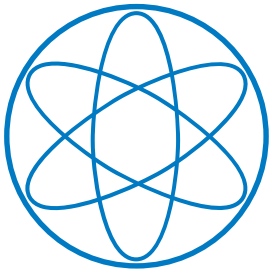
Physik-Department



Application of Deep Neural Networks on Event Type Classification in IceCube

Masterarbeit

von Maximilian Kronmüller (03649019)

Prof. Elisa Resconi

13.12.2018

Technische Universität München

Erstgutachter (Themensteller): Prof. Elisa Resconi
Zweitgutachter: Prof. Susanne Mertens

Contents

CHAPTER 1

Introduction

In September 2017, the first likely source of high-energy astrophysical neutrinos was found, the blazar TXS 0506+056 [5]. Beforehand, the existence of a diffuse flux of high-energy astrophysical neutrinos had been proven but it hadn't been possible to locate individual sources [1]. The event, IceCube-170922A, was measured by the IceCube collaboration, which operates the largest neutrino telescope on Earth. Follow-up measurements were made, which were in accordance with IceCube-170922A. The Large Area Telescope (LAT) on the Fermi Gamma-ray Space Telescope observed a state of enhanced gamma-ray activity in the area of TXS 0506+056. Additionally, gamma rays in a spectrum up to 400 GeV were measured by the Major Atmospheric Gamma Imaging Cherenkov (MAGIC) telescopes [5]. This first evidence of neutrino sources is seen as one of the biggest breakthroughs in the field of multi messenger astronomy. Yet, the discovery of additional sources is still one of the key goals in multi messenger astronomy. With more sources we would have a significantly higher chance to learn more about the sources themselves as well as the flux produced by them. While source searches proceed, new methods must be developed to improve event reconstructions, flavor identifications and computational speed.

In recent years, machine learning methods have shown remarkable progress and are therefore now applied to a wide range of problems across all areas. Especially the field of deep neural networks experienced a tremendous boost [26] [24]. Previously unsolvable problems have become possible to tackle. A good example illustrating this enormous progress was the win of Google's AlphaGo over Mr. Lee Sedol, winner of 18 world titles, in the ancient game of Go in 2016 [32]. The competition received a lot of public attention as computers have been far from competing with professional players in this game beforehand.
Identifying malignant cancer, critical pedestrian detection through autonomously driving cars or the separation of good and bad products are just some examples for classification problems, that arise across almost all fields. It is hoped that one will be able to automate and fasten up processes by solving classification problems through intelligent machines. Furthermore, one thinks to be able to take better decisions, as machines are able to detect patterns in data that humans maybe can't. Machine classification is thus expected to have the potential to transform whole industries. A lot of research was therefore been done in this field in recent years. Many new techniques have been developed and successfully applied to various problems.

The stunning results achieved in other areas through the use of deep neural networks have inspired us to utilize deep learning for classification problems in neutrino

physics. This thesis will concentrate on the IceCube experiment located at the South Pole. Within IceCube different analyses are conducted to answer open questions in astroparticle physics, as for example the search for astrophysical neutrino sources. Many of these analyses require the identification of the event topology or the classification according to other criteria.
One type of analysis focuses on the identification of single events of a particular event topology, for example the double bang structure. The interest arises because double bang structures can only originate from tau neutrinos ν_τ and are therefore very likely to be of astrophysical origin. Other analyses base on larger dataset of one specific event type. Thereby the systematics can be handled easier, compared to a multi-flavor case. In those cases it is important to have an event selection that results in a dataset that is as pure as possible. Often point source searches are performed on datasets consisting only of track-like events, due to their good angular resolution. But also analyses working across event topologies, also called all-flavor analyses, rely on an event classification to distinguish and handle the events appropriately.

As event classification is an essential part of many analyses conducted in IceCube, this thesis will focus on the application of deep neural networks on this task. At the moment in IceCube no method exists that is dedicated to distinguish between the different event topologies. Hence in the scope of this thesis we tried to achieve a well working, fast and easy to use neural network to do so. In the following, the structure of this thesis is outlined.

In **Chapter 2** the IceCube neutrino telescope is introduced in more detail. The general setup and the basic detection unit are explained. Furthermore, the measurement principle within the ice and the data handling are discussed. To conclude this chapter we present the different event topologies that can be seen by IceCube.

Chapter 3 introduces the basic physics principles important for this thesis. Firstly, the neutrino interactions and their cross sections at energies relevant for IceCube are outlined. Secondly, we discuss the different tau decay channels.

In **Chapter 4** the fundamental theory of neural networks is presented. First neural networks are put in context with the broad field of machine learning. Next their basics are introduced, followed by more advanced neural network architectures which are important for of this thesis. Additionally the concept of multi-task learning is outlined. And finally a concept to analyze classification tasks, the confusion matrix, is explained.

In **Chapter 5** we first take a look at the specifics of IceCube data, especially for the use with neural networks. Further the different labels needed as ground truth in supervised learning are defined. The dataset on which a network is trained plays a major role onto the final result. Therefore we continue by introducing our dataset. Moreover our event selection and the different properties of the resulting dataset are presented. To conclude this chapter we discuss the development of the dataset

over time and possible introduced biases.

Chapter 6 explains the final classifier. First its general architecture and its training process are outlined. Next the performance of the classifier is explained separately for each task, namely event topology classification, starting event identification and coincident event identification. At the end of this chapter some exemplary events and their predictions by the classifier are presented.

Chapter 7 presents some of the potential applications of the classifier for the use in IceCube. We show the results again, now weighted to the IceCube best-fit neutrino flux. Further the use for event selections is discussed. To conclude this chapter we present the possibility to use the classifier to find tau neutrino candidates.

The final Chapter 8 ends this thesis with a conclusion and an outlook. Supplementary information, like additional plots or details to our used data, are given in the appendix.

CHAPTER 2

The IceCube Neutrino Observatory

The IceCube Neutrino Observatory or most of the time shortly called IceCube is a large volume neutrino detector, which is operated deep down in the ice near the geographical South Pole. With its cubic-kilometer scale it is up to today the largest neutrino telescope on Earth.
In the following chapter we introduce IceCube in more detail. First the general detector setup, followed by a short outline of the basic measurement principle and how IceCube handles the measured information is discussed. To conclude this chapter, we present the different event topologies which can be seen by IceCube.

2.1 Detector Setup

This section gives an overview about the IceCube detector itself. A comprehensive description can be found in [4], which also builds the foundation of this chapter. For a first overview of the experiment the general setup of IceCube is shown in Figure 2.2.
IceCube is located near the Amundsen-Scott South Pole Station. At the surface the IceCube Laboratory and the IceTop detector are placed. In the ice between 1450 m and 2450 m depth is the InIceArray with its subarray DeepCore. The InIceArray is arranged in 86 strings each instrumented with 60 Digital Optical Modules, short DOMs. In general the DOMs are 17 m apart from each other in vertical direction. The DOMs represent the basic detection units of IceCube. They will be presented in more detail in Section 2.1.1. In total the detector consists of 5160 DOMs. The strings themselves are arranged in a nearly hexagonal grid. Their horizontal distance is about 125 m. The exact arrangement of the strings is shown in Figure 2.1. The strings marked in red are arranged in a denser manner, those strings build the so called subarray DeepCore. While IceCube measures events in an energy range from 100 GeV to several PeV without DeepCore, including the latter allows measurements down to 10 GeV. In the following of this this thesis we will use the term IceCube in a narrower sense for the InIceArray and DeepCore.

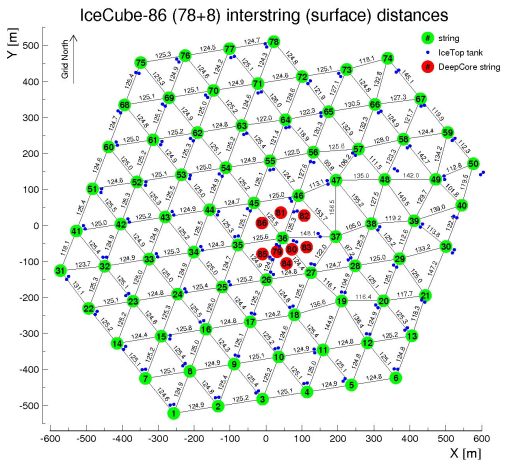


Figure 2.1: The exact placement of the IceCube strings is sketched. The strings marked in green belong to the InIceArray, the ones in red to DeepCore.

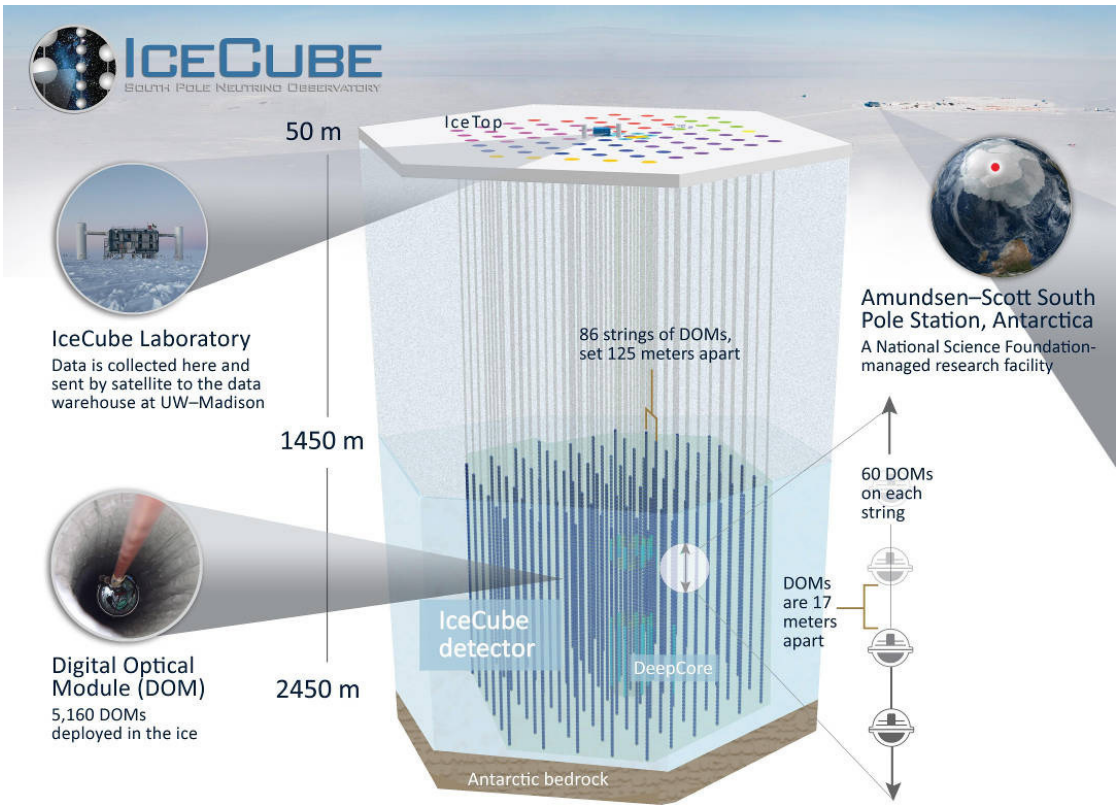


Figure 2.2: To give a first overview of IceCube, many parts of the experiment are depicted. IceTop and the IceCube Laboratory are at the surface. Deep down in the ice the InIceArray with its subarray DeepCore are placed. They consist of 86 strings each equipped with 60 digital optical modules. In average they are horizontally 125 m apart of each other. In vertical direction the digital optical modules are separated by 17 m. More detailed information is given in the text.

2.1.1 Digital Optical Modules

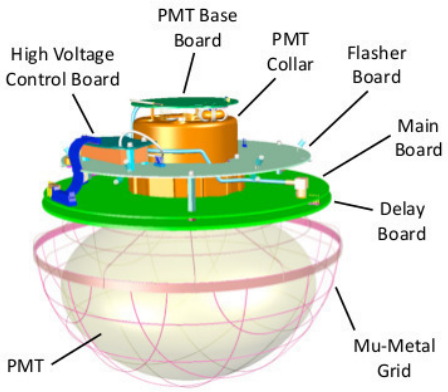


Figure 2.3: Schematic sketch of the setup of a Domestic Optical Module (DOM) as used in IceCube [4].

The DOM is the basic detection unit of IceCube. Its schematic setup is shown in Figure 2.3. It mainly consists of a 10″-diameter down-facing photomultiplier tube (PMT) and the associated circuit boards. All components are surrounded by a glass sphere to be protected from the surrounding. The PMT can measure at single photon level. As the signals reaching the PMT are of wide range, the DOMs have multiple ways to handle it. More detailed information can be found in [4]. Each launched DOM outputs a continuous charge measurement over time, which is called the waveform. Section 2.3 will explain the further handling of the data.

2.2 Measurement Principle

Neutrinos relevant for IceCube produce secondary relativistic particles, which move faster than the phase velocity of light in ice. If this condition is fulfilled Cherenkov light is emitted [18].
The wavefront forms a cone along the particles movement direction. The opening angle of this cone is

$$\cos(\upsilon_c) = \frac{1}{n\beta} \tag{2.1}$$

depending on n, the refractive index of the surrounding medium and β, the speed of the particle in units of the speed of light in vacuum. Figure 2.4 illustrates this principle. These Cherenkov photons can move trough the clear ice, in average the ice in IceCube has a absorption length of over 100 m [23]. If the photons hit a DOM they can be measured by the PMT.

2.3 Waveform and Pulses

A waveform is the most basic level of IceCube data. A higher level representation of it are so called pulses. In this section we will look at both in more detail.
Each DOM measures charge. If a given charge threshold is surpassed, the DOM launches. The waveform is the charge over time measurement of a single DOM. The launched DOMs waveforms are then digitized and saved . For the digitization two different ways exist in IceCube. First, the analog transient waveform digitizer (ATWD) and second a continuously sampling fast analog digital converter (fADC). The ATWD recording duration is 427 ns, with a sampling of 3.3 ns. Light within

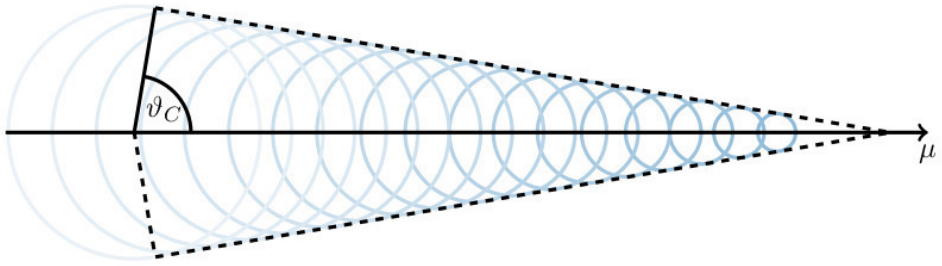
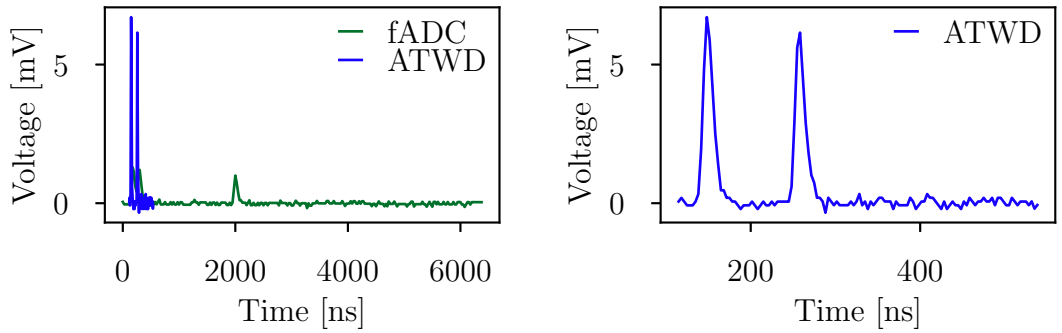


Figure 2.4: Principle of a Cherenkov Cone, illustrated for a muon traveling with nearly the speed of light, $\beta = 0.95$, in a medium with a refractive index of n = 6.31 [23].



(a) fADC and ATWD signal

(b) ATWD signal

Figure 2.5: On the left the same signal once digitized by the ATWD and once digitized by the fADC is shown. The fADC uses a sampling of 25 ns and measures over a time window of 6.4 μs. In comparison the ATWD uses a sampling of 3.3 ns and measures over a shorter time window of 427 ns. On the right a closer look on the ATWD signal is depicted.

tens of meters of a DOM can be measured. The fADC measures for a longer time period of 6.4 μs. Thereby the fADC samples with a rate of 25 ns, resulting in twice as many steps as the ATWD [4]. The digitalizations and therefore data recording can be triggered multiple times, especially the ATWD. The same signal digitized by the two different digitizers is shown in Figure 2.5a. In Figure 2.5b a zoomed version of the ATWD signal is shown.
A more compressed form of the digitized waveforms explained in the last paragraph are the so called pulses. A pulse template is fitted to each significant peak. The resulting pulse has charge and time information. In addition the width of each pulse is stored. The corresponding pulses for the waveform shown in Figure 2.5a are depicted in Figure 2.6. The pulses combine the information of both digitization methods, the ATWD and the fADC [4].

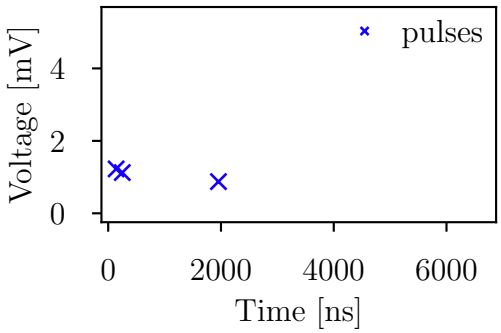


Figure 2.6: The corresponding pulses to the waveform depicted in Figure 2.5 are illustrated.

2.4 Event Topologies in IceCube

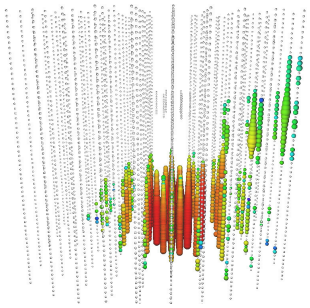
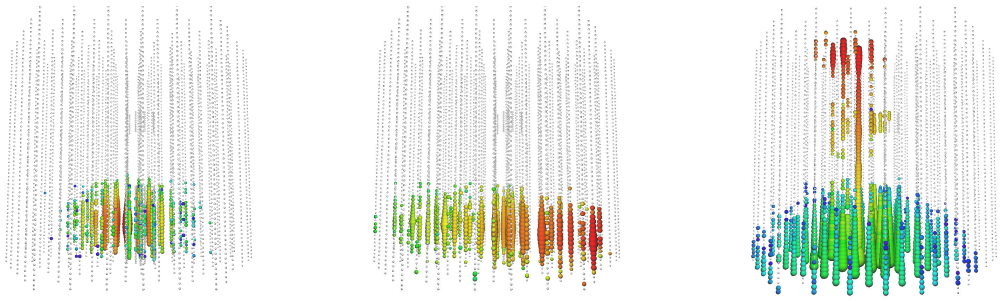


Figure 2.7: Starting Track

In IceCube we have three main event topologies: cascade, track and double bang. Additionally we want to distinguish between incoming and starting tracks, as subclasses of tracks. Event views of these topologies are shown in Figure 2.7 and Figure 2.8.
In an IceCube event view the detector is indicated by all DOMs drawn as small black dots. The DOMs that got hit during the event are highlighted. Thereby the size of the bubble correlates with the total measured charge at the DOM. The color symbolizes the time information. Early hit DOMs are colored in red, late hit DOMs in blue.

Cascades

Cascades are mainly produced by two different processes, either by charged current interactions of electron neutrinos ν_e within the ice or by neutral current interactions of neutrinos of any flavor [14]. In the case of charged current interactions of electron neutrinos a light electron is produced. It looses most of its energy via bremsstrahlung. This results in a hadronic and an electromagnetic cascade, which appears nearly omni-directional with a radius of only a few meters depended on the energy. The hadronic cascade produced by a neutral current interactions looks very similar. Both result in an cascade-like event which shows up in the IceCube detector as shown in Figure 2.8a.



(a) Cascade (b) Track (c) Double Bang

Figure 2.8: On the left a cascade-like event, in the middle a track-like event and on the right a event with a double bang-like structure is presented. The primary particle causing the cascade had an energy of 410 TeV. The track is a through-going muon caused by a muon neutrino with an energy of 342 TeV. The primary tau of the double bang had an energy of 7.6 PeV and is therefore very rare. Furthermore the perfect contained signature of the event inside the detector is very idealized. On the contrary however, the characteristics of a double bang event can be identified more clearly.

Tracks

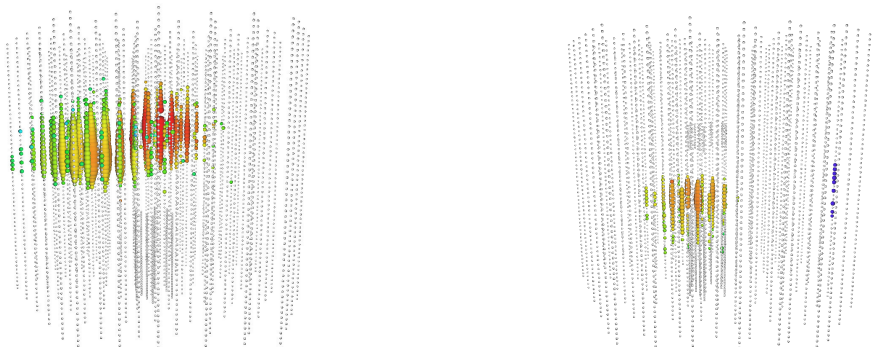
Tracks are only produced by muons μ. These muons originate from three different processes, first of charge current interactions of muon neutrinos ν_μ second from taus τ which decay into muons and finally from atmospheric muons. Muons have a heavier mass than electrons and a long lifetime of about 2.19 μs [7]. As a result they travel through the ice up to several kilometers [14]. Thereby they produce Cherenkov light on their way, resulting in the track-like events as shown in Figure 2.8b.

Double Bangs

Double bangs can only originate from one process: a tau neutrino interacts via a charged current interaction, a tau will be produced, which travels and decays inside the detector. In about 82.5% of the cases the tau decay produces an electron or a hadronic cascade [3]. If all parts are contained in the instrumented volume this results in the unique structure of two cascades connected by a track. Such an event is shown in Figure 2.8c. It is the optimal case, a much more realistic scenario is depicted in Figure 2.9a. Here the track-like component is harder to identify and the cascades overlap. The more energetic the incoming tau is, the longer is its decay length. In IceCube it scales in average about 5 cm/TeV = 50 m/PeV [3]. As a result, we hope to be able to detect tau neutrinos with IceCube starting at energies of a few hundred TeV [3].

Coincident Events

In IceCube coincident events are events that have at least a second particle going through the detector in the time window of the event. Such a secondary particle



(a) Double Bang (b) Cascade with an coincident muon

Figure 2.9: A more realistic double bang, compared to the one in Figure 2.8c, is depicted on the left. It was caused by a tau neutrino with an energy of 4.3 PeV. A event labeled as a coincident event is shown on the right. In the middle of IceCube a cascade is located accompanied by a coincident muon. The muon is going straight down and is marked in blue.

can for example be a muon or a muon-bundle. An exemplary event view is shown in Figure 2.9b. A coincidence is a hint for the atmospheric origin of the event, due to their common production in showers.

Starting Events

In general starting events are, as the name already suggests, events, whose initial vertex is inside the detector volume. The exact definition also differs inside IceCube and depends on the use case. The definition for this thesis will be clarified in Section 5.2.2. The majority of the observed cascade-like events are also starting events, due to their small spacial extension. Therefore this distinction is of greater importance for track-like events. As a result one often sees a further split between through-going or incoming tracks and starting tracks. For simplicity we will call incoming tracks in the following short only tracks. An example of a starting track is depicted in Figure 2.7.

CHAPTER 3

Particle Physics

3.1 Neutrino Interactions

At energies relevant for IceCube all neutrinos and their anti-particles, independent of their flavor, mainly interact in deep inelastic scattering. They do so in two different ways, neutral current interactions (NC) or charged current interactions (CC) [10].

$$\nu_l + X \rightarrow l + Y \qquad (CC) \tag{3.1}$$

$$\nu_l + X \rightarrow \nu_l + Y \qquad (NC) \tag{3.2}$$

In the Equations 3.1 and 3.2 ν_l denotes a neutrino of flavor l, X is a nucleus, l the corresponding lepton and Y symbolizes a hadronic cascade that is produced along. In a CC interaction the weak force is mediated by a W boson and a charged lepton is produced. In contrast, for NC interactions the weak force is mediated by a Z boson [10].
Furthermore a third interaction has to be considered, the Glashow resonance,

$$\bar{\nu}_e + e^- \rightarrow W^-. \tag{3.3}$$

An anti-electron-neutrino $\bar{\nu}_e$ scatters from the bound electrons of the molecules e^- and produces a charged W^-. The latter can decay further into all leptons with its corresponding flavored anti-neutrino or into hadrons. The branching ratio for the W^- to decay into hadrons is $67.60 \pm 0.27\%$, to decay via leptons is hence $32.8 \pm 0.27\%$. Non of the flavors is preferred, so each has a probability of $10.8 \pm 0.09\%$ to occur [7]. This process is suppressed in most energy ranges. However, at an energy of several PeVs, with a sharp peak at approximately 6.3 PeV, the cross section becomes considerably large [12]. Figure 3.1 shows the cross sections of all three processes in the energy range that is relevant for IceCube. The difference between the cross sections of neutrinos and anti-neutrinos becomes negligible for high energies [11].

3.2 Tau Decay

In the charged current interaction of a tau neutrino a tau-lepton is produced

$$\nu_\tau + X \rightarrow \tau + Y. \tag{3.4}$$

The lifetime of the τ, 29,06 ps, is very short [7]. The tau decay length in ice scales with the energy $50 \frac{m}{PeV}$. Multiple ways for the tau decay exist. First it can decay

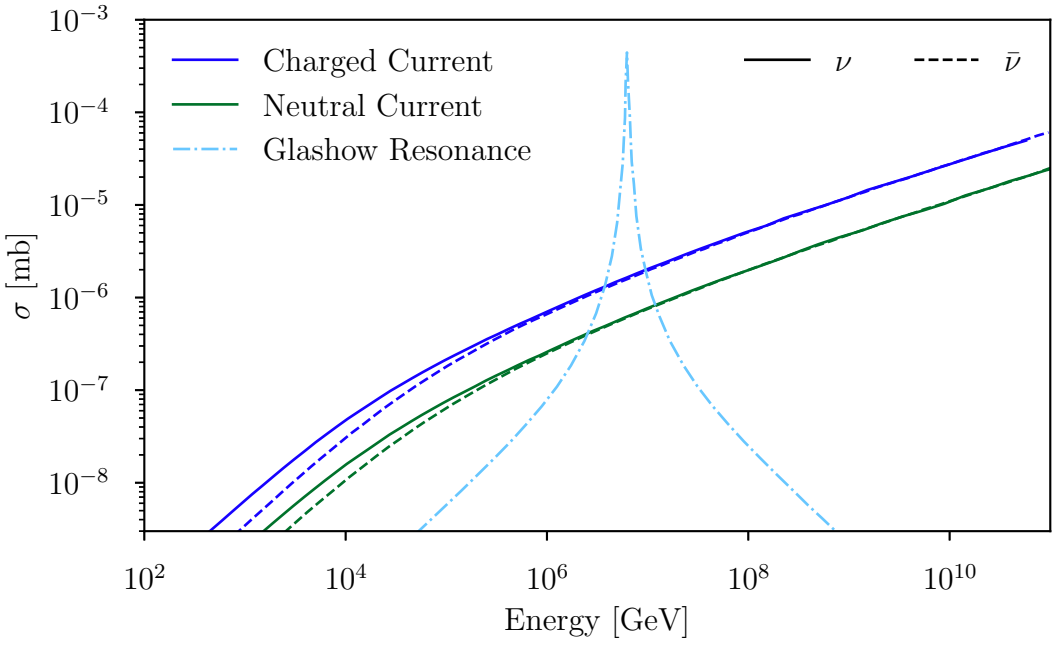


Figure 3.1: Cross sections for deep inelastic neutrino nucleon scattering via CC and NC interactions and the Glashow resonance in the energy range relevant for IceCube.

into the lighter leptons

$$\tau^- \to e^- + \bar{\nu}_e + \nu_\tau \qquad (17.83 \pm 0.04)\% \qquad (3.5)$$

$$\tau^- \to \mu^- + \bar{\nu}_\mu + \nu_\tau \qquad (17.41 \pm 0.04)\% \qquad (3.6)$$

which both have a probability to occur of around 17%. Otherwise the tau can decay into different hadrons. The most important channels are denoted below.

$$\tau^- \to \pi^- + \pi^0 + \nu_\tau \qquad (25.52 \pm 0.09)\% \qquad (3.7)$$

$$\tau^- \to \pi^- + \nu_\tau \qquad (10.83 \pm 0.06)\% \qquad (3.8)$$

$$\tau^- \to \pi^- + 2\pi^0 + \nu_\tau \qquad (9.30 \pm 0.11)\% \qquad (3.9)$$

The anti-particle τ^+ behaves in the same manner, where the corresponding equations can be derived replacing all particles with their corresponding anti-particles. The information about the tau decay is based on [7].

CHAPTER 4

Theory of Neural Networks

In this chapter we first discuss the role of neural networks in the broad field of machine learning. Afterwards the basics of neural networks are explained in more detail, followed by more advanced architectures needed in the context of this thesis, namely residual units and inception units. Furthermore, the concept of multi-task learning is explained. To conclude we present a common way to analyze classification tasks, the confusion matrix.

4.1 Neural Networks in the Field of Machine Learning

Machine learning is the approach to teach computers to solve problems based on data without explicitly programming them. It is a broad field that summarizes many techniques and algorithms. A common way of categorizing these is to discriminate them based on the way they are learning. Generally one distinguishes between supervised learning, unsupervised learning and reinforcement learning [27]. In the following each of them is described shortly.

- **Supervised Learning** - In supervised learning a model's prediction is evaluated through comparison with the correct solution, the *ground truth* [27]. Hence the correct solution to the given problem needs to be known. The problems targeted with supervised learning are classification and regression tasks. In regression tasks a continuous variable is predicted for example the energy of a particle. If the outcome is of a discrete set it is a classification task. One example is the categorization of images dependent on the shown animal.

- **Unsupervised Learning** - As opposed to supervised learning, unsupervised learning does not make use of direct feedback in terms of the correct solution. Instead it tries to find hidden structures inside the data [27]. Typical use cases for this type of learning are clustering problems, for example analyzing customer data and grouping them.

- **Reinforcement Learning** - It is used in more dynamic environments where a interactive behavior depending on the current situations should be learned. Therefore a rewarding system that gives feedback to the model is implemented [27]. For example if playing a game should be learned, the system will get positive feedback if it wins and negative feedback if it loses. Based on this feedback the model is adapted. One of the greatest breakthroughs of

reinforcement learning was AlphaGo developed by Google DeepMind. It won against the worlds best player in the old Asian game of go. Before computers were not able to compete with professional players in this game, due to the huge amount of possible moves [32].

As our goal is to classify different event types in IceCube, we face a classification problem which we will solve through supervised learning. From the variety of algorithms used for solving supervised learning problems, we chose to work with deep neural networks. They have shown great performance in recent years, especially in the field of image classification tasks [24] [31], which our task is related to.

4.2 The Basics of Neural Networks

In this section a basic introduction to the theory of neural networks is given. This shall enable the reader to understand the techniques used in this thesis more easily. First this section discusses how a neural network can be built and which different parts are needed. In a second step we introduce the different aspects of the training process.

4.2.1 Building a Neural Network

A network is structured in a layer-wise manner. According to their position inside the network, three layer types can be distinguished. The basic structure of each network is composed of first an input layer, second, one or several, hidden layer(s) and last an output layer.

- **Input Layer** - It is the first layer in a neural network. It has the same size and dimension as the input data and acts as an interface between the data and the rest of the neural network.
- **Hidden Layer(s)** - All layers between the input layer and the output layer are so-called hidden layers. Their number is arbitrary. These layers process the data and forward it to the final output layer.
- **Output Layer** - The final prediction of a neural network is given by the last layer, the output layer. The shape of the output is predefined through the sought-after solution to the given task.

Rosenblatt Perceptron

The Rosenblatt Perceptron is the most basic unit of neural networks. It was developed and named by F. Rosenblatt in 1957 [28]. The perceptron mimics a biological neuron, therefore it often is shortly called neuron as well. It is mathematically described through

$$y = f\left(\sum_{i=1}^{N} w_i x_i + b\right), \tag{4.1}$$

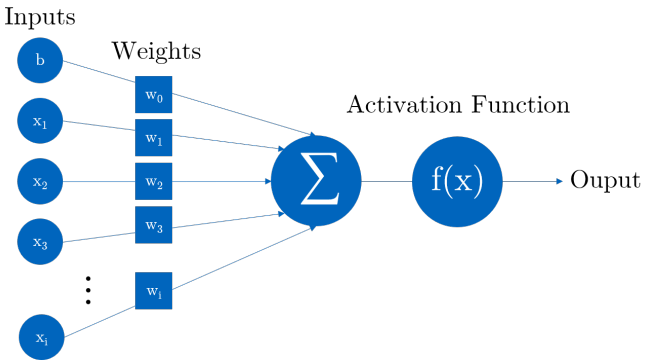


Figure 4.1: A schematic sketch of a Rosenblatt perceptron is shown. All inputs and the bias get weighted and summed. Afterwards an activation function is applied to the sum.

where y represents the output, x_i a single input value, w_i the corresponding weight, b an added bias and f an activation function. i goes from 1 to N, where N is the number of inputs. The activation function is chosen non-linear so non-linear problems can be challenged, otherwise a neural network could be reduced to one matrix multiplication. In the following paragraph the activation functions are presented in more detail. A sketch of a perceptron is shown in Figure 4.1.

Activation Function

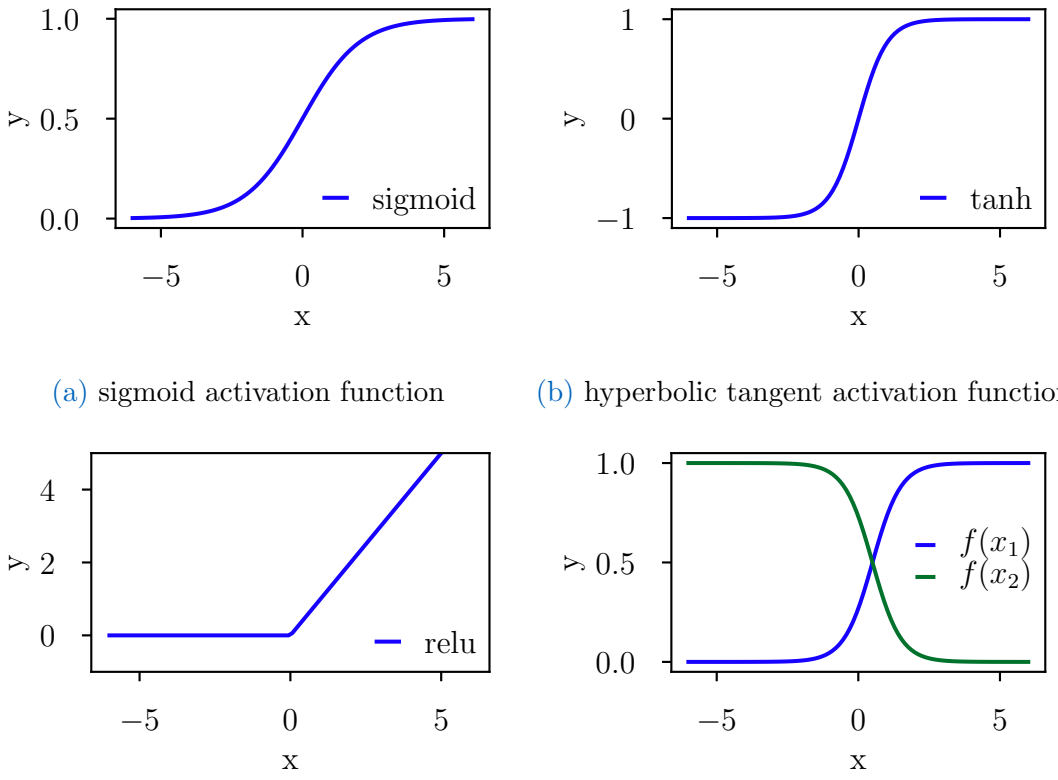
The activation function is applied to the sum of all weighted inputs. Due to its non-linearity neural networks can solve non-linear problems. In most of the cases it maps it's input values into the range $[0, 1]$ or $[-1, 1]$ to stabilize the trainings process. Two common activation functions, the sigmoid and hyperbolic tangent functions [27], are depicted exemplary in the Figures 4.2a and 4.2b.
ReLu, short for rectified linear unit, is a term for a unit that uses the rectified activation function [6]. It is one of the most commonly used activation function. It maps all negative values to zero and all positive values to themselves. Mathematically simply expressed by

$$f(x) = \max(0, x). \tag{4.2}$$

An illustration is shown in Figure 4.2c.
For classification tasks the activation function for the last output layer of the network is typically chosen to be the softmax function, which is defined as

$$f(x_i) = \frac{\exp(x_i)}{\sum_{j=0}^{k} \exp(x_j)} \quad i = 1, ..., k \quad , \tag{4.3}$$

where k denotes the number of classes and x_i the prediction for class i. A sketch of the softmax function for two classes is depicted in Figure 4.2d. Its two main characteristics are that all output values are in the range of $[0, 1]$ and sum up to 1. This two characteristics make a probabilistic interpretation possible, but its meaning still needs some additional discussion. In classification tasks the class with maximum softmax function output will be predicted as the solution class.



(a) sigmoid activation function

(b) hyperbolic tangent activation function

(c) rectified activation function

(d) softmax activation function

Figure 4.2: Four commonly used activation functions are depicted, on the top left the sigmoid function and on the top right the hyperbolic tangent. On the bottom left the rectified activation function is shown. On the bottom right the softmax function for two possible classes is sketched.

Dense Layer and a Fully Connected Neural Network

The easiest layer form in a neural network is a dense layer. It consists fully out of neurons/perceptrons arranged parallel to each other [31]. The neurons of each layer i are connected to the neurons in the preceding $i-1$ as well as the subsequent layer $i+1$: the outputs of the preceding layer $i-1$ thereby serve as input to the neurons of the layer i, while the output of the neurons of the layer i serve as input to the neurons of the subsequent layer $i+1$. In dense layers each neuron is connected to all neurons in the subsequent layer, this is called fully-connected. If neurons are only connected to neurons in subsequent layers and information is thus only passed on in one direction we call this a feed-forward network. In case both requirements are fulfilled, this results in one of the easiest forms of a neural network, a feed forward neural network built fully out of dense layers. A schematic sketch of such a network with two hidden layers is illustrated in Figure 4.3.

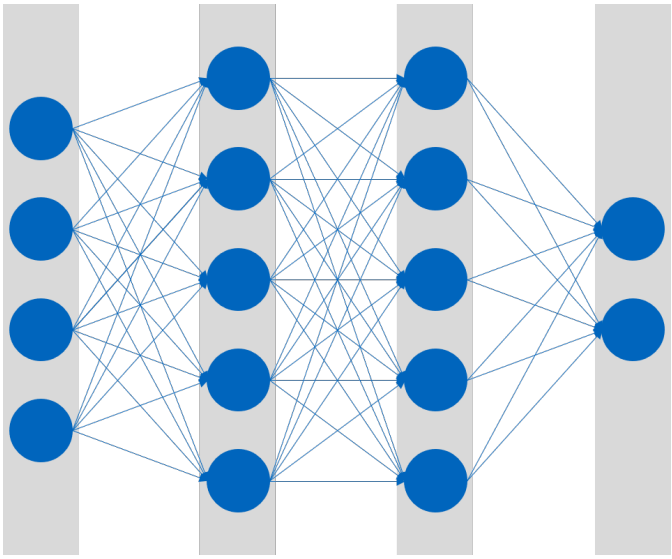


Figure 4.3: A schematic sketch of a fully connected feed forward neural network is shown. The network has two hidden layers and consists of 16 neurons. Thereby each neuron is connected to all neurons of the subsequent layer.

Convolutional Layers

A layer type that has shown excellent performance in recent years is the convolutional layer. A detailed introduction may be found in [24] and [25].
Instead of using the output of every neuron in the preceding layer as input to neurons in the following layer, as it is the case for fully connected layers, convolutional layers extract features through filter operations. As a result they create feature maps, that can be used as input for consecutive layers. By the filters spacial relations are taken into account and the number of weights can be decreased.
The concept is based on the mathematical operation called convolution. In a discrete two dimensional case the convolution operation may be defined as

$$(g \star h) = \sum_{\tau_u} \sum_{\tau_v} f(\tau_u, \tau_v) h(x - \tau_u, y - \tau_v), \tag{4.4}$$

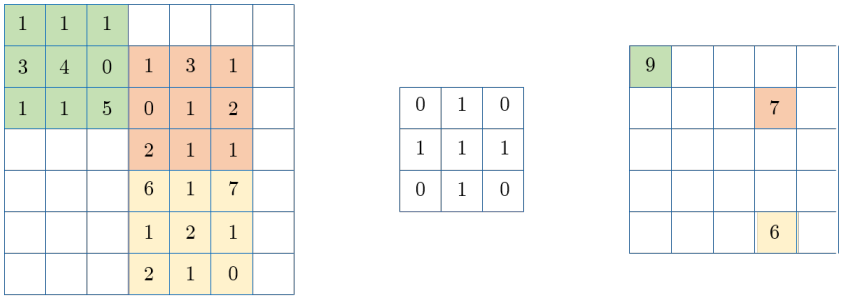


Figure 4.4: The general principle of the convolution operation in a two dimensional discrete case is shown. The highlighted areas indicate the mapping of the input by the filter to the output. The filter is of size 3×3 and uses a stride of 1 in every direction with no padding. This results in a transformation of the input size from 7×7 to 5×5.

where g symbolizes the input and h is the filter or kernel. The filter size is usually a lot smaller than the input size. The idea of the convolution can be clarified very well at the example of a two dimensional grey-scale image. It is may be understood as the movement of a small filter over the whole image. Thereby, the filter output at one position is the sum of the dot product of the image pixel lying below the filter and the corresponding filter value at each filter position. Further the filter is moved with a step size, this is called stride. Depending on the filter size the output can have a different size than the input. A method to prevent that is to pad the input with additional values, most of the time chosen as zero. The convolution method is schematically depicted in Figure 4.4.
While we have introduced a discrete two-dimensional convolution above, the operation can be extended to both higher dimensions as well as the continuous case.
In a convolutional layer multiple convolutions with different filters are performed, whereby each filter tries to extract a different feature. These are stacked afterwards. The resulting set of filtered images are also called feature-maps. These can be used as input for further convolutional layers.

Pooling Layers

Pooling layers are used to reduce the size of the neural network. A common pooling type is "MaxPooling" [30]. Instead of using a filter it uses a kernel that outputs the maximum value of the observed area. Normally a stride is used such that the kernels not overlap. If for example a 2×2 kernel is used the data shrinks by a factor of four. An illustration of this principle is shown in Figure 4.5. Also other pooling strategies exist, for example "AvgPooling". Here the average over the observed area is the output.

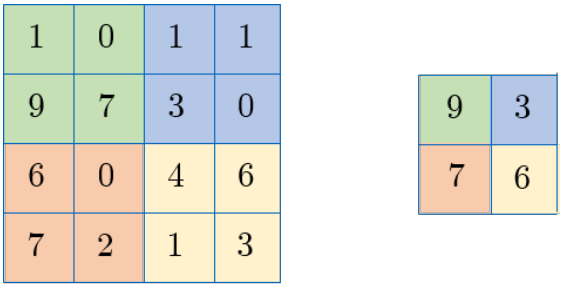


Figure 4.5: The general principle of the MaxPooling operation in a two dimensional case is shown. The highlighted areas indicate the mapping. The input is of size 4×4 and the filter is of size 2×2, this results in a grid of size 2×2.

4.2.2 Training of a Neural Network

After the definition of the network structure, it has to be trained to solve a given problem. During the trainings process the weights and biases of the network are repeatedly updated in order for the network output to match the correct solution as close as possible. The optimal weights are usually found by minimizing a loss function, through back-propagation. The definition of a suitable loss function, which will be explained in more detail in the next paragraph, is therefore essential for a good performance.

Loss

The loss function L provides a measure of the "quality" of the predictions of the neural network and depends on the weights of the network, here denoted all together as θ. The loss function evaluates the difference between the predictions p and the correct solution q, the ground truth.
A common example for a loss function is the mean squared error [20], mathematically described as

$$L(\theta) = \frac{1}{N} \sum_{i}^{N} \left[p(\theta, x_i) - q(x_i) \right]^2 , \tag{4.5}$$

where N denotes the number of all evaluated events. The mean squared error loss is mostly used for regression tasks and calculates the averaged sum of the squared differences between the prediction and the ground truth.
A typical loss function for classification tasks is the categorical crossentropy [20] defined as

$$L(\theta) = \frac{1}{N} \sum_{i}^{N} \sum_{j}^{n} q_j(x_i) \cdot \log(p_j(\theta, x_i)), \tag{4.6}$$

where the ground truth q will be 1 for the correct class and 0 for all others. N denotes the number of all evaluated events and n the number of classes. Hence only the prediction p of the correct class contributes to the loss. The categorical crossentropy penalizes all wrong predictions but especially those that are certain but wrong. So the lower the predicted output for the correct class is, the higher the contribution to the loss gets.

Optimizers

Multiple ways to update the weights based on the loss exist and are summarized in the term optimizers. They provide strategies to efficiently update the weights of the network to minimize the output loss and lead to stable convergence. We will introduce two different algorithms in the following: gradient decent and Adam.
Gradient descent is one of the most important and basic algorithms for weight adaptation in machine learning [29]. The output loss is minimized by iteratively updating the model parameters θ in direction of the gradient of the loss $\frac{dL}{d\theta}$. The update steps are scaled through the constant learning rate α, resulting in

$$\theta_{i+1} = \theta_i - \alpha \frac{dL}{d\theta_i}, \tag{4.7}$$

where i denotes the number of the respective iteration step. The gradient descent algorithm is not guaranteed to find a global minimum nor does it find a minimum the fastest way possible.
The Adam optimizer is a more advanced version of the gradient descent algorithm. The main difference to basic gradient descent is that the learning rate α is not constant. Instead it is adapted during the training process to fit the change of the weights to the current situation. The update of the learning rate is based on both the averages of the first and second moment of the gradient. The Adam algorithm has empirically proven to work very well for a large variety of problems and efficient. More details on Adam can be found in [22].

Regulators

An important aspect of training neural networks is the stability of the training process. A variety of methods to improve the training procedure, called regulators, exist. In the following we introduce two of them, dropout and batch normalization.
Dropout is utilized to improve the generalization of a neural network while training. In each training step a given percentage of neurons, the drop-out-rate, is randomly selected and dropped. This means they are neglected completely in this training step and do not contribute to the output. The dropout of neurons forces the units to learn similar or the same features redundantly, which can lead to better generalization and furthermore prevents overfitting [33].
The goal of batch normalization is to accelerate the training of the neural network by reducing the internal covariance shift [19]. This is achieved by performing an additional step between layers, in which layer outputs are normalized. In the context of this thesis normalization refers to a transformation of the data to zero mean and unit variance.

Control the Training Process

If a network is big enough and trains long enough it will be able to predict the data used for training perfectly. On the counter side it performs bad on unknown data because the network does not generalize very well. This effect is called overtraining.
To prevent overtraining the dataset is commonly split into three different parts: a trainings set, a validation set and a test set. The trainings set is used for the actual training. The validation set is used for monitoring while training. One

way is to apply a network to the validation set after each epoch. If the loss on the validation set increases compared to the previous epoch this is a sign for overtraining. Therefore training is stopped at the minimum of the so called validation loss. The test set is only used once on the fully trained network for all final tests.

4.3 Residual and Inception Units

4.3.1 Residual Units

Residual units have first been introduced by Microsoft in 2015 through their neural network ResNet, which won the ImageNet competition [15]. The basic idea behind residual units is that one can build deeper and deeper network structures with the minimal condition not to impair the neural network by an additional layer.
In general, a complete network layer can be described as a function $y = f(x)$, where the network learns the function f and x and y symbolize the input and output. For a residual unit an additional connection with the most simple function, the identity function $\mathrm{id}(x) = x$, is added. The resulting structure, compare Figure 4.6, can be expressed as

$$y = f(x) + \mathrm{id}(x). \tag{4.8}$$

Identity connections enable layers to only learn additions to the previous layers. The layer does not have to learn how to keep the previous status. In order for the residual units to learn incremental features we have to initialize the layers weights close to zero. Another advantage of neural networks which consist mostly of residual units is, that they suffer less from vanishing gradients, as the identity connection propagates the gradient through the model.

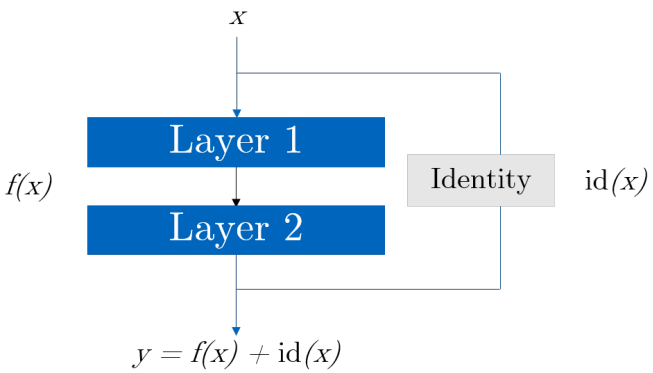


Figure 4.6: A sketch of the working principle of a residual unit with two layers is depicted.

4.3.2 Inception Units

Inception units were introduced by Google end of 2014. Several versions have been developed, each iteratively improving upon the previous one. The three different versions together with their corresponding papers are listed below:

- Inception v1 [35]
- Inception v2 and v3 [36]
- Inception v4 and InceptionResNet v1 and v2 [34]

The general problem inception units target is that for a convolution the size of the filter has to be chosen beforehand. However, depending on the situation different filter sizes have different advantages. The inception units overcome this problem by providing the network a variety of filter sizes. This way they train the network to learn, which of the filter sizes it needs and how to combine the gained information. The naive concept of one inception unit is shown in Figure 4.7. The unit uses a 1×1-convolution, a 3×3-convolution, a 5×5-convolution and a 3×3-max-pooling operation at the same time. Afterwards the individual convolution outputs are concatenated and can then be used as input for further inception units. This naive approach is however computationally extremely expensive, hence a more efficient version adding 1×1-convolutions to decrease the computations was introduced. The 1×1-convolutions lead to a dimension reduction before the bigger and more expansive convolutions are performed [35]. This was the overall first version (v1) also named GoogLeNet.
Version v2 and v3 mainly improved upon the accuracy of version v1 while decreasing the computational cost at the same time. The main change between the versions was the splitting of convolutions into several smaller ones. 5×5-convolution can for example be split into two 3×3-convolutions. The 5×5-convolution is 2.78 times more expansive than one 3×3-convolution [36].
Lastly, Google combined the two concepts of inception units and residual units to create the InceptionResNet v1 and v2. The main idea here is to add an identity connection around the inception unit. Three different InceptionResNet modules, A, B and C, exist, they mainly differ in the number and the size of performed convolutions.They are depicted from left to right in Figure 4.8. Further those architectures have a stem at the beginning of the model and reduction blocks to adapt the sizes between the different modules. The exact details can be found in [34].

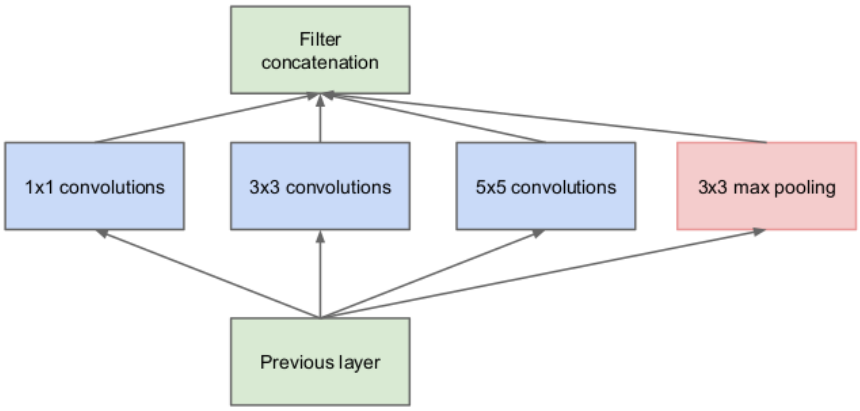


Figure 4.7: The sketch of a naive implementation of the basic inception unit is shown. Four different operations are performed simultaneously. The results are concatenated before passing them on to the subsequent unit [35].

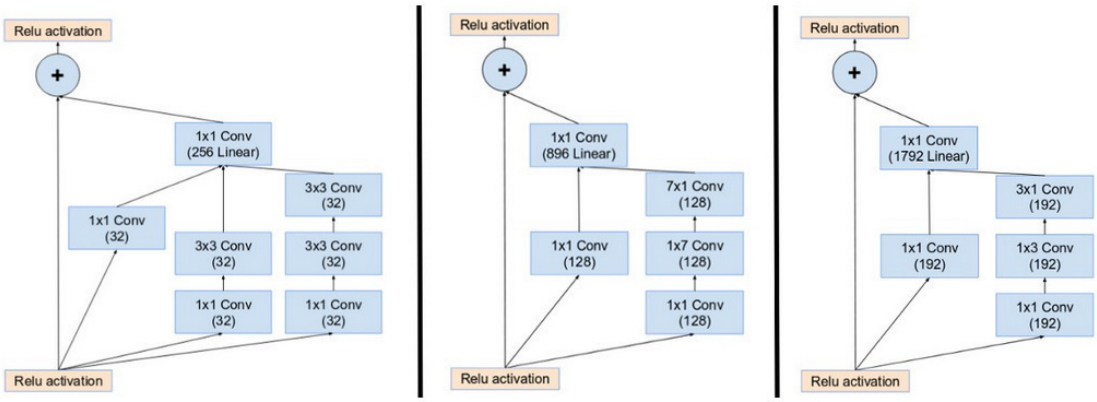


Figure 4.8: The three different modules A, B and C of the InceptionResNet v1 are depicted from left to right [34].

4.4 Multi-Task Learning

In multi-task learning, short MTL, the neural network performs several tasks simultaneously instead of only one. For example a network may classify an animal shown in a picture while reconstructing the direction in which the animal is looking at the same time.

One way of MTL is to share a large part of the neural network, which is therefore forced to learn a more general representation of the given problem, and then have task specific layers on top. This technique is called hard parameter sharing [8]. A schematic sketch of MTL using hard parameter sharing is shown in Figure 4.9.

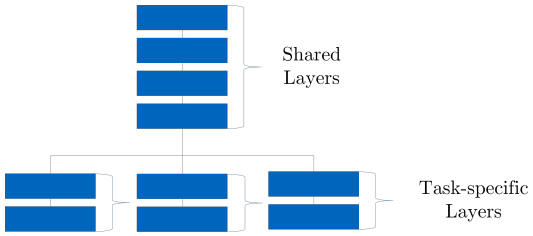


Figure 4.9: General idea of MTL: the neural network consists of shared layers, layers that are common to all tasks, and specific layers, that only belong to a specific task. The shared layers learn a general representation of the problem, while the task specific layers are supposed to add the features needed for the specific task.

As loss functions are usually defined task specific, we need a new way of handling the loss for the network in MTL. To train for several tasks within one neural network at once, we need to combine the single loss functions to one main loss. A very simple way to do so is by adding up the single losses and introducing weights for each loss component. In order for this aggregated loss function to stay in the same order of magnitude as a single loss function one can further divide the sum by the number of aggregated loss functions. By that we get

$$L_{main} = \frac{1}{N} \sum_{i=1}^{N} w_i L_i, \tag{4.9}$$

where N is the number of single loss functions, w_i the weight corresponding to the loss function L_i.

4.5 Confusion Matrix

A good way to analyze classification tasks is to use so-called confusion matrices. In the following we want to introduce them and discuss how they can be interpreted.

The potential solutions of each classification task belong to a finite set of classes: each event originates from one class and is classified by the neural network as one class of this finite set. Therefore it is possible to create a matrix with one axis corresponding to the ground truth and one to the predictions. The size of the matrix will be $n \times n$, where n denotes the number of classes.
In case the predicted class and the ground truth coincide, an event will be sorted into a diagonal entry of the confusion matrix. Thus, in general, a classifier works fine if the matrix has the majority of its entries on the diagonal axis, because this means that the events have been classified correctly.
Let's consider an exemplary image classification task in which we distinguish between images of cats, dogs and birds. This task results in a confusion matrix of size 3×3. One possible event is an image of a dog. Let's assume the network falsely classifies the dog as a cat. The corresponding matrix and the sorting of the case are sketched in Figure 4.10.

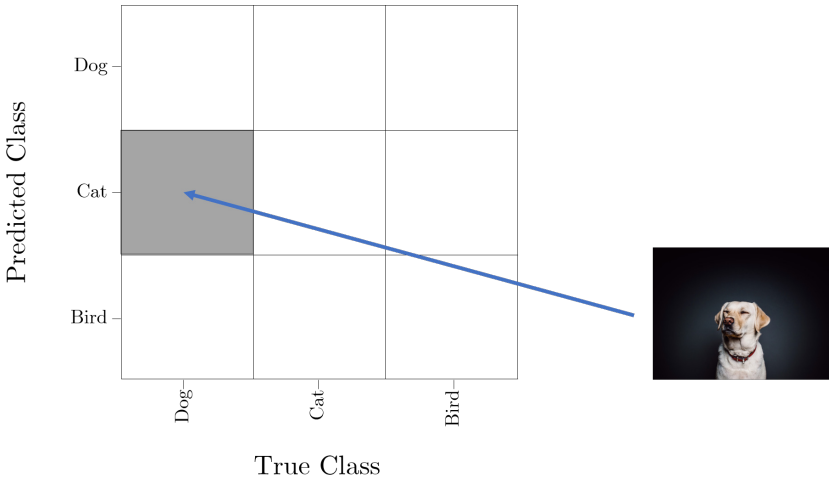
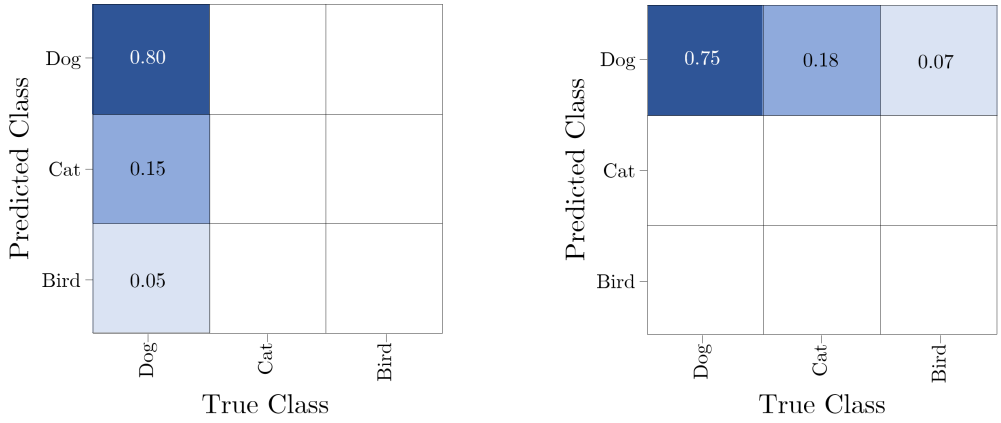


Figure 4.10: A exemplary confusion matrix and the sorting of one specific image is illustrated. The image shows a dog which was classified as a cat.

The final matrix represents the total number of events and their corresponding combinations of ground truth and prediction. For easier interpretation the confusion matrix can be normalized. There are two different ways to do so: first one can normalize on the ground truth, motivated by the fact that each event has to belong to one class. Second, one can normalize on the predictions, as each event has to be predicted as one of the classes. The normalized confusion matrices for our example classification task are schematically shown in Figure 4.11. The confusion matrix normalized on the ground truth values can be interpreted as how many of the actual events of this class were found. We will call this fraction of correctly



(a) Confusion matrix normalized on the ground truth

(b) Confusion matrix normalized on the predictions

Figure 4.11: The difference between the two normalization methods are illustrated. On the left, the confusion matrix is normalized on the ground truth. The confusion matrix on the right is normalized on the prediction.

identified events the *accuracy*. The confusion matrix normalized on the prediction values in contrast can be seen as the percentage of predictions which were actually correct for this class. The fraction of correct predictions is also called the *precision*. Let's look at our example again to clarify the difference between the two normalization methods: a very simple algorithm that predicts always dog independently of the input, will classify all given dogs as dogs. It achieves an accuracy of 100% for the class dog . This seems nice at the first glance, but the precision of these predictions will be bad. It will equal the fraction of dogs inside the dataset. If the classes are distributed equally, this will result in a precision of 33.3%. Therefore one should always use both normalization methods to interpret the results of a classification task.

CHAPTER 5

Specification of the Dataset

In this chapter we introduce our dataset. First we discuss specifics of the IceCube data especially for their use with neural networks. Further we define the labels of all tasks needed in the training process. The baseline of the dataset and the event selection are outlined next. Finally the properties of the resulting dataset are described and the impacts of the dataset composition are discussed.

5.1 Specifics of IceCube Data for Neural Networks

In its rawest form IceCube data for one event consists of multiple charge over time measurements, one for each launched DOM. The DOMs are arranged in a three dimensional grid. The input data is therefore four dimensional. This leads to two main issues: first the dimensionality of the data and second the large size of the data.
High dimensional data results in many weighted connections in a neural network which in turn makes it hard to train a stable model on it. Additionally a four dimensional convolution method would be needed, which is not standardly available yet. Further there are doubts if they are even numerically stable. Hence other approaches are needed.
The second challenge we face is the sheer size of the data for a single event. Using the smaller ATWD digitization each lunched DOM has a time series of 128 measured charges. For 5160 DOMs this results in $5160 \times 128 = 660,480$ inputs for a single event. Training such a network is quite complex and in addition a lot of sample events are needed.
One approach to deal with this situation is the use of recurrent neural networks (RNN). They do not solve the second problem but deal with the time component more naturally. Due to a harder numerical implementation and a more fundamental strategy we did not use them. A second approach is the use of three dimensional convolutions on beforehand calculated features. The second option was chosen as this will also further narrow down the data size. The details on the calculated features will be explained in Section 5.1.1. Further the three dimensional grid was rearranged to narrow down the data even more. This is described in Section 5.1.2.

5.1.1 Input Features

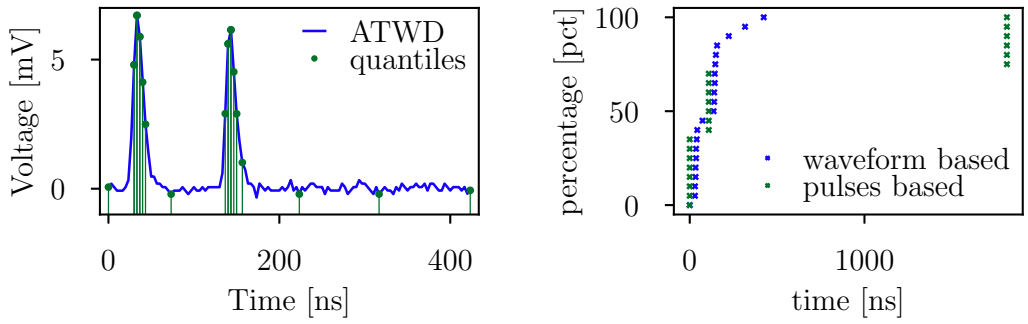
By calculating specific features, used for input we are reducing the amount of data. These features can than be stacked together similar to the RGB values in image

classification tasks. The input size is therefore reduced to $5160 \times n$, where n is the number of input features. Chosen input features will be described in detail in the following.

Quantiles

To achieve less input values the discretization of the waveform is downsampled. One could for example take only every second data point, this would equal a longer sampling time. Or the combination of several points for example by averaging them is another possibility. In this work we used a different approach. To get more detailed information in interesting areas, we utilized charge quantiles. More precisely, time information at which a given percentage of the total charge of each DOM was collected was calculated. An illustration of this digitization is shown in Figure 5.1a. A problem when using the waveform is that due to jitter it also has negative charge entries. To calculate the quantiles we therefore handled all charge entries as absolute values. When using pulses instead of a waveform this problem disappears automatically including all further characteristics jitter may introduce. A comparison once using the waveform and once using pulses is shown in Figure 5.1b.

A problem when using the ATWD information alone is that only a small time window of the event is covered. Hence the pulses can contain more information, often these are stored in a second ATWD series. At first we only used the first ATWD series. This difference can also seen in Figure 5.1b where the quantiles based on pulses show more late entries, corresponding to a pulse which is not in the ATWD series. A further divergence is introduced because the pulses were used without their width, therefore all entries of one pulse are at the same time. Also the distortion introduced by jitter vanish. This can be seen best at the three latest quantiles based on the ATWD waveform.



(a) Charge Quantiles

(b) Comparison of quantiles

Figure 5.1: On the left the ATWD waveform of Figure 2.5b is shown again. The times calculated as charge quantiles are marked. On the right a comparison of the calculated input times once based on the ATWD waveform and once on the pulses is depicted.

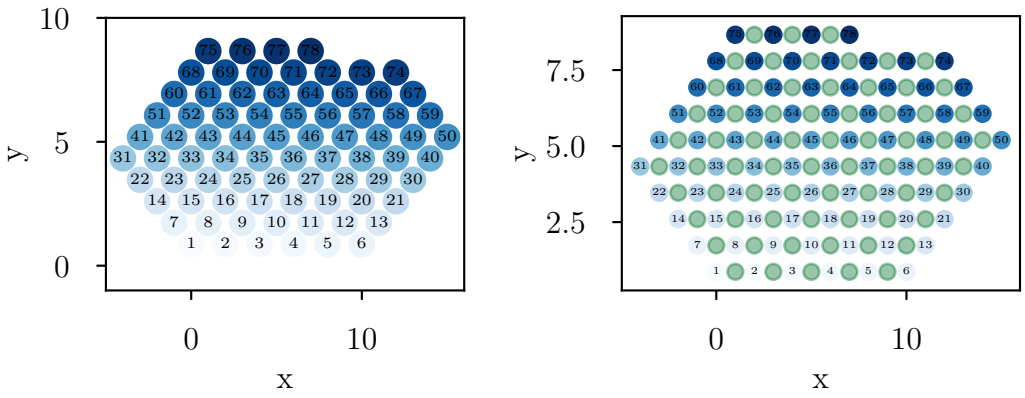
Additional Features

Below some additional features are listed. They have been tested in training as complementary input to the charge quantiles and alone. In the final architectures only the total charge per DOM was used.

- Total charge
- Time of the first charge
- Number of pulses
- Height of the first charge
- Time spread
- ...

5.1.2 Grid

The grid of IceCube, as described in detail in Section 2.1, is irregular. A schematic sketch of the top view on the strings of the InIceArray is shown in Figure 5.2a. For simplification DeepCore is excluded for now. To reduce complexity for the neural network and to work efficiently, a compact representation of the grid has to be found, while several requirements need to be fulfilled. First, neural networks work with matrices. In the three dimensional case this is a cube. The cube should be as small as possible while at the same time maintaining as much spatial information as possible.
To stick with the exact position of each string, as in Figure 5.2, we need a grid of the size $20 \times 10 \times 60$. This representation introduces permanent vacancies inside the grid. They are an additional characteristic the neural network has to learn to cope with. However by this representation the neighborly relations of the strings are represented in more detail. This situation is sketched in Figure 5.2b.

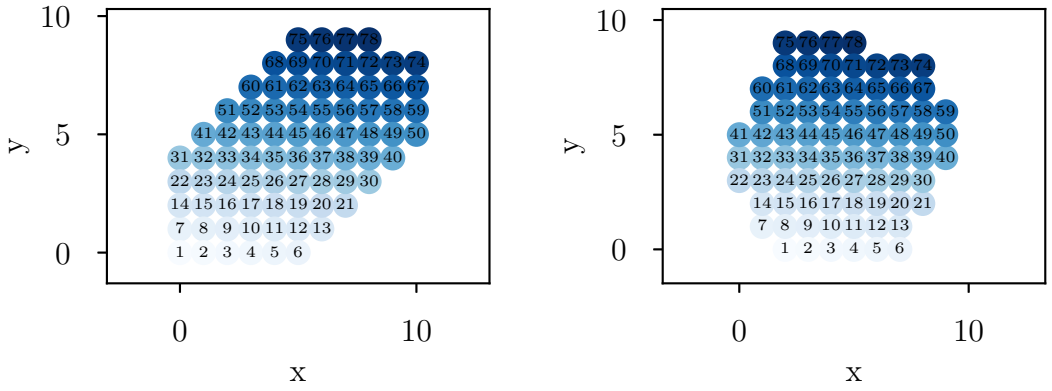


(a) Schematic IceCube grid out of the top view

(b) Same grid than in Figure 5.2a but the vacancies are highlighted

Figure 5.2: A sketch of the top view of the IceCube strings is depicted. The grid was rotated so that the the horizontal lines match with the direction of the x-axis. DeepCore is excluded. On the right side additionally the vacancies are highlighted by the green dots.

A more compressed representation of the grid can be found if we shift strings in the x-direction, the y- and z-direction can thereby stay the same. We came up with

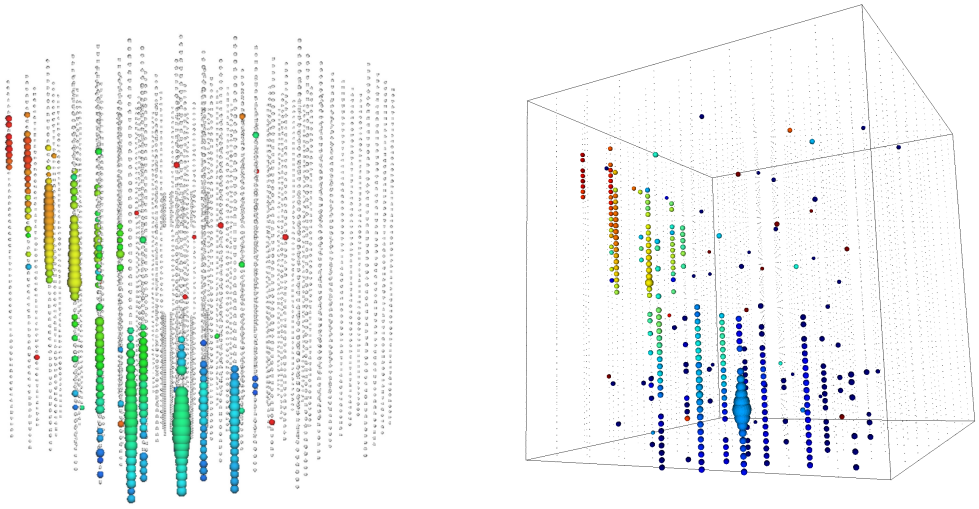


(a) Version A

(b) Version B

Figure 5.3: The top view of the rearranged grid in version A and B are shown. Version B is the smallest possible representation while maintaining the neighbor ship relations of the single strings better than version A.

two different solutions for the compressed representation which are both depicted in Figure 5.3. For version A the strings are rearranged in a way that the lower left side, strings 1, 7, 14, 22 and 31 are mapped to the same x-value on the outer left. The same is performed for the upper right border, strings 50, 59, 67, 74 and an imaginary string 80, are pushed to the outer right. This way the horizontal lines are pushed to their corresponding sides to match at the outside. The grid shrinks to the size of $11 \times 10 \times 60$. The direct neighbors of each strings are nearly fully maintained. If we push the horizontal lines of strings in an alternating pattern to the left and right it result in version B. In comparison to version A the grid decreases even further to the size of $10 \times 10 \times 60$. While direct neighbors are equally well preserved in version B as in A, the relation between farer strings improves further. As an example lets look at the three strings 1, 14 and 16. In IceCube, Figure 5.2, string 1 is equally far away from the strings 14 and 16, this holds also for version B, Figure 5.3b, but not for version A, Figure 5.3a.
In Figure 5.4 a comparison of one concrete event once based on the original IceCube grid and once based on the grid version B is illustrated. The event based on the new grid, Figure 5.4b, can be clearly identified as the corresponding event, Figure 5.4a. The differences in the color coding are handmade and indicate no difference. A slight difference can be seen in the cascade in the lower layers, in Figure 5.4b, based on the grid version B, the cascade seems a bit more starched to the sides.
For the rest of this thesis we will use grid version B as it is most effective in reducing the size and keeps at the same time a maximum of detector symmetry.



(a) Original IceCube Grid (b) Grid Version B

Figure 5.4: A comparison of one event once based on the original IceCube grid, shown on the left, and once based on the grid version B, depicted on the right, is shown.

5.2 Label Definitions

In supervised learning the ground truth, the correct output to the given problem is of main importance. It is responsible for the feedback the neural network gets during the training process. For each task a network should learn, a specific corresponding label has to be defined.
In our case the network will learn to predict a label corresponding to a specific event type. In the following section it will be explained how labels are assigned to the different event types.

5.2.1 Event Topologies

To assign the labels for the different event topologies we followed a two-level process. The events were at first distinguished as fine as possible. In a second step we recombined them into the classes that actually should be distinguished by the classifier.
On the finest level, we differ between the following event types:

- Neutral Current Interaction (NC)
- Cascade
- Starting Track
- Stopping Track
- Through-Going Track
- Double Bang
- Stopping Tau
- Glashow Cascade
- Glashow Track
- Glashow Tau

The event types have been explained in more detail in Section 2.4 as well as Chapter 3. To distinguish between the above mentioned types the information provided

Event Classification	Label
Neutral Current	Cascade
Cascade	Cascade
Through Going Track	Track
Starting Track	Starting Track
Stopping Track	Track
Double Bang	Double Bang
Stopping Tau	Double Bang
Glashow Cascade	Cascade
Glashow Track	Track
Glashow Tau	Cascade

Table 5.1: The allocation of the ten different event types to the different labels is shown.

in the Monte Carlo simulations for each event was exploit. The corresponding decision process is sketched as a decision tree in Figure 5.5. Here we classified the events based on their type of interaction, their flavor of the produced lepton and their eventual further decay. Finally the events signature inside the detector was deciding.
The second step, the recombination is necessary because the different event types can't be distinguished well enough on the finest level. Which labels are assigned to which event type is summarized in Table 5.1.
We further relabeled according to two additional criteria not included in Table 5.1. Events labeled as starting tracks that have a track length inside the instrumented volume of less than 50 m were relabeled as cascades. Also events labeled as double bangs were relabeled according to their tau decay length. All events with a tau decay length of less than 5 m were also labeled as cascades.

5.2.2 Further Classes

Simultaneously to the event topology class the events can also be classified in further categories. Their definitions are explained below.

Starting

An event is classified as a starting event if the vertex of the event is inside a predefined volume. This volume is normally the volume of the InIceArray, but it can be changed by padding it's surface. If nothing else is mentioned no padding was used.

Coincident Events

An event is classified as a coincident event if a second primary particle is hitting the detector inside the trigger window.

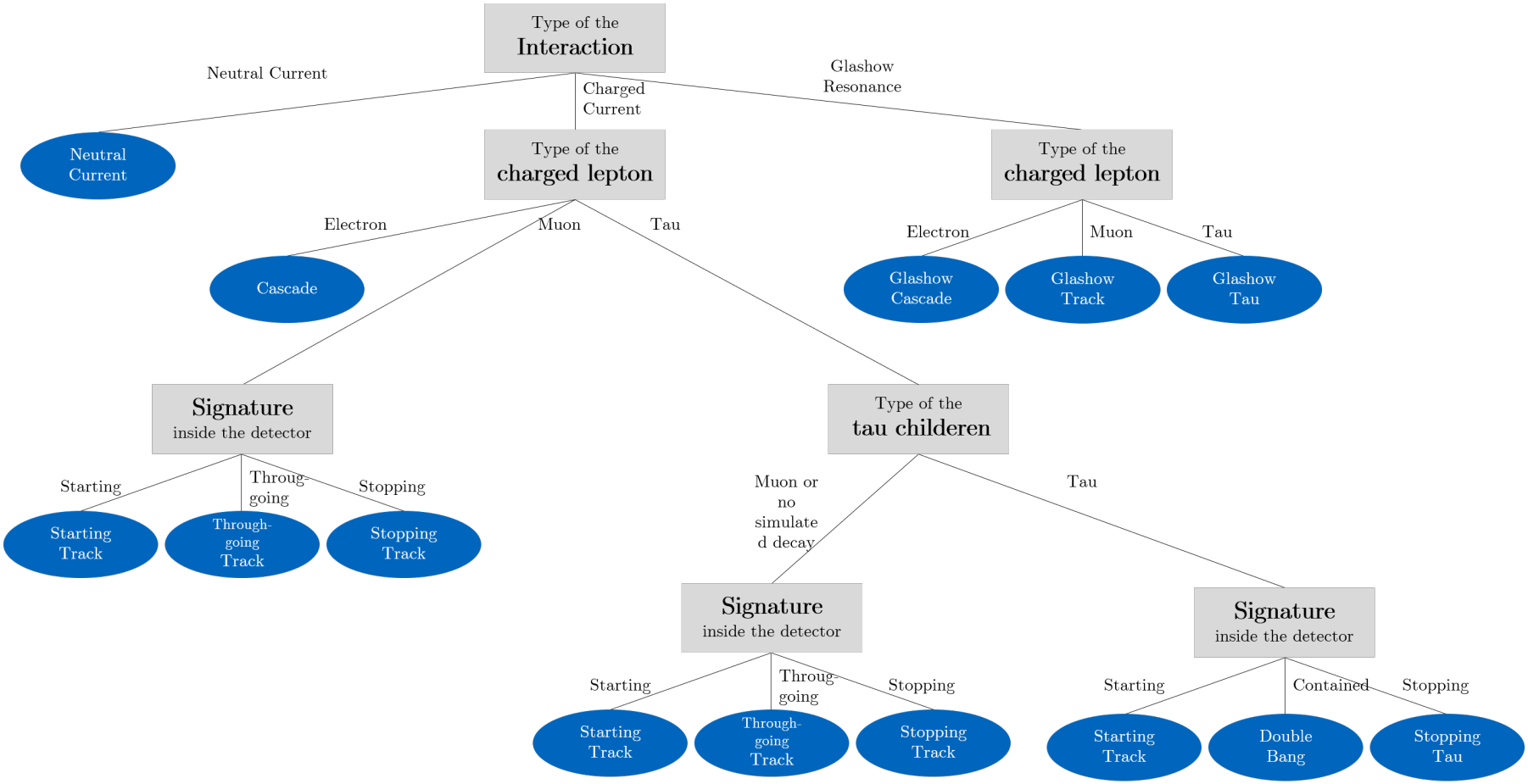


Figure 5.5: A visualization of the decision tree used to distinguish the different event types is illustrated. The grey boxes depict a split, on the axes the respective characterization is shown. The blue oval nodes depict the event types on the finest level. First we consult the interaction type, followed by the type of the charged lepton. For charged current interactions producing a tau its decay is examined. In the last stage, the signature of the event inside the detector gives the information about the final event type.

5.3 Properties of the Monte Carlo Simulation

To obtain our dataset we used Monte Carlo simulations produced by Nancy Wandkowsky for the IceCube collaboration. To be clear which simulations were used, the directories of the respective simulation files at the IceCube cluster in Madison are listed in the Appendix A.1. The corresponding simulation parameters are given in Table 5.2. The simulation was done for all three flavors. Around 15% of the events have a coincident CORSIKA background. In this thesis we use a Monte Carlo with energies between 5 TeV and 50 PeV. The data was prepared with IceCubes processing up to level 2.

Parameter Name	
Generator	NuGen
Photon Propagator	Clsim
Ice Model	Spice 3.2 [2]
Hole Ice Model	Dimas flasher-fit-model (p1=0.3, p2=0)
DOM Efficiency	0.99

Table 5.2: The parameters used for the simulations are presented.

5.4 Event Selection

Only events where at least one simulated particle hit the instrumented volume were selected. Also events where all particles missed the detector closely, such that light still could be seen, were cut. Thereby we obtained well contained event topologies, to not confuse the classifier at first.
Additionally the composition of the dataset was adjusted, events depending on their label were selected. Goal of this procedure was to have a dataset with good properties for training. Thereby the dataset should be as big as possible, while maintaining a suitable ratio between the different labels. A two-level procedure was found to be very useful. First a large base dataset is created. In this step all time-consuming calculations, like the adaption to our grid or the calculation of the various input features, were done. In a second step a further selection is performed. The goal is to obtain a fine tuned dataset for training. As the work on this thesis showed it is helpful to have the opportunity to change the final composition of the dataset in a reasonable time. Therefore this two-level approach is good, because the expansive computations are not needed over and over again. For fine tuning only the second selection step needs to be done again.
The percentages of each event class that were kept in the first step are listed in Table 5.3. The full base dataset consists of around 6,500,000 single events. In the second step we again applied filters that only kept specific percentages of each class, these are also listed in Table 5.3. For true neutral current events and true cascades a variable percentage was used. It depends on the primary energy of the neutrino. The goal of this variable percentage was to obtain more high energetic cascades

as a counterweight to the double bangs. Also for true double bangs we applied to different filters. For double bangs with a tau decay length smaller than 5 m only 1% was kept, while for all longer ones we kept all. The resulting distributions can be found in the following Section 5.5.

Event Classification	Step 1	Step 2
Neutral Current	21 %	$(24.75 \times \log_{10}(\text{Energy}) + 35.25)$ %
Cascade	18 %	$(24.75 \times \log_{10}(\text{Energy}) + 35.25)$ %
Through Going Track	9 %	20 %
Starting Track	35 %	20 %
Stopping Track	10 %	20 %
Double Bang [$< 5m$, $\geq 5m$]	[100, 100] %	[1, 100] %
Stopping Tau	100 %	100 %
Glashow Cascade	100 %	100 %
Glashow Track	100 %	100 %
Glashow Tau	100 %	100 %

Table 5.3: The percentage of each event type we kept in our dataset in the first and second selection step are presented.

5.5 General Properties of the Dataset

5.5.1 Training, Validation and Test Set

The whole dataset consist out of 2,841,632 independent events. We further divide the set into a training, validation and test set.

Set	Number of Events	Percentage of the Dataset
Training Set	1,989,142	70 %
Validation Set	426,245	15 %
Test Set	426,245	15 %

Table 5.4: Composition of the whole dataset according to its purpose in the training process.

5.5.2 Event Distributions in the Dataset

The number of events per class in the dataset on the finest level are not equally distributed, as can be seen in Figure 5.6. The distribution which results from relabeling the classes according to Table 5.1 are shown in Figure 5.7. The same distributions for the starting and coincidence label are shown in the Figure 5.8a and Figure 5.8b, respectively.

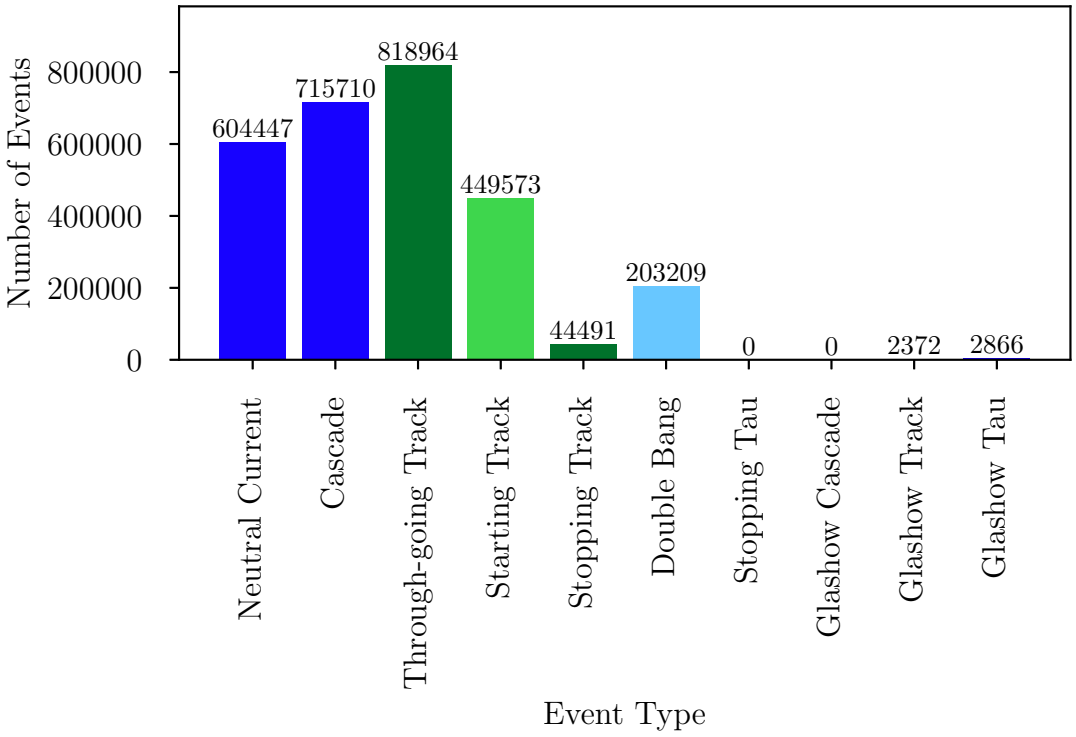


Figure 5.6: The distribution of the event type labels on its finest level is shown.

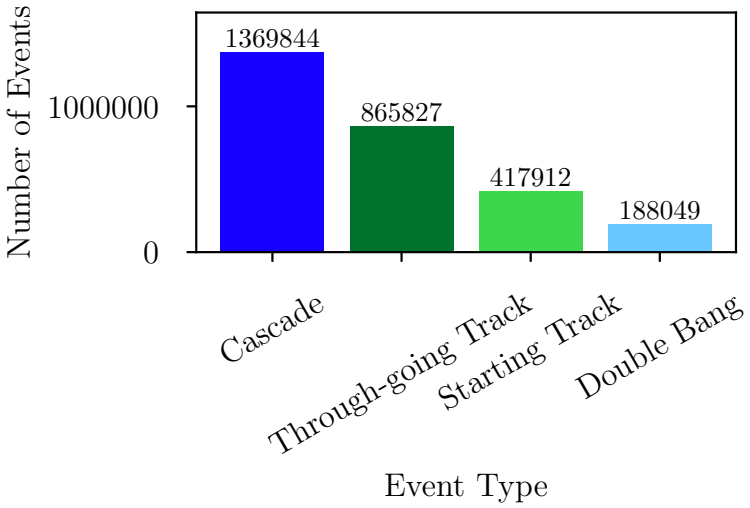
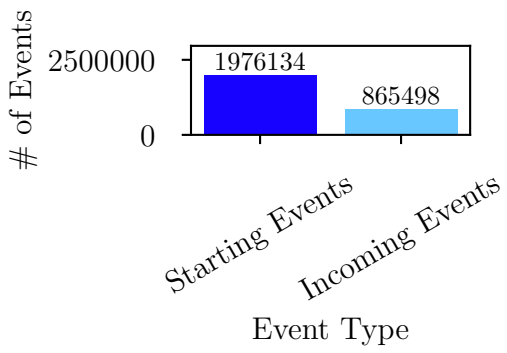
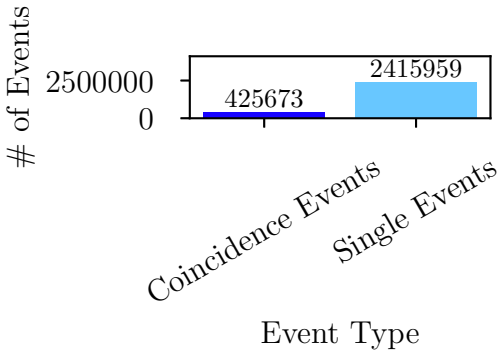


Figure 5.7: The same distribution as in Figure 5.6 after relabeling the events, according to Table 5.1, is presented.



(a) Starting Label

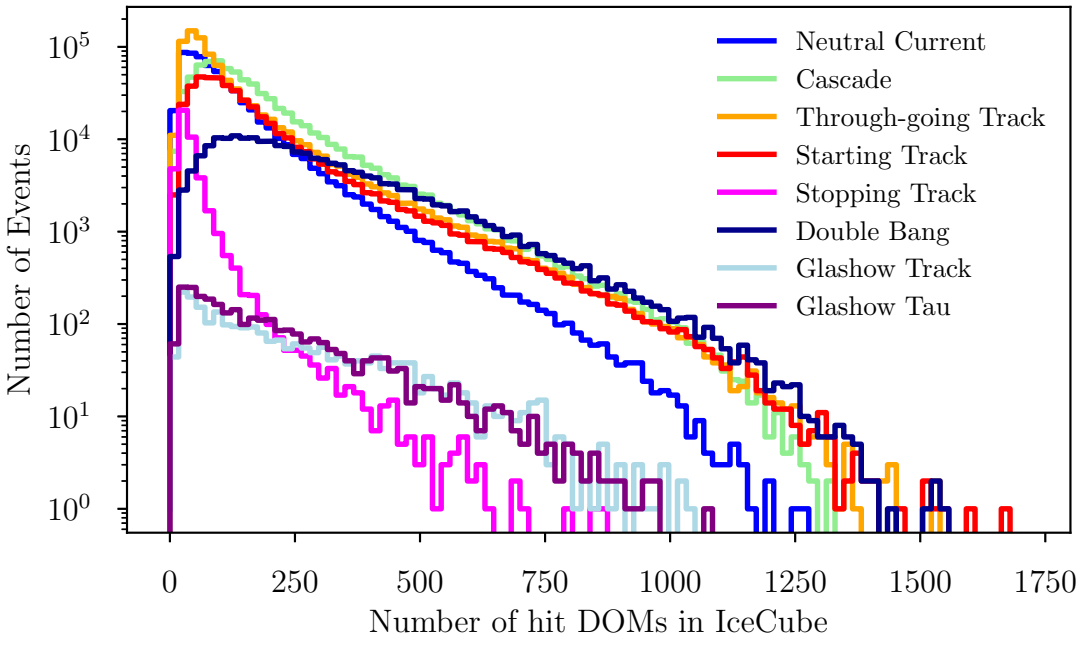


(b) Coincidence Label

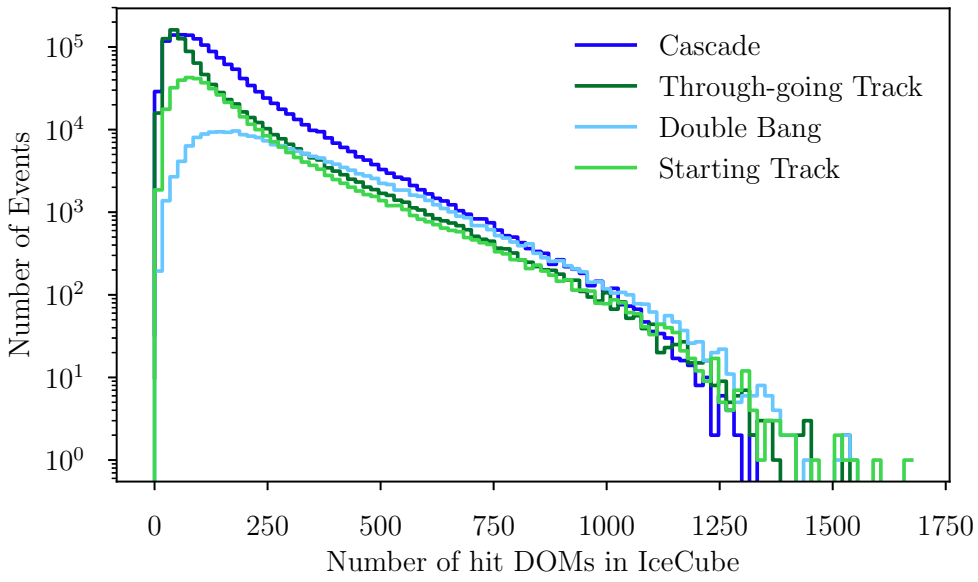
Figure 5.8: The distributions of the starting and coincidence label of the whole dataset are plotted.

5.5.3 Distribution of Physics Parameters

The distributions of the most important physical parameters of the dataset will be shown in the following. For most quantities the distribution is shown twice, once for all event types on the finest level and once for the event types after relabeling. For the event types on the finest levels "Stopping Taus" and "Glashow Cascades" can not be found in the distributions, because there are no such events in the dataset, as can be seen in Figure 5.6.
First the number of events in the dataset against number of hit DOMs in the InIceArray is presented, in Figure 5.9. Additionally for all track-like events the track length inside the detector is shown in Figure 5.10 and for all events involving a tau, the tau decay length is presented in Figure 5.11. Further distributions, including the number of events against the deposited energy, the inelasticity, the zenith and azimuth and the energy of the first particle can be found in the Appendix A.2.



(a) All Event Types



(b) Event Types after relabeling

Figure 5.9: Distributions of the whole dataset against number of hit DOMs in the InIceArray once split for all event types and once for the event types which should be classified.

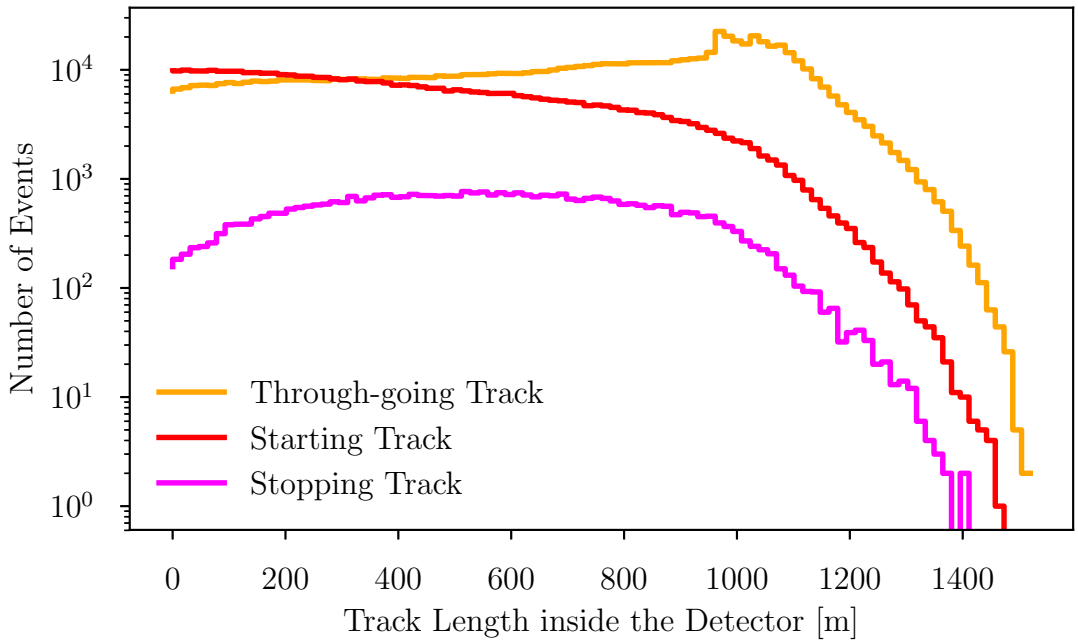


Figure 5.10: The track length inside the detector for all track-like event types is shown.

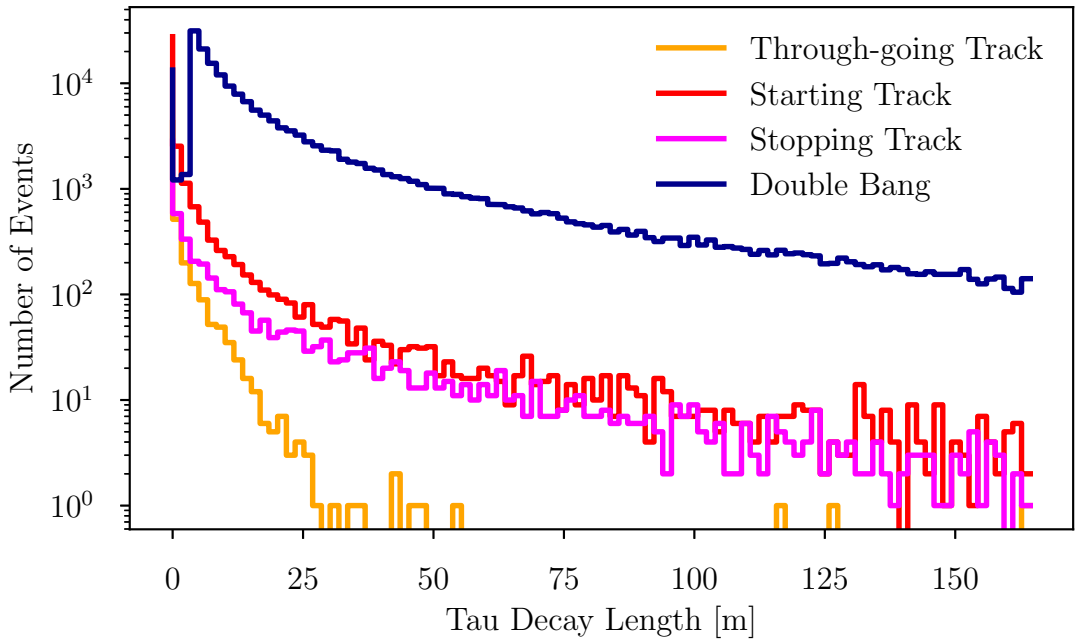


Figure 5.11: For all events including a tau the corresponding tau decay length is shown.

5.6 Impact of the Dataset Composition

The impact of the dataset composition on the final classification results can be significant. Therefore we first discuss how an optimal dataset should look like and which effects may introduce biases. In the following, we will outline the experiences we gathered during the work on this thesis with respect to the underlying dataset.

From a theoretical point of view an optimal dataset should be as diverse as possible. That means, every aspect that should get learned should be represented in as many different ways as possible. Thereby a good generalization becomes achievable and every detail that is needed to perform the given task can be learned. Further, the dataset composition should not introduce any external bias. A bias can for example be introduced by having an imbalance in the number of events per class. Furthermore, the classes among themselves should be similarly distributed in quantities, that can effect the classification. Last the dataset should be sufficiently large, the more events are available the better.

The first thing we encountered was that we had defined our labels to roughly. The label definitions led to events that basically looked the same but had different ground truths. As a result the network got confused as it was shown the same data but told that the events belong to different classes. We saw a tremendous boost in defining the ground truth more carefully.
In a next step we forced minimal requirements for specific classes. If they were not fulfilled we relabeled those events. Two examples should clarify our relabeling strategy: we forced a minimal track length of 50 m inside the detector for starting tracks if not they were labeled as cascades. Secondly, for an event to be labeled as a double bang a tau decay length of more than 5 m was necessary, if not they were also relabeled as cascades. This step can be seen critical, due to the change of the ground truth for difficult events only. On the other side the relabeling can be seen as a further tuning of the class definitions as it is nearly impossible to distinguish these event topologies. The events relabeled are close to the resolution of IceCube.
In a third run, we additionally forced a nearly equal distribution of labels with regard to the event type class. Thereby we wanted to avoid any bias to any specific class. However, this wasn't sufficient for our task. Due to the characteristics of the classes biases were still introduced. For example, double bangs got over predicted in case many DOMs had been hit in the detector while no cascades got predicted for these cases. On the other hand cascades got heavily over predicted for few hit DOMs. By enforcing the same amount of cascades and double bangs in the dataset, the number of true double bangs was a lot higher than the number of true cascades in the regime of many hit DOMs. Therefore, we guess that a more uniform distribution of the events with respect to the number of hit DOMs for each class would be beneficial for the classification task we face.
In a final step we corrected the distribution of events in hit DOMs within one class to be more uniform. This has only been done for the class of cascades to obtain more of them in the regime of many hit DOMs. By doing so we provided the network with more training events in the regime important for the distinction

of double bangs and cascades, where the network had previously struggled. Due to time reasons no further dataset was created. Hence extending this approach to the remaining classes and forcing an even more similar distribution among the classes offers potential for further improvements in the future.

CHAPTER 6

The Classifier

In this chapter the final version of our classifier is discussed. The term classifier represents the deep neural network that is trained to perform our different classification tasks. The chapter starts by introducing its architecture and the training process it was passed through. Further the results of each classification task are presented individually. The classifier performs an event topology classification, a starting event identification and a coincident event identification simultaneously. Generally the results of the classifier will be shown in form of a confusion matrix. The concept of a confusion matrix and its normalizations is explained in Section 4.5.

6.1 Specification of the Classifier

6.1.1 Input

As described in Section 5.1.1, we used charge quantiles calculated in 5% steps as input to the network. Individual quantiles are stacked behind each other. Finally we appended the total charge measured at each DOM. Thereby the neural network can easily distinguish between hit and non hit DOMs. Each quantity is arranged according to the grid version B, which was described in Section 5.1.2. Hence this results in an input of size $10 \times 10 \times 60 \times 21$. Further, each input quantity was normalized to have a mean of 0 and a standard deviation of 1.

6.1.2 Architecture

The architecture of the final neural network used is based on the IncResNet v2 of Google [34]. An introduction to the architecture was given in Section 4.3.2. More comprehensive information can be found in [34], [35] and [36].
After the input layer we first perform five three dimensional convolutions each paired with batch normalization and two MaxPooling operations. Google calls this first part of the neural network the stem. It is a simplified version of the stem used by Google in [34], which was extended to three dimensions. Exact details on the amount of used filters, their size and other parameters can be found in the Appendix A.3.1.
The middle part of the network consists of one Inception-A block, followed by seven Inception-ResNet-A blocks and one Reduction-A block. Additionally seven Inception-ResNet-B blocks and one Inception-ResNet-C block are performed. Each one similar to the version presented by Google in [34]. The main difference is the

change from two dimensional to three dimensional convolutions.
Because we perform three tasks simultaneously we have to copy this status of the network three times for the final task-specific parts. Each task performs one additional Inception-resnet-C block and one final three dimensional convolution. For this final convolution the event type classification uses 1024 filters while the starting identification and coincidence identification use 320 filters each. All tasks end with an AveragePooling. This setup results in a total of 27,460,504 parameters. A sketch of this setup is shown in figure 6.1.

Development of our Architecture

Our final architecture as described above evolved over time. This paragraph gives a short outline of this development that led to the final architecture.

We started based on the results of Johannes Kager in [21]. He had shown that for the purpose of event reconstructions in IceCube deep convolutional neural networks are more suitable than conventional feed-forward neural networks. Also Mirco Huennefeld already applied deep learning techniques to reconstruct muon neutrino events in IceCube [17].
Many different versions of convolutional networks were tested. A broad exploration on the effects of different hyperparameters was performed. Investigated parameters included the size of the network, different input features, different loss functions, optimizers and the extend of using different regularization methods. During our tests we found that charge quantiles work well as input features. Furthermore, the best results were achieved by using Adam as an optimizer. The use of residual units led to a more stable architecture and training.
In a next step, considerable improvement was achieved by the introduction of multi-task learning. This led to an increase in the overall accuracy due to a greater generalization.
Our next modification of the architecture was inspired by Google's InceptionResNet v2 [34]. By using InceptionResNet blocks we achieved a better overall accuracy while training a lot faster. The time needed to run one epoch decreased by a factor of three. In addition, physical tendencies as we would expect them were matched better. In the following of this thesis this is discussed in more detail.
The number of parameters in the final architecture is immense and training is therefore prone to overfitting. The size of the current neural network was chosen by simple testing. Architectures with more or less InceptionResNet blocks showed no significant improvement.
Potential improvements and alternatives to the described architecture will be outlined in the outlook in Section 8.2.

6.2 The Training Process

For the training process of our network we used the Adam optimizer and batch normalization for regularization, as introduced in section 4.2.2. Adam was used

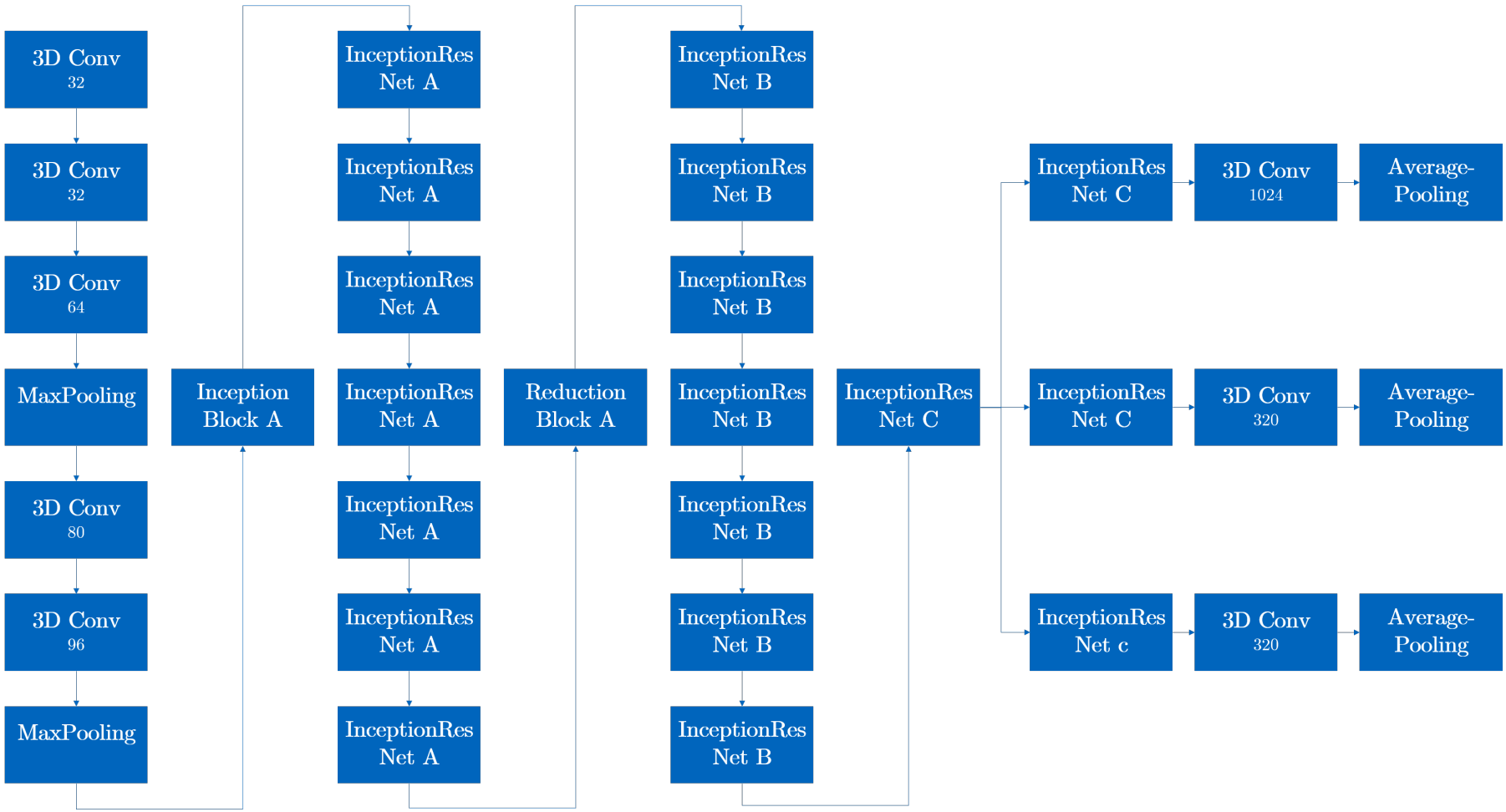


Figure 6.1: The architecture of the classifier is sketched. The network consist out of three main parts: the stem, a block of seven Inception-ResNet-A modules and a block of seven Inception-ResNet-B modules. They are connected by an Inception A module and later by a Reduction A block. Before the split into task-specific layers one Inception-ResNet C module is performed. For each task one further Inception-ResNet C module and one three dimensional convolution as well as an AveragePooling layer is finally added. The small number for convolutional layers represent the number of filters used.

with its default parameters. Our choice of all further hyperparameters can be found in the Appendix A.3.2.
In one epoch the full trainings dataset is used. It takes about 33 h on three GPUs (GeForce GTX 1080 Ti).
For the evaluation of the training process we distinguish between a main loss, that includes all tasks and task specific losses. To calculate the main loss, according to Section 4.4, all tasks were weighted equally by $w_i = 1$ and N was chosen as well as 1. Each loss is further spitted into a trainings loss calculated for the trainings set and a validation loss calculated based on the validation set. Figure 6.2 shows a summary of all losses. A more detailed view of each task-specific loss is illustrated in the Figures 6.3a, 6.3b and 6.3c.

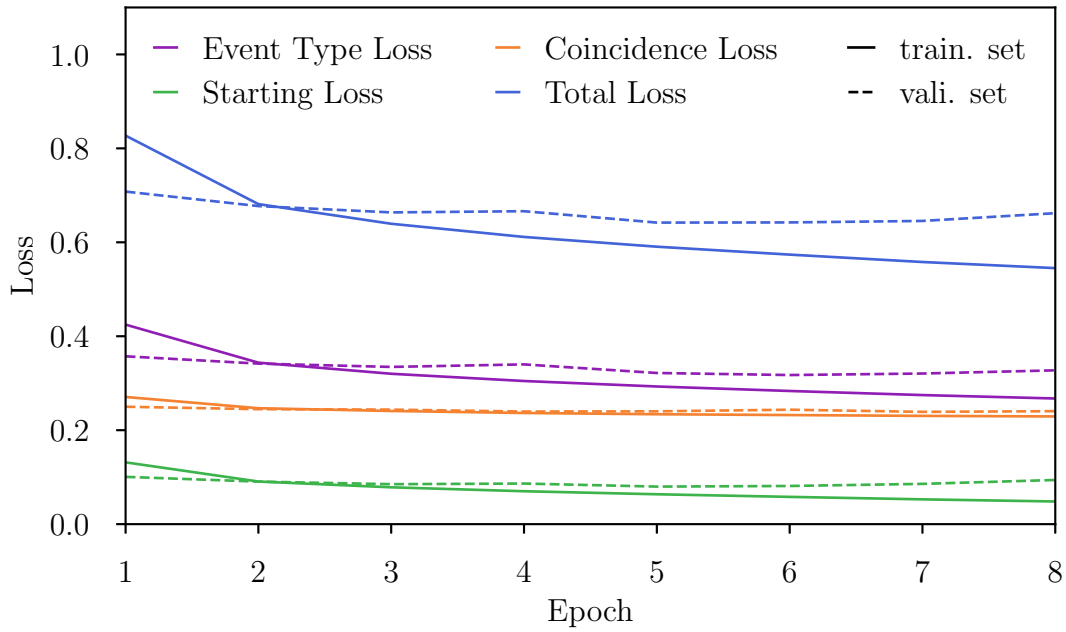
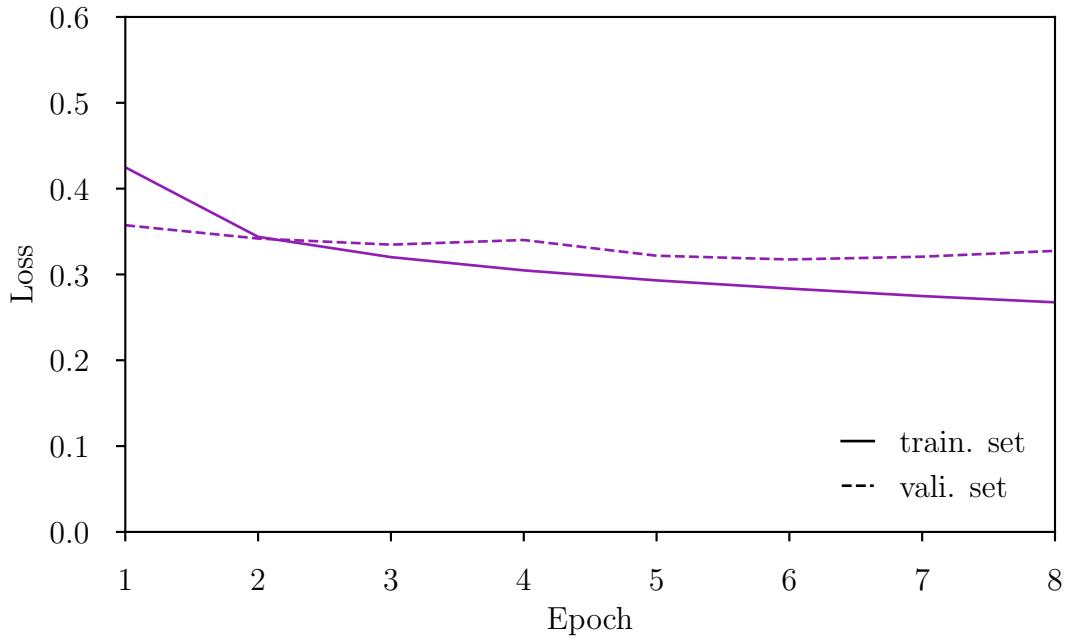
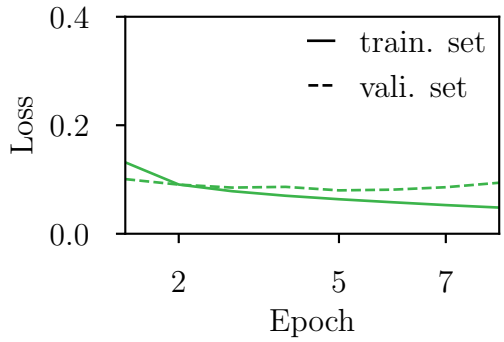


Figure 6.2: The losses of training and validation set for the main loss and all individual tasks are shown.

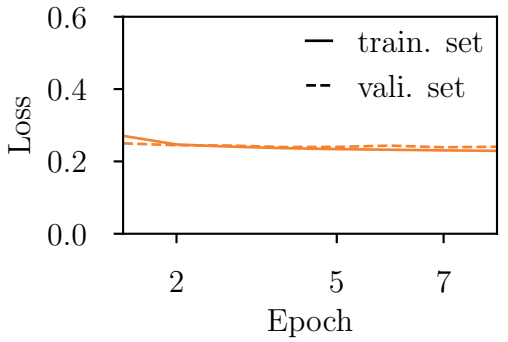
Figure 6.2 shows a rapidly decreasing main loss in the beginning. The main training loss and the main validation loss already reach the same level after the second epoch. In the next epochs the main trainings loss decreases further, while the main validation loss stays relatively constant, with only slight changes. The losses diverge more and more. Starting at epoch eight we see an increase of the main validation loss which is a clear sign of overtraining. This effect will get stronger over time.
The task-specific losses show similar behavior during training as the main loss, including the start of the overfitting. The event type loss contributes most to the main loss, the starting loss least. The coincidence loss is nearly flat, which could result from the unbalanced composition of the ground truth data in the dataset.



(a) Loss of training and validation set for the event type classification only



(b) Loss of training and validation set for the starting identification only

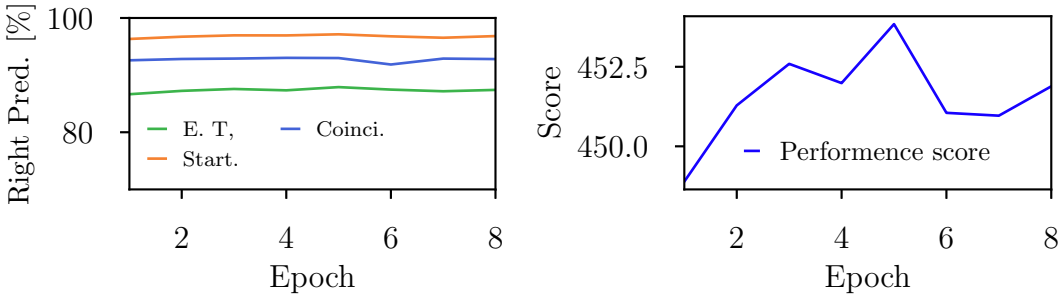


(c) Loss of training and validation set for the coincidence identification only

Figure 6.3: Validation and training loss are presented once for each performed task: on the top the event type classification losses are shown. On the bottom left the losses for the starting identification are depicted and on the right the ones for the coincidence identification.

6.2.1 Choice of the Final Neural Network

The decision upon which version of the neural network is finally used is normally based on the main validation loss. Due to the fact that the individual tasks are not equally important to us, this does not hold here. We also did not adapt our main loss function to weight the task accordingly. Instead we defined our own "performance score" based on the percentage of predictions that were correct. The percentage of correct predictions was calculated for each task. The results are sketched in Figure 6.4a. Afterwards the percentages were summed up. However, as the event classification is most important to our evaluations, its percentage was weighted with a factor of three. The resulting score for each epoch is shown in Figure 6.4b.
Our best classifier achieved a performance score of 453.84. It uses the learned parameters after five epochs. We use this network for all results discussed in the following.

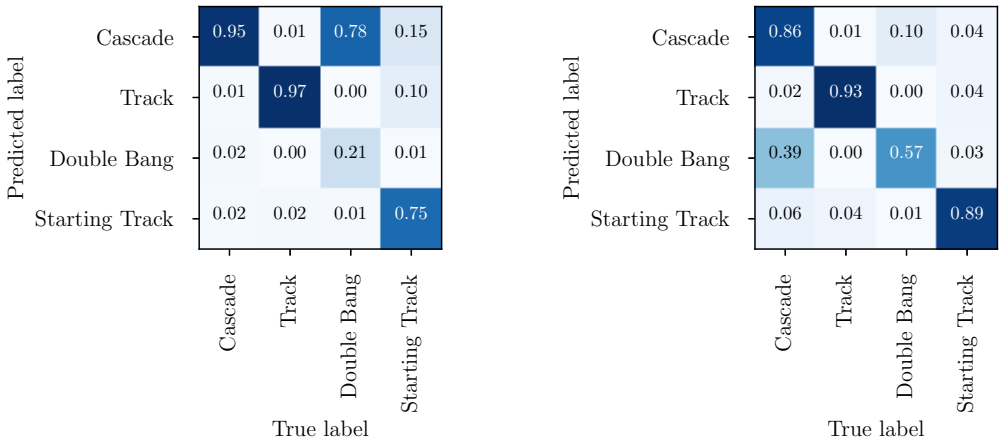


(a) Percentage of right predictions per task (b) Performance score over all epochs

Figure 6.4: The percentages of correct predictions per task over the epochs is shown on the left. On the right the resulting performance score is plotted.

6.3 Event Type Classification

In our event type classification task we want to distinguish four event topologies in IceCube: cascades, tracks, double bangs and starting tracks. The event topologies and exact label definition have been explained in Section 2.4 and Section 5.2.
Figure 7.7a and Figure 7.7b show the confusion matrices of the event type classification once normalized on the ground truth of each class and once normalized on the predictions of each class.



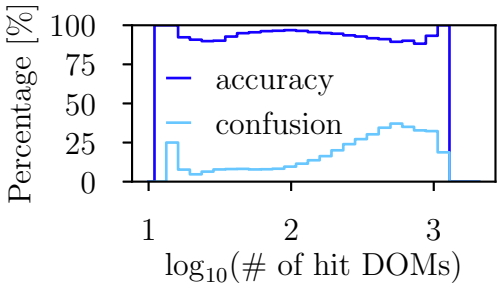
(a) ground truth normalized (b) prediction normalized

Figure 6.5: The confusion matrices of the event type classification once normalized on the ground truth of each class and once normalized on the predictions of each class are presented.

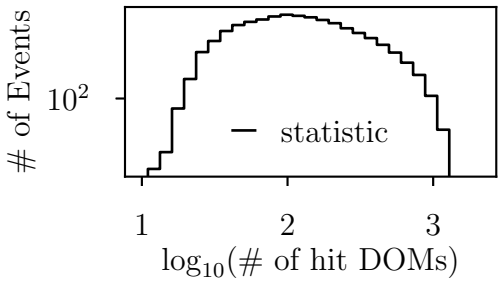
The network achieves best results for the track class, 97% of them are found with a precision of 93%. Nearly all their confusion is with the class of starting tracks. Cascades are similarly well classified with an accuracy of 95% and an associated precision of 86%. While the confusion with tracks is negligible, this isn't the case for double bangs and starting tracks. Only 21% of the double bang events are classified as such and a prediction of a double bang is correct in 57% of the cases. Double bangs are heavily confused with cascade events as double bang events are classified as cascades in 78% of the cases. 75% of the starting tracks are identified correctly with an according precision of 89%. Starting Tracks are confused with two classes, tracks and cascades. In the following we will look at the different confusions in more detail and investigate possible reasons for their occurrence.

We generally expect that events with more hit DOMs are easier to classify as events with very few hit DOMs. This follows the simple intuition that if more has been seen by the detector, more information is available to distinguish the events. The accuracy and confusion as a function of the number of hit DOMs of the detector are shown in Figure 6.6. Accuracy represents the fraction of correct identified events of the corresponding class. In contrast, the confusion is the false positive rate, the percentage of events that got falsely predicted as a specific class. Additionally the number of events against the number of hits DOMs is shown.

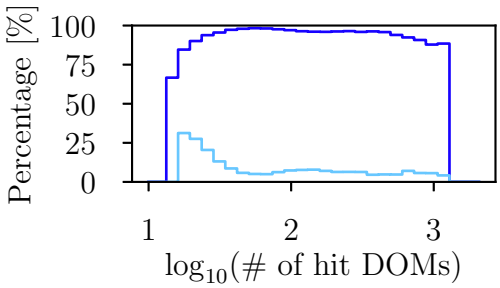
It can be seen that the accuracy for cascades stays at nearly 100% over the full regime. However the confusion is starting to rise counter-intuitively from 100 hit DOMs with an increase in hit DOMs. This confusion comes from double bangs which occur much more likely at large numbers of hit DOMs, see Figure 6.6f. The accuracy for tracks is also constantly high except for small impairments for both very few and many hit DOMs. The track confusion has its peak for few hit DOMs. It decreases the more DOMs are hit before rising again at the end of the



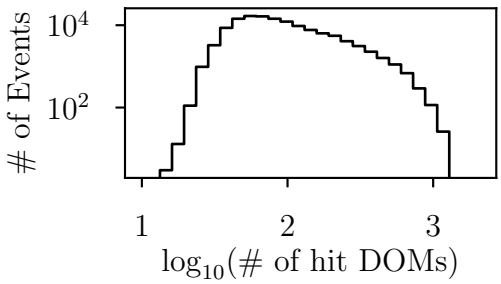
(a) accuracy and confusion of true cascades



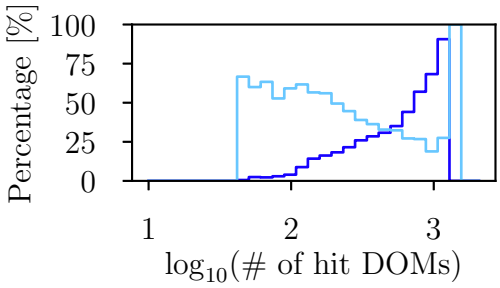
(b) statistic of true cascades



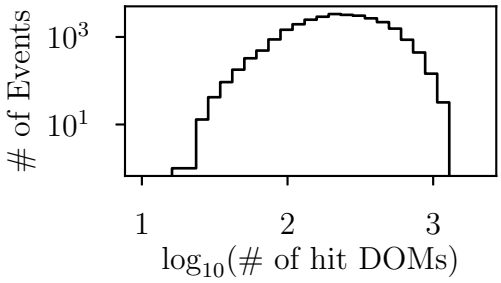
(c) accuracy and confusion of true tracks



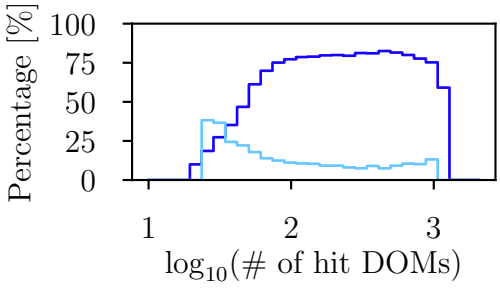
(d) statistic of true tracks



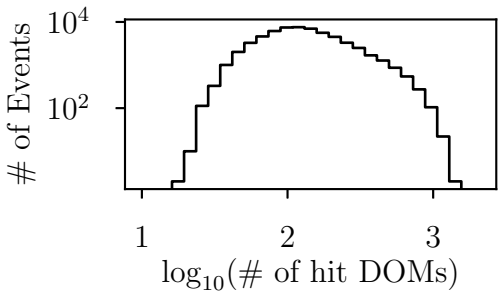
(e) accuracy and confusion of true double bangs



(f) statistic of true double bangs



(g) accuracy and confusion of true starting tracks



(h) statistic of true starting tracks

Figure 6.6: The accuracy and confusion as a function of the number of hit DOMs together with the underlying statistic for each event type is presented. The accuracy, confusion and statistics of each bin are shown in dark blue, light blue and black respectively.

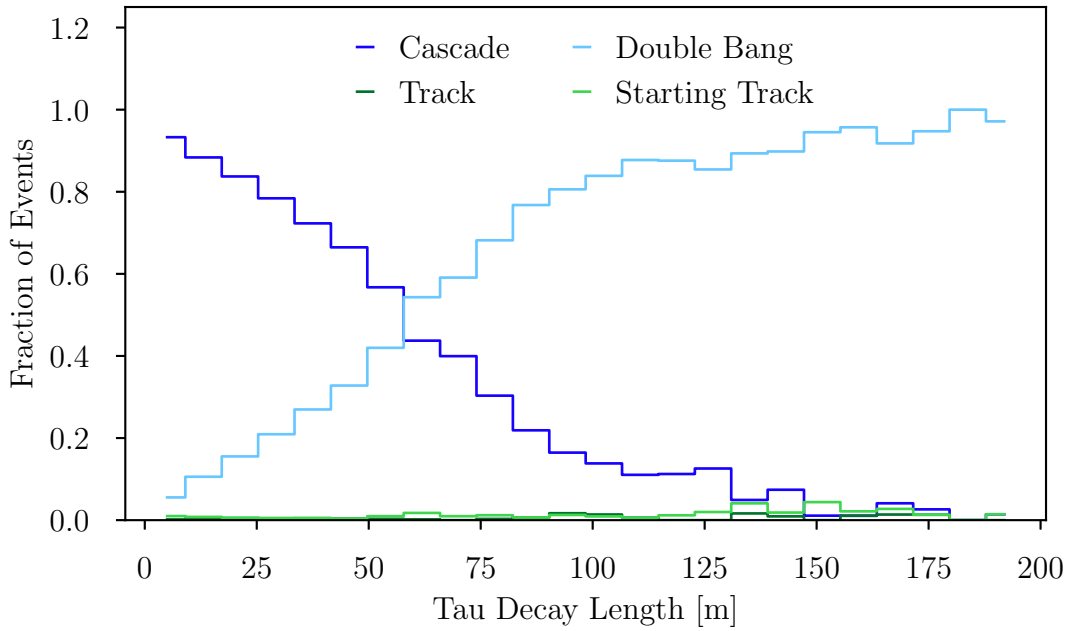


Figure 6.7: For true double bang events the prediction of each class as the fraction of all events as a function of the tau decay length is shown. Only tau decay lengths up to 200 m are plotted.

interval. Double bangs are predicted best if many DOMs are hit. The accuracy rises continuously for an increase in the number of hit DOMs as the confusion decreases accordingly. In the last bin the confusion increases strongly again, however the statistic is also very low there. The classifier predicts nearly no double bangs below 100 hit DOMs correctly. Starting tracks have a rising accuracy for few hit DOMs. Above around 100 hit DOMs it reaches a plateau of around 75% at which it stays. The confusion is highest at the beginning, decreases further and increases slightly again for around 1000 hit DOMs due to their confusion with double bangs. The exact distributions can be found in the corresponding subfigures in Figure 6.6.

We further expect to see a tendency that double bangs are in general more easily predicted for higher tau decay lengths, as the two cascades are better separated from each other. For very small tau decay lengths we expect a confusion with cascades, due to the small separation of the two cascades, which can therefore look like one. In Figure 6.7 we show the predictions of the classifier of all events with the true class double bang as a function of the tau decay length. As expected the confusion with cascades dominates the low regime completely. For tau decay lengths of about 60 m, the fraction of events predicted as cascades and the correct predictions are in balance. The confusion constantly decreases for larger tau decay lengths until it becomes negligible starting at a tau length of 150 m. Double bangs show no significant confusion with neither tracks nor starting tracks over the whole regime.

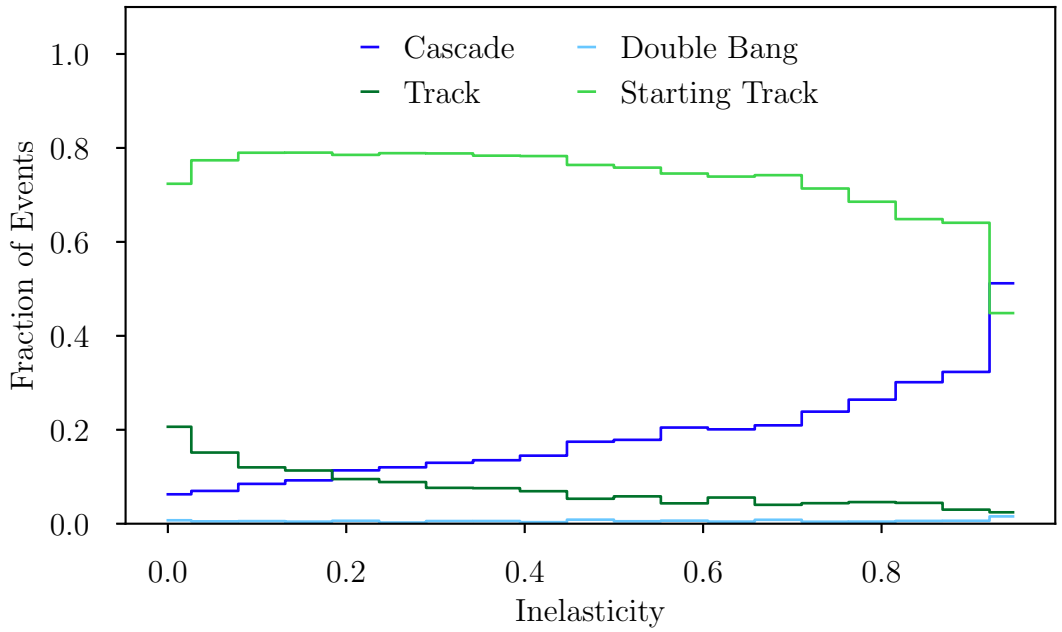


Figure 6.8: The fraction of predictions for each class for true starting tracks as a function of the inelasticity is presented.

Starting tracks are confused with tracks and cascades. For events with the ground truth starting track 15% of the events are identified as cascades and 10% as tracks. In the case of starting tracks we expect to see a heavy correlation between the number of correctly classified events and the inelasticity. The inelasticity of an event is defined as the ratio between the energy that goes into the hadronic cascade compared to the total energy of the primary neutrino,

$$\text{inelasticity} = \frac{\text{energy of the hadronic cascade}}{\text{total energy of primary neutrino}}. \tag{6.1}$$

We expect a higher confusion of starting tracks with cascades for high inelasticities and a confusion with tracks for very low inelasticities. Analogous to Figure 6.7 we show all events with the ground truth starting track and their corresponding predictions as a fraction of all predictions against the inelasticity in Figure 6.8. Again as expected, the confusion with tracks has its maximum at a inelasticity of zero and decreases with increasing inelasticy. The confusion with cascades behaves exactly inverse. It reaches its maximum of over 50% of starting track events being wrongly classified as cascades in the last bin where nearly all energy goes into the hadronic cascade. The two expected tendencies, a higher confusion of starting tracks with cascades for high inelasticities and a confusion with tracks for very low inelasticities, are thus confirmed.

A further quantity that influences the prediction of starting tracks is the track length inside the detector. A starting track with its vertex at the border of the detector and an outgoing track is in general harder to identify than its counterpart with a track going into the instrumented volume. Figure 6.9 shows the classifiers

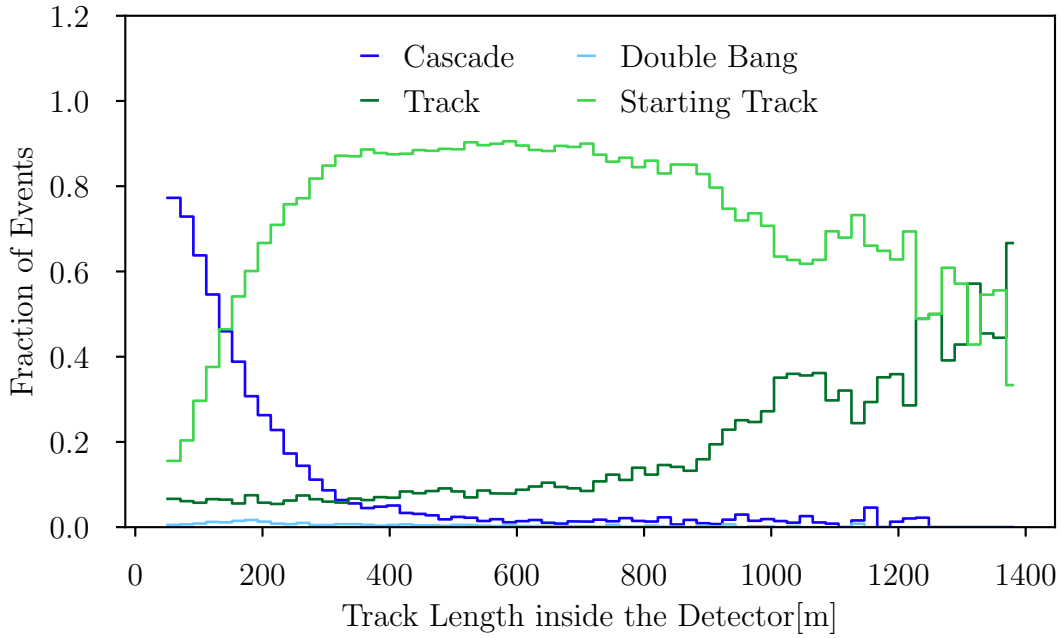


Figure 6.9: The fraction of predictions for true starting tracks as a function of the track length inside the instrumented volume is plotted.

predictions of true starting tracks against the track length inside the instrumented volume. A clear correlation between the track length and the confusion with cascades is observed: the shorter the track length the more likely a cascade is predicted. This effect dominates up to a track length of 150 m. The confusion with tracks shows a slight linearly increasing dependency up to the track length of around 800 m. Afterwards the confusion increases stronger up to more than 70% at the end of the spectrum.

As we have shown a dependency of starting track prediction on both the track length inside the detector and the inelasticity, this paragraph will examine the interplay between starting track predictions and both quantities simultaneously. Figure 6.10 presents the accuracy of predictions for true starting tracks as a function of the inelasticity and the track length inside the detector. The two beforehand observed tendencies are confirmed again together with some additional insights: starting tracks with a long track length are especially confused for very low inelasticities. The confusion with cascades for high inelasticity goes to far longer track lengths, what should have a positive effect, than for low ones. Events with an inelasticity over 0.9 and up to a track length of about 500 m are predicted on the same level as starting tracks with a track length of 200 m and an inelasticity below 0.3.

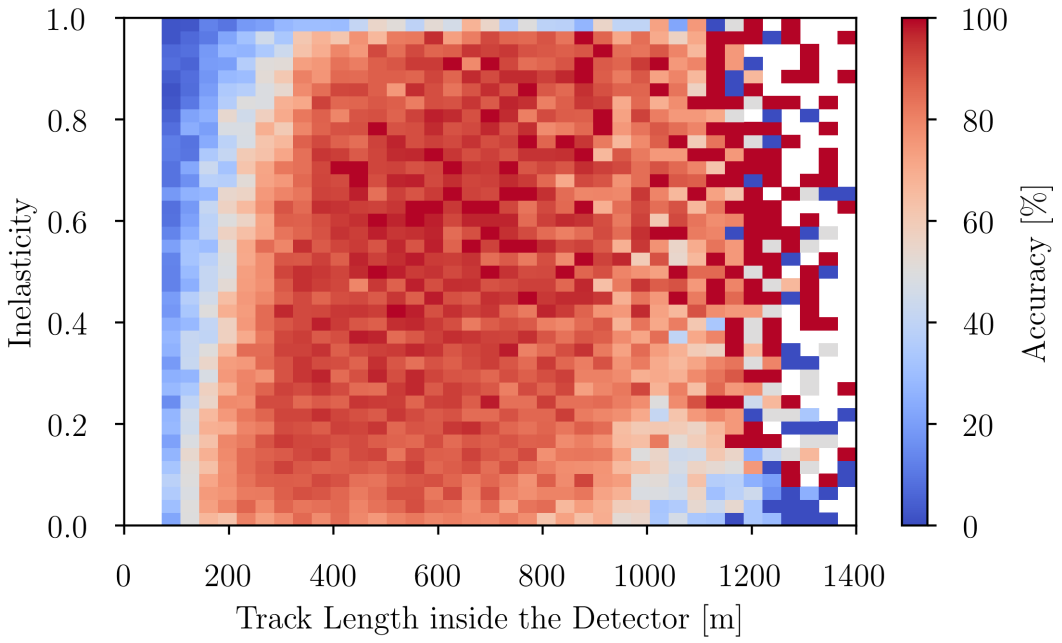


Figure 6.10: The accuracy of true starting tracks shown as a function of both the inelasticity and the track length inside the detector is presented.

6.3.1 Probabilistic Interpretation of the Neural Networks Output

Due to the use of the softmax activation function in the last layer of the neural network, compare Section 4.2.1, we can interpret the output of the classifier in a probabilistic manner. It is important to note that the output is not a real probability in the sense that an observed event will be of the corresponding class with this probability. Still it can be seen as a measure how confident the classifier is about his prediction. To confirm this correlation we first define a quantity we call the p-score. The p-score is defined as the maximum softmax output of one event. It is very important to not confuse this definition with the p-value in statistics. To use our p-score as a measure of certainty we require as minimal condition that the accuracy should increase if we only base its calculation on events that got predicted with an increasing p-score threshold. In other words the correlation is confirmed if for higher minimal p-scores higher accuracies are observed. Figure 6.11 shows the accuracy for each event type over the p-score threshold. For a certain p-score threshold of k, the accuracy is calculated based on all events that got predicted with an p-score higher than k.
It can clearly be seen that the upper requirement is satisfied. Therefore we have shown that we can use the p-score as a measure of certainty of the neural networks prediction. Moreover, we can see that predictions for double bang events are far more insecure than for the other classes, because for the same p-scores lower accuracies are observed. Following this logic tracks are predicted the safest followed by starting tracks. All event types converge to a accuracy off nearly 100% for a p-score threshold of 1, double bangs only converge to around 96%.

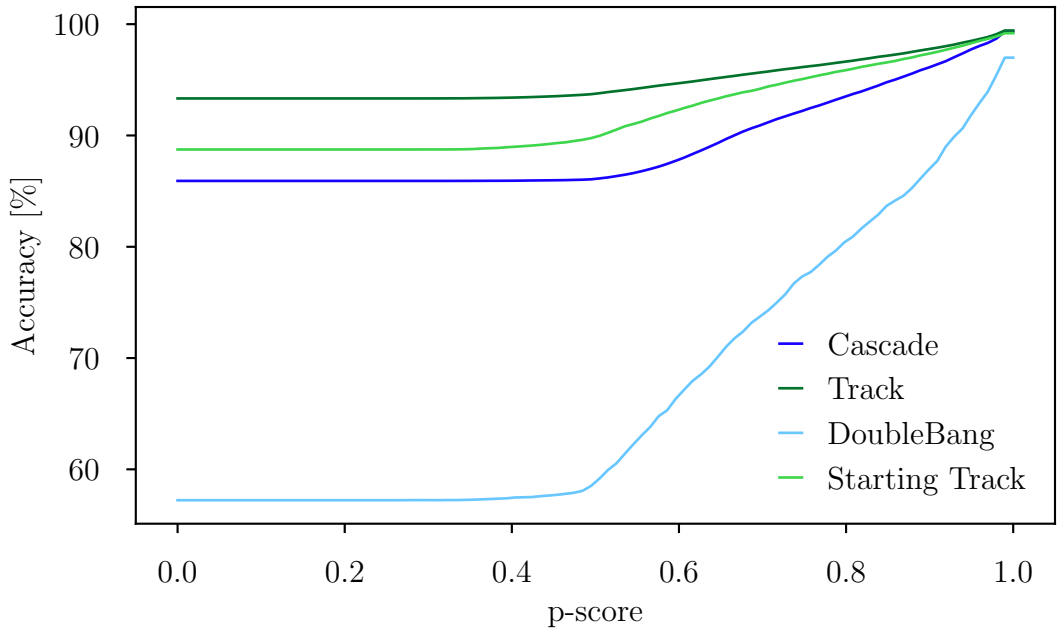
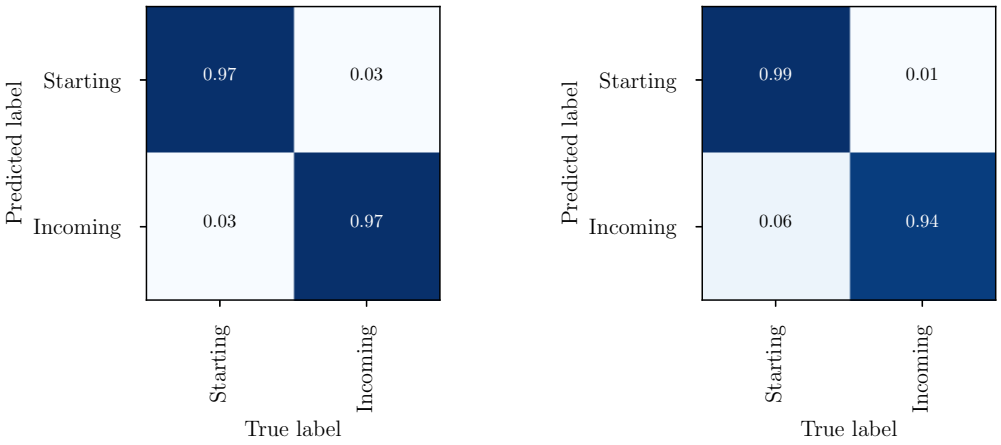


Figure 6.11: The accuracy calculated for all events that got predicted with a p-score higher than a certain p-score threshold is shown.

6.4 Starting Events Identification

This section focuses on the task to distinguish between incoming and starting events. Starting events are explained in Section 2.4 and the exact label definition can be found in Section 5.2. Here a padding of 0 m was used, so the volume in which the vertex of a starting event can be equals exactly the instrumented volume. Every event that enters the detector from outside is an incoming event. In the previously explained event type classification task this distinction was already used for track-like events, resulting in the two classes: tracks and starting tracks. Cascades can be of both types, but starting is more likely due to their smaller spatial extension. The same is valid for double bangs as they also consist of two connected cascades.
The classification results for the starting events identification are shown in the confusion matrices in Figure 6.12. 97% of the events were identified correctly. Starting events were found in 97% of the cases with an corresponding precision of 99%. Incoming events were identified with the same rate of 97% but with an worse resolution of 94%.

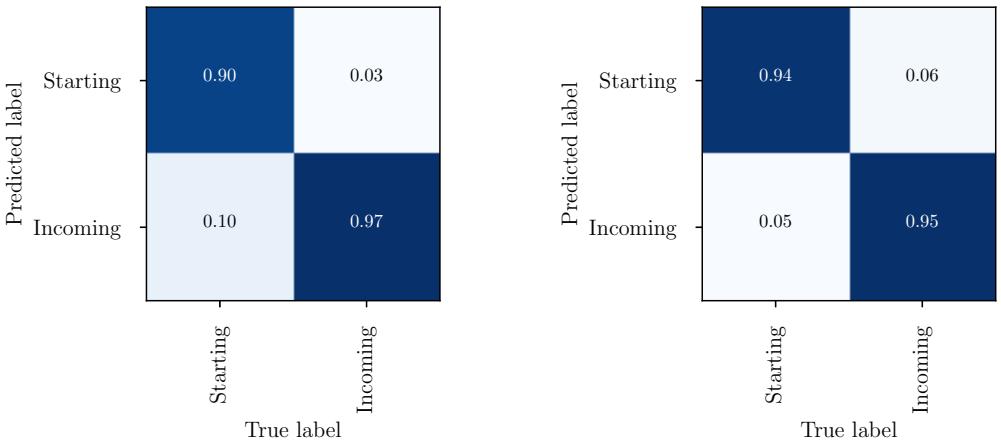
For the starting event identification the spatial expansion of the events plays a major role. Due to the large differences of the event types we show the confusion matrices again, once for all true track-like events and once for true cascades and double bangs. Due to their long track lengths, tracks are more likely to be incoming.



(a) ground truth normalized (b) prediction normalized

Figure 6.12: The confusion matrices of the starting classification once normalized on the ground truth of each class and once normalized on the predictions of each class are presented.

On the contrary, cascades are mostly starting. The confusion matrices obtaining only events with the ground truth track or starting track are presented in Figure 6.13.

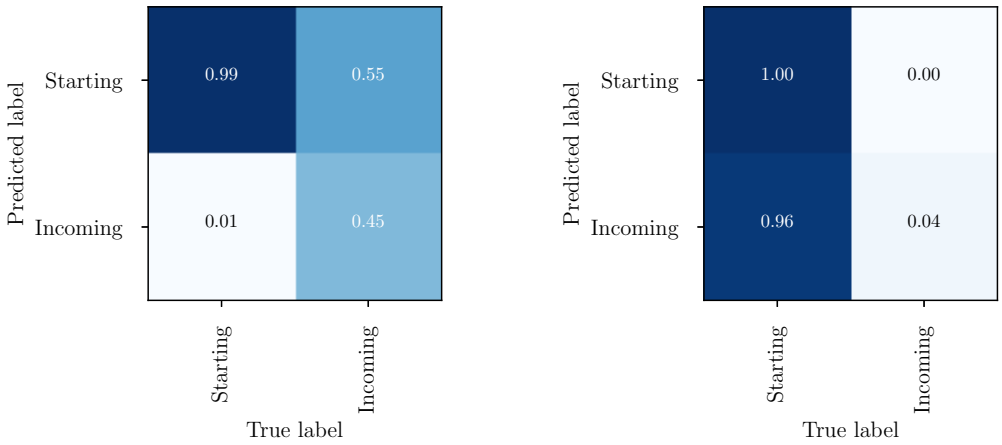


(a) ground truth normalized (b) prediction normalized

Figure 6.13: Confusion matrices of the starting event identification once normalized on the ground truth of each class and once normalized on the predictions of each class, only tracks and starting tracks are shown.

Comparing the confusion matrices of all events in Figure 6.12, and the confusion matrices of only track-like events in Figure 6.13, we can see a slight impairment of 7% for the starting class accuracy and no change for the accuracy of the incoming class. The corresponding precisions changed similarly: the starting class lost 5% in precision while the incoming class gained 1% in precision.

Figure 6.14 shows the confusion matrices again, now based on events that are true cascades or true double bangs. Here we see a completely different behavior. Starting tracks are found in 99% of the cases and their precision reaches full 100%. On the counter side incoming events are only detected with an accuracy of 45% while having a really bad precision of 4%. This results of the characteristics of the shown event that are mostly starting. In absolute values only 225 of the 236,090 events in the test set are incoming and true cascades or true double bangs. This is also the reason for the impairment of the starting class if only track-like events are observed. A huge fraction of the starting events are cascades, which are classified easier than track-like events.



(a) ground truth normalized (b) prediction normalized

Figure 6.14: Confusion matrix of the starting event identification once normalized on the ground truth of each class and once normalized on the predictions of each class, only cascades and double bangs are shown.

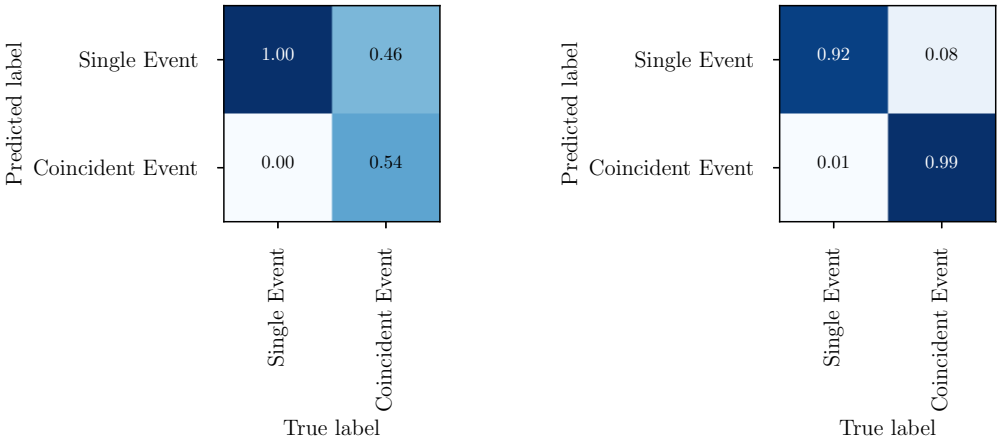
Finally we can state that starting event identification for track-like topologies works fine. On the other hand it is hard to distinguish cascades and double bangs in starting and incoming. Firstly, this results from their spatial characteristics and secondly, from their distribution in our dataset. A way to tackle this problem is to train a specialized network on a dataset especially created for this task. The dataset should be adjusted to contain a fair fraction of incoming cascades and double bangs, so that the network has the chance to learn from them.

6.5 Coincidence Identification

The coincidence identification task aims at finding events that have a second particle hitting the detector simultaneously. Coincident events are explained in Section 2.4 and the exact label definition can be found in Section 5.2.

Due to the rare appearance of coincidences in this dataset, around 15%, the prediction are strongly biased towards single events. As a result the loss will decrease rapidly to a relatively constant level just by predicting no coincidence independently of the event. This then results in a nearly constant loss after initial reduