory will be created.

A.3 PaxNeuroBayesNet Class

The `PaxNeuroBayesNet` class is a wrapper for the neural network provided by the NeuroBayes® package. In addition to the functionality given by the base class, several other methods are available to utilise the preprocessing abilities of the package and to steer the training.

During the training a file containing the Expertise as well as a ROOT file containing various analysis data about the network and the training are written to the *results* directory. The `Analyse()` method can then be used to convert this information into a more visual postscript file, stored in the *results* directory, too. The functionality of the `Analyse()` method is provided by an external compilation of functions in the file `NbAnalysis.cc`, originally developed at the IEKP by Claudia Lecci.

- **PaxNeuroBayesNet(int in, string file, string directory=""):**
  This constructor expects the number of input variables the neural network is going to have, the name for the analysis- and network description-file and optionally a directory name, where the *results* directory will be created.

- **void SetIndividualPrepro(int var, int prepro):**
  The individual preprocessing method for each input variable can be set by this method. In general 34 and 14 are good settings for continuous variables with or without a delta-function, respectively. All preprocessing possibilities and their assigned number can be found in [64].

- **void SetIndividualPreproParam(int var, int paramNum, int param):**
  This method is used to force the preprocessor in the decorrelation step to especially attempt to decorrelate certain variables. Currently this method is exactly the same as the `SetIndividualPreproParameter` function in NeuroBayes®, leading to a somewhat complicated employment. It is planned to be improved in the future.

  In order to tell the preprocessor to decorrelate an input variable A with some other input variables B_i the following method calls are required: On the first call the number of input variables B_i is passed to the method as third parameter and on the subsequent calls the individual variables are passed. The first parameter is always variable A and the second parameter is a counting parameter incremented by one for each method call with the same input variable A, starting with zero.

  Due to the internal structure of the preprocessor, variable A has always to be larger than the B_i pertaining to their numbering. In addition the variables B_i are numbered starting with 2 instead of 0; this does not affect the afore mentioned constraint.

- **void SetTrainingParams(int speed, int learn, int epoch):**
  General training parameters can be set by this method, namely the training speed, the learn rate and the number of training epochs before the weights are updated. The speed value should be chosen to be higher than the value of the learn rate. Good values for the training, figured out in the course of this thesis, encompass a training speed of 300, a learning rate of 150 and a weight update after 100 training epochs. If the network training aborts, the training speed and learning rate have to be reduced.

- **void SetTrainingPrepro(string task, int pre, string reg, string loss, string shape):**
  This method is used to alter further network settings. The network task (DENSITY, REGRESSION; default: CLASSIFICATION), the preprocessing shape for all variables (default: 11) and the regularisation method (OFF, REG, ASR, ALL; default: ART) can be changed. The error function (QUADRATIC; default: ENTROPY) can be chosen and direct connections between the input and output layer can be toggled (INC, TOT; default: OFF). A detailed explanation of the parameters is given in [64].

A.4 Applying the Neural Network Framework

To apply the neural network framework at the IEKP the following steps have to taken:

ROOT

ROOT has to be installed in order to use the `PaxRootMLP` class; it is also required for NeuroBayes®. In general a ROOT environment is set up at every desktop machine. If not, this can be easily done by adding the following environment variables:

```
setenv ROOTSYS /usr/users/software/root-versions/4.02.00
setenv PATH "$ROOTSYS/bin:"$PATH
setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":"$ROOTSYS/lib"
```

When compiling a program using ROOT functionality, the following libraries have to be added:

```
LIBS += -lMLP -lTreePlayer
```

NeuroBayes®

To use the `PaxNeuroBayesNet` class, NeuroBayes® has to be prepared for each machine it will run on. At first a local software license for the NeuroBayes® package has to be acquired from $<$ Phi $-$ T $>$ – the IEKP is provided with free licenses. The license path and the installation path have to be added to the environment variables.

```
setenv PHIT_LICENCE_PATH ~phit/Licences
setenv NEUROBAYES /usr/users/kerzel/NeuroBayes/NB-C++/v1_3
setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":"$NEUROBAYES/lib"
```

When compiling a program using this functionality, the following libraries have to be added:

```
LIBS += -L$(NEUROBAYES)/lib -lNeuroBayesRoot -lNeuroBayesTeacherCPP
```

Neural Network Framework

After ROOT and NeuroBayes® have been set up, the neural network framework can simply be used by including the appropriate header files in the analysis code and then compiling and linking the source files. If the altered `TMultiLayerPerceptron` is going to be used, these files have to compiled and linked as well.

Notice:

When using `PaxNeuroBayesNet`-specific methods for preprocessing or training adjustments, they should be capsuled in an if-query so that this code only gets executed when really using a `PaxNeuroBayesNet` object.

Network Parameters and Input Variables

This chapters is intended to give an overview of the training parameters and the input variables used for training both neural networks developed in this thesis. The distributions and correlations of the input variables to each other and to the target value are shown in the following sections as well as their preprocessing parameters. In addition the general settings for the neural networks are listed.

B.1 Jet Pairing

The jet pairing network is intended to identify the correct matching of final state particles to the original partons of a process. This is done by returning a probability value for each possible combination passed to the network. The matching with the highest probability is then chosen as true for the following analysis.

Network Settings

The following settings – deviating from the default settings – have been applied to the NeuroBayes® network to adjust the training:

Number of inner nodes:	20
Training iterations:	200
Training fraction:	1.0
Training speed:	150
Learn rate:	100

The training has been performed on the good matched fraction of 60,000 signal events. For each event 12 possible jet matchings have been generated and for each of them one or two input patterns have been passed to the neural network, depending on the possible reconstructions of the missing energy. Particulars about the jet pairing network are explained in chapter 5.

Variables Used

The following variables are used as input for the training of the jet pairing network. The hadronically decaying W boson is labelled $W_{hadronic}$ and its associated b and t quark are named $b_{Whadronic}$ and $t_{Whadronic}$. A corresponding denotation is chosen for the leptonically decaying W boson ($W_{leptonic}$) and its associated bottom ($b_{Wleptonic}$) and top quark ($t_{Wleptonic}$). The heaviest decay product of the $W_{hadronic}$ deacy is called q_1 and the muon μ is a daughter of the $W_{leptonic}$ boson. The value $t_{theory} = 175\ GeV/c^2$ denotes the current rough assumption of the top mass [14]. The applied preprocessing setting is indicated in the brackets.

1. $\Delta R(W_{hadronic},\ b_{TWhadronic})$ (14)
2. $\Delta R(\mu,\ b_{TWleptonic})$ (14)
3. $\Delta R(b_{higgs1},\ b_{higgs2})$ (14)
4. $\Delta R(b_{Wleptonic},\ t_{Whadronic})$ (14)
5. $\Delta R(W_{leptonic},\ b_{Whadronic})$ (14)
6. $\Delta R(b_{Whadronic},\ t_{Wleptonic})$ (14)
7. $\Delta R(W_{hadronic},\ b_{Wleptonic})$ (14)
8. $\Delta R(q_1,\ b_{Wleptonic})$ (14)
9. $\Delta R(t_{Whadronic},\ t_{Wleptonic})$ (14)
10. $(t_{Whadronic} - t_{theory})$ mass (14)
11. $(t_{Wleptonic} - t_{theory})$ mass (14)

Distributions

All input variables described in the previous subsection are shown on the following two pages (fig. B.1, B.2). Each variable is separately plotted for signal (green) and background (red) patterns.

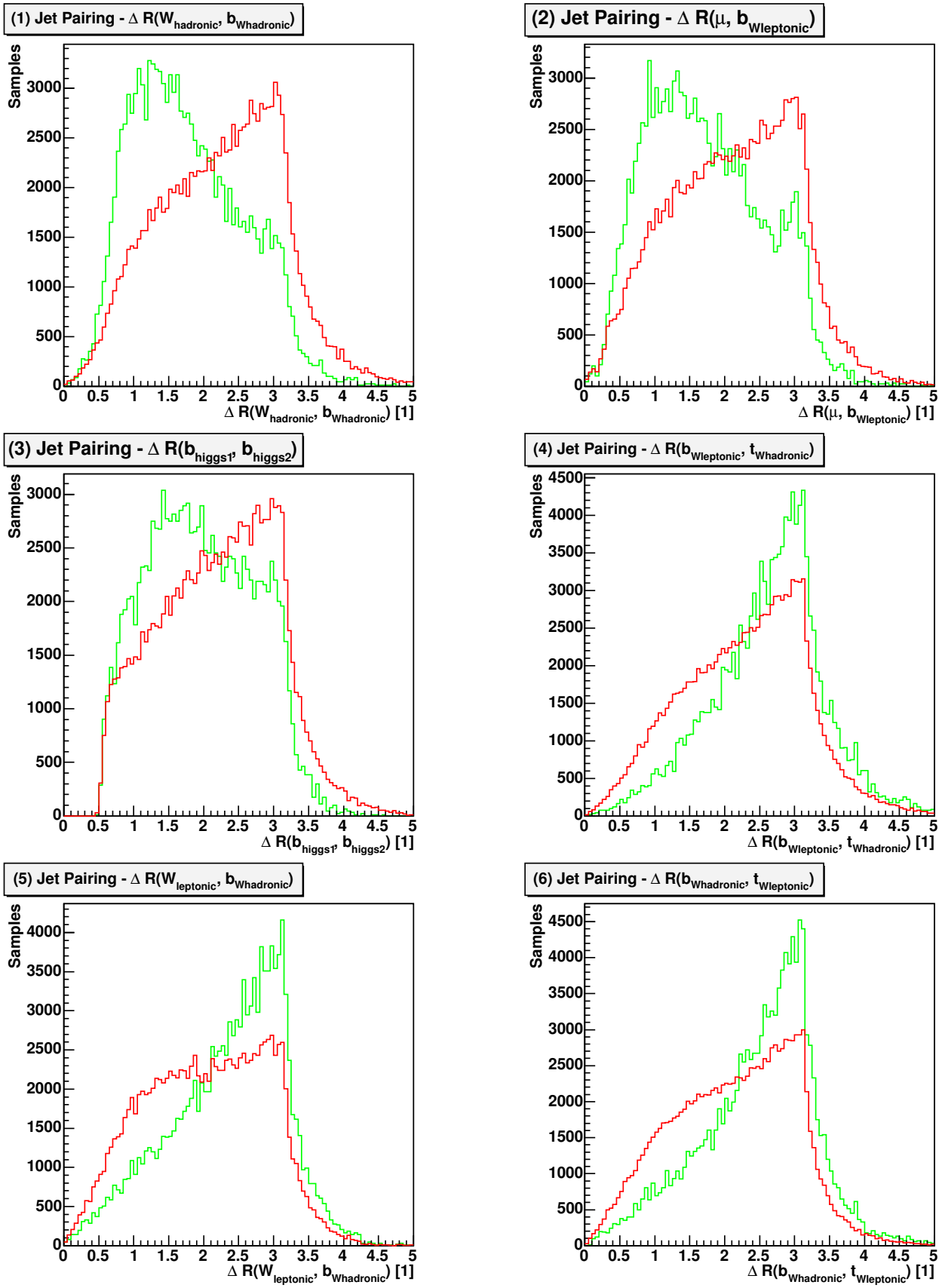

Figure B.1: *Distributions of the input variables used in the training of the jet pairing neural network, separated for signal (green) and background (red) patterns - Part I.*

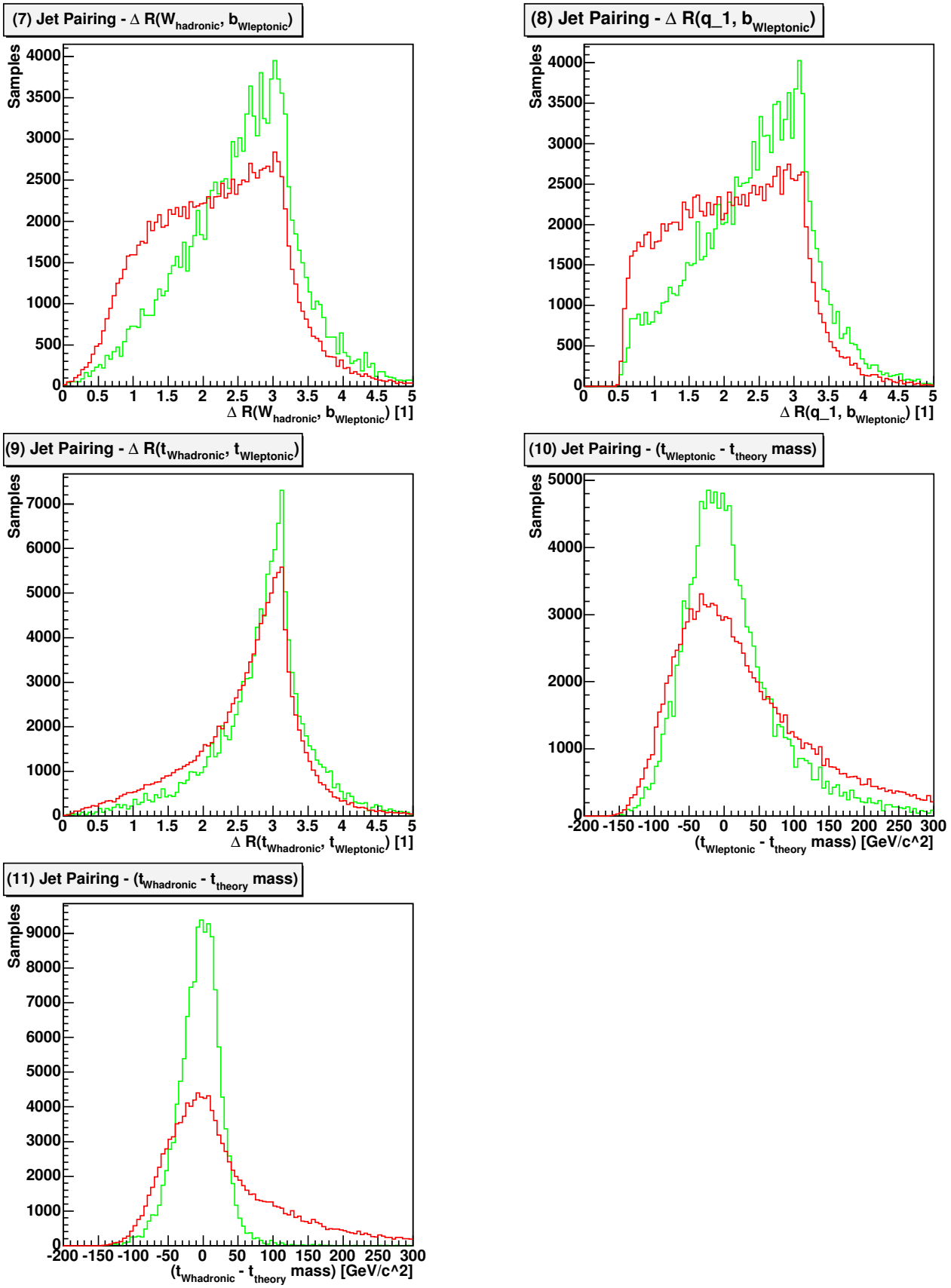

Figure B.2: *Distributions of the input variables used in the training of the jet pairing neural network, separated for signal (green) and background (red) patterns - Part II.*

Correlations

The correlations between the input variables and between the input variables and the target value are shown in fig. B.3. The value of the correlation is colour-coded as depicted on the right. The target value is marked with a T, the input variables are numbered serially, starting with one. Strongly correlated variables are decorrelated in particular, using NeuroBayes®' capabilities.

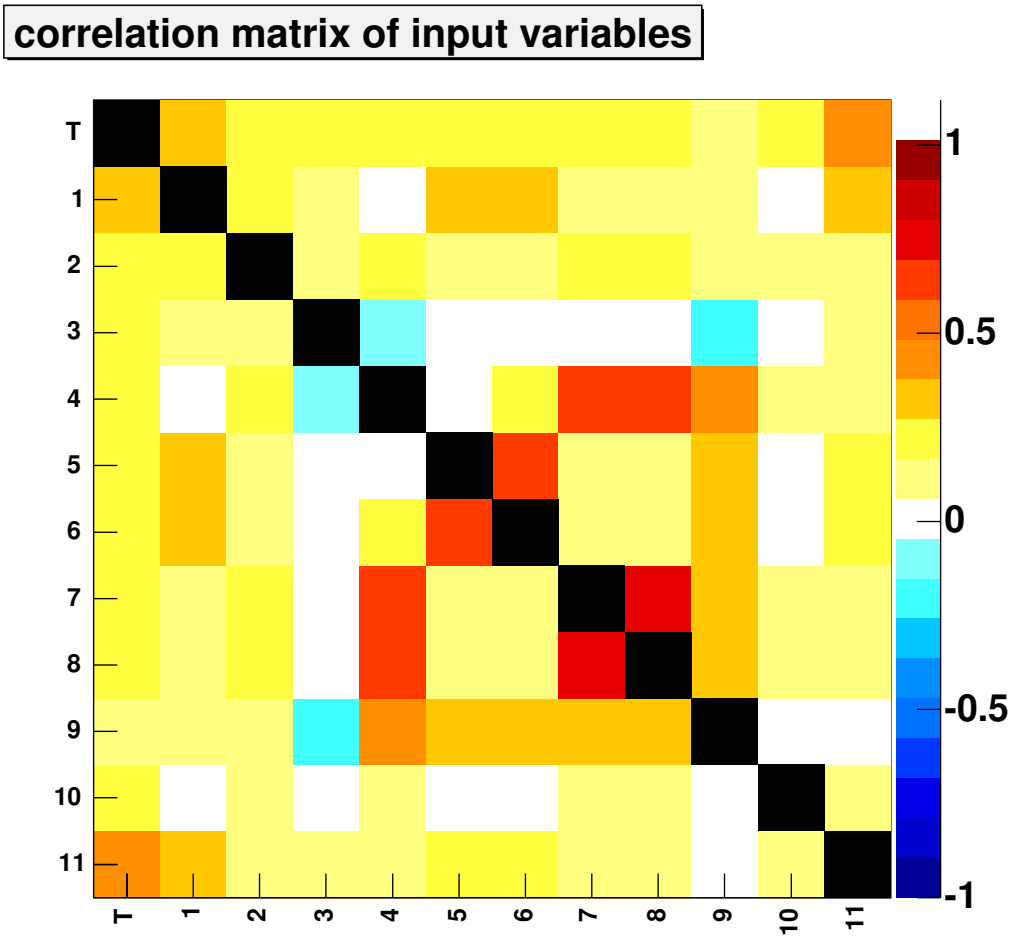

Figure B.3: *Correlations between the input variables (1 - 11) and between the input variables and the target variable (T). The value of the correlation is colour-coded, as shown on the right.*

B.2 Background Suppression

The background suppression network analyses events and gives an estimation whether they are $t\bar{t}H$ signal or $t\bar{t}b\bar{b}$ background events. The latter can then be removed for the following analysis.

Network Settings

The following settings – deviating from the default settings – are applied to the NeuroBayes® network to adjust the training. These values are also used for the combined analysis:

Number of inner nodes:	20
Training iterations:	500
Training fraction:	1.0
Training speed:	50
Learn rate:	20

The training is performed on the good matched fraction of 60,000 signal and 60,000 background events. Particulars about the neural network for background suppression are explained in chapter 6.

Variables Used

The following variables are used for the training of the background suppression network. The two b jets from the Higgs decay are labelled b_{higgs1} and b_{higgs2} and form a $b\bar{b}$ system. In case of background events two arbitrary b jets not coming from a top decay are chosen instead. The top jet nearest in angle to the $b\bar{b}$ system is labelled t_{near}. The applied preprocessing setting is indicated in the brackets.

1. $\measuredangle(b\bar{b}, t_{near})$ (12)
2. $\Delta r(b\bar{b}, t_{near})$ (12)
3. $\Delta R(b\bar{b}, t_{near})$, using pseudo-rapidity for the calculation (12)
4. $\Delta R(b\bar{b}, t_{near})$, using rapidity for the calculation (12)
5. $\Delta R(b_{higgs1},\ b_{higgs2})$ (12)
6. $|\eta(b_{higgs1})|$ (12)
7. $|\eta(b_{higgs2})|$ (12)

Distributions

All input variables described above are shown on the following two pages (fig. B.4, B.5). Each variable is separately plotted for signal (green) and background (red) events.

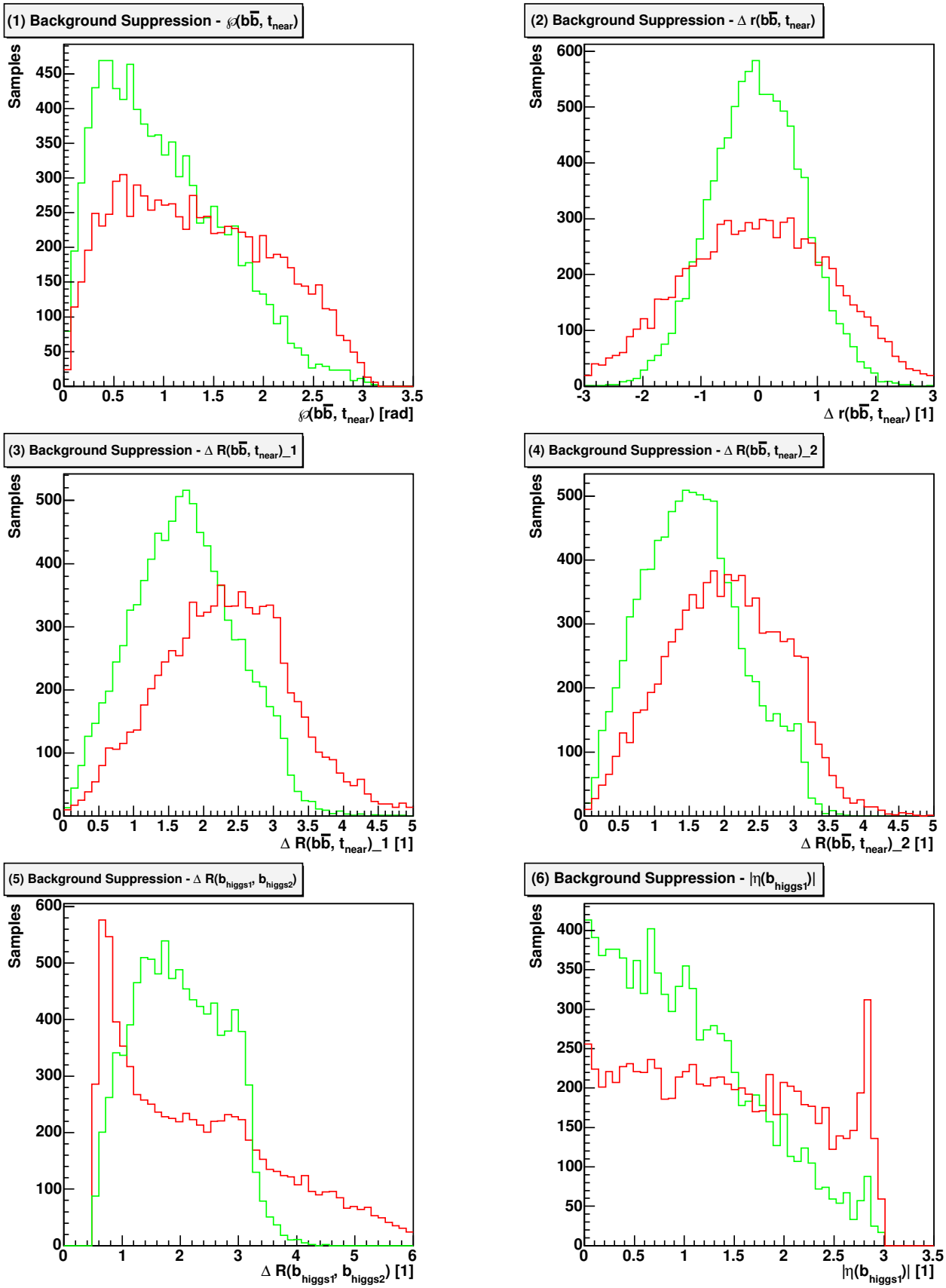

Figure B.4: *Distributions of the input variables used for the training of the background suppression neural network, separated for signal (green) and background (red) events- Part I.*

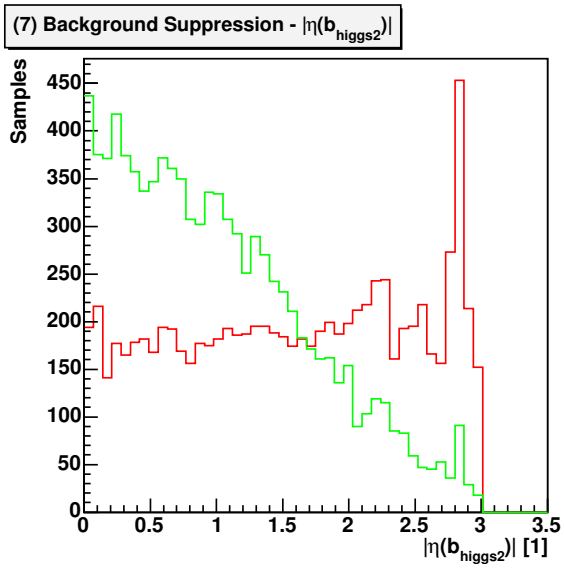

Figure B.5: *Distributions of the input variables used for the training of the background suppression neural network, separated for signal (black) and background (red) events - Part II.*

Correlations

The correlations between the input variables and between the input variables and the target value are shown in fig. B.6. The value of the correlation is colour-coded as depicted on the right. The target value is marked with a T, the input variables are numbered serially, starting with one. Strongly correlated variables are decorrelated in particular by NeuroBayes®.

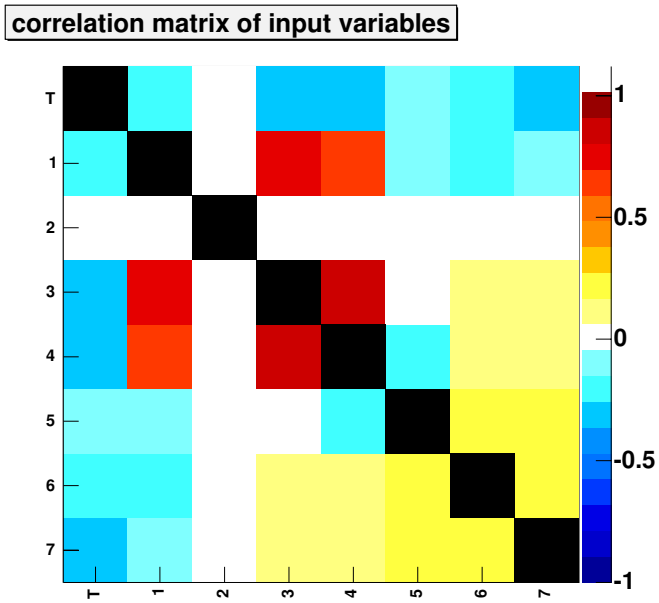

Figure B.6: *Correlations between the input variables (1 - 7) and between the input variables and the target variable (T). The value of the correlation is colour-coded, as shown on the right.*

List of Figures

List of Tables

Bibliography

[1] A. Schmidt, *Studies of the $t\bar{t}H$ channel at CMS*
CMS Note in preparation (2006). II, 53, 56, 67, 81

[2] W. Weimin et al., *A Study of $t\bar{t}$ + Higgs at CMS*
CMS Note 2001/039 (2001). III, 70, 71, 81

[3] The CMS Collaboration, *CMS: ECAL Technical Design Report*
CERN/LHCC 97-33 (1997). 3

[4] The CMS Collaboration, *CMS: The Hadron Calorimeter Technical Design Report*
CERN/LHCC 97-31 (1997).

[5] The CMS Collaboration, *CMS: The Magnet Project Technical Design Report*
CERN/LHCC 97-10 (1997).

[6] The CMS Collaboration, *CMS: MUON Technical Design Report*
CERN/LHCC 97-32 (1997).

[7] The CMS Collaboration, *CMS: The Tracker Project Technical Design Report*
CERN/LHCC 98-6 (1998).

[8] The CMS Collaboration, *CMS: The TriDAS Project Technical Design Report*
CERN/LHCC 2000-38 (2000).

[9] The CMS Collaboration, *CMS: Data Acquisition and High-Level Trigger TDR*
CERN/LHCC 2002-26 (2002). 3

[10] CERN – European Organization for Nuclear Research, *LHC Design Report*
http://ab-div.web.cern.ch/ab-div/Publications/LHC-DesignReport.html. 3

[11] *Compact Muon Solenoid Outreach Activities*
http://cmsinfo.cern.ch. 3, 5, 7, 8, 9

[12] D. Griffiths, *Introduction to Elementary Particles*
ISBN 0-471-60386-4 (1987). 12

[13] D. Perkins, *Introduction to High Energy Physics*
ISBN 0-521-62196-8 (2000). 12

[14] S. Eidelman, et. al, *Review of Particle Physics*
ISSN 0-370-2693 (2004). 13, 20, 90

[15] *Higgs Mechanism – Wikipedia*
http://en.wikipedia.org/wiki/Higgs_mechanism. 13

[16] S. Weinberg, *A Model of Leptons*
Phys. Rev. Lett. 19 (1967). 14

[17] H. Spiesberger, et al., *The Standard Model: Physical Basis and Scattering Experiments*
hep-ph/0011255 (2000). 15

[18] A. Djouadi, et al., *HDECAY: A Program for Higgs Boson Decays in the Standard Model and its Supersymmetric Extension*
hep-ph/9704448 (1997). 18

[19] S. Kappler, *Higgs Search Studies in the Channel $t\bar{t}H$ with the CMS Detector at the LHC*
IEKP-KA/2004-17 (2004). 21, 62

[20] R. Rojas, *Theorie der Neuronalen Netze*
ISBN 3-540-56353-9 (1993). 23

[21] D. Patterson, *Kuenstliche neuronale Netze*
ISBN 3-827-29531-9 (1997).

[22] M. Berthold, *Intelligent Data Analysis*
ISBN 3-540-43060-1 (2000).

[23] A. Zell, *Simulation neuronaler Netze*
ISBN 3-486-24350-0 (1994). 23

[24] W. McCulloch and W. Pitts, *A logical Calculus of the Ideas immanent in nervous Activity*
Bulletin of Mathematical Biophysics 5 (1943). 25, 29

[25] S. Haykin, *Neural Networks, a comprehensive Foundation*
ISBN 0-132-73350-1 (1998). 25

[26] J. Hertz, et al., *Introduction to the Theory of Neural Computation*
ISBN 0-201-51560-1 (1991). 25, 35, 36

[27] F. Rosenblatt, *The Principles of Neurodynamics*
ASIN B-000-6AXUI-I (1962). 28

[28] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the behavioral Sciences*
Ph.D. thesis (Harvard University, Cambridge) (1974). 32

[29] A. Weigend et al., *Generalization by Weight Elimination with Application to Forecasting*
Advances in Neural Information Processing Systems 3 (1991). 33

[30] G. Orr and K. Mueller, *Neural Networks: Tricks of the Trade*
ISBN 3-540-65311-2 (1999). 33

[31] J. Zurada, *Introduction to Artificial Neural Network Systems*
ISBN 0-314-93391-3 (1992). 34

[32] C. Bishop, *Neural Networks for Pattern Recognition*
ISBN 0-198-53864-2 (1996). 34, 36, 40

[33] M. Riedmiller and H. Braun, *A direct adaptive Method for faster backpropagation Learning: The RPROP Algorithm*
Proceedings of the IEEE International Conference on Neural Networks (1993). 34

[34] G. Cybenko, *Approximation by Superpositions of a sigmoidal Function*
Mathematics of Control, Signals and Systems 2 (1989). 36

[35] R. Reed, *Pruning Algorithms: a Survey*
Proceedings of the IEEE International Conference on Neural Networks (1993). 36

[36] M. Feindt, *A Neural Bayesian Estimator for Conditional Probability Densities*
IEKP-KA/04-05 (2005). 38

[37] J. Mason and M. Cox, *Algorithms for Approximation*
ISBN 0-198-53612-7 (1987). 40

[38] T. Poggio and F. Girosi, *A Theory of Networks for Approximation and Learning*
ASIN B-000-72YCX-W (1989). 40

[39] J. Moody and C. Darken, *Fast Learning in Networks of locally tuned Processing Units*
Neural Computation 1 (1989). 40

[40] T. Kohonen, *Self-Organizing Maps*
ISBN 3-540-67921-9 (2000). 41

[41] T. Kohonen et al., *SOM-Pak, the Self-Organizing Map Program Package*. 42

[42] J. Hopfield, *Neural Networks and Physical Systems with emergent collective computational Abilites*
Proceedings of the National Academy of Sciences Vol. 79 (1982). 43, 44

[43] J. Hopfield, *Neurons with graded Response have collective computational Properities like those of two-state Neurons*
Proceedings of the National Academy of Sciences Vol. 81 (1984). 43

[44] D. Amit et al., *Statistical Mechanics of Neural Networks*
Annals of Physics 173 (1987). 44

[45] *CMKIN Release Area*
http://cmsdoc.cern.ch/Releases/CMKIN. 45

[46] *CompHEP*
http://www.ifh.de/ pukhov/comphep.html. 45

[47] *PYTHIA (and JETSET) Webpage*
http://www.thep.lu.se/ torbjorn/Pythia.html. 46

[48] *CMSIM Main Page*
http://cmsdoc.cern.ch/cmsim/cmsim.html. 46

[49] *GEANT - Detector Description and Simulation Tool*
http://wwwasd.web.cern.ch/wwwasd/geant. 46

[50] *The CMS Simulation Project – OSCAR*
http://cmsdoc.cern.ch/cmsoo/projects/OSCAR. 46

[51] *CMS OO Reconstruction*
http://cmsdoc.cern.ch/orca. 46

[52] *COBRA*
http://cobra.web.cern.ch/cobra. 46

[53] *CMS Fast Simulation*
http://cmsdoc.cern.ch/famos. 46

[54] *SCRAM Page*
http://cmsdoc.cern.ch/Releases/SCRAM/current/cgi/scrampage.cgi. 46

[55] *XCMSInstall*
http://cmsdoc.cern.ch/cms/oo/repos_standalone/download/index.php. 46

[56] *CMS Software Page*
http://cmsdoc.cern.ch/cms/cpt/Software/html/General. 47

[57] *The ROOT System Homepage*
http://root.cern.ch. 47

[58] *Physics Analysis Workstation – PAW*
http://wwwasd.web.cern.ch/wwwasd/paw. 47

[59] R. Brun, et al., *ROOT: Users Guide 4.08* (2004). 47

[60] M. Erdmann, et al., *Physics Analysis eXpert Users Guide* (2003). 47

[61] *CLHEP – A Class Library for High Energy Physics*
http://wwwasd.web.cern.ch/wwwasd/lhc++/clhep. 48

[62] E. Gamma, et al., *Design Patterns*
ISBN 0-201-63361-2 (1994). 48

[63] *PAX Home*
http://cern.ch/pax. 49

[64] $<$ phi $-$ t $>$, *NeuroBayes User's Guide* (2004). 49, 50, 51, 86

[65] $<$ *phi* $-$ *t* $>$ *Physics Information Technologies GmbH*
http://www.phi-t.de. 49

[66] V. Buege, *Aufbau eines Grid-Standorts zum Einsatz der CMS-spezifischen Software und Messung von Parametern des W- und Z-Bosons am LHC*
IEKP-KA/2005-21 (2005).

[67] U. Kerzel, *Erste inklusive Messung der b-Quark-Fragmentation f(Z) in Z0-Zerfllen mit dem DELPHI-Detektor bei LEP I*
IEKP-KA/2002-3 (2002).

[68] A. Schmidt, *Entwicklung von Analyse-Software und Bestimmung von Parametern des W-Bosons am LHC durch Vergleich mit Z-Bosonen*
IEKP-KA/2004-2 (2004).

[69] M. Weber, *Vergleich von Monte Carlo Generatoren zur Higgs-Suche am LHC*
IEKP-KA/2005-22 (2005).

Danksagung

Zuerst möchte ich mich bei Herrn Prof. Dr. Günter Quast für die großartige Betreuung meiner Diplomarbeit bedanken. Er hatte stets ein offenes Ohr für meine Fragen und stand mir mit Rat und neuen Ideen zur Seite. Ebenso danke ich Herrn Prof. Dr. Michael Feindt für die Übernahme des Korreferats und die nützlichen Ratschläge zur Verwendung von neuronalen Netzen.

Des Weiteren möchte ich mich bei den Mitgliedern der CMS Arbeitsgruppe bedanken, die mir im Laufe des Jahres bei Problemen und vor allem zum Ende der Diplomarbeit beim Korrekturlesen sehr geholfen haben. Besonders bedanke ich mich bei Herrn Dr. Christian Weiser und Alexander Schmidt für die fachliche Unterstützung. Außerdem gebührt mein Dank natürlich auch Volker Büge, Christopher Jung, Andreas Öhler, Christian Piasecki, Dr. Klaus Rabbertz, Christophe Saout, Armin Scheurer, Dr. Anja Vest und Markus Weber.

Zudem möchte ich den Systemadministratoren und den weiteren Personen danken, die die reibungsfreie Arbeit an diesem Institut ermöglicht haben.

Außerdem danke ich den restlichen Mitgliedern des Instituts und der Arbeitsgruppe, die für eine angenehme Atmosphäre gesorgt haben, so dass ich mich im letzten Jahr hier sehr wohl gefühlt habe.

Insbesondere möchte ich mich noch bei meinen Eltern bedanken, die mich während des gesamten Studiums in jeder erdenklichen Hinsicht immer unterstützt haben.

Hiermit versichere ich, die vorliegende Arbeit selbständig verfasst
und nur die angegebenen Hilfsmittel verwendet zu haben.

Dennis Schieferdecker

Karlsruhe, den 1. Februar 2006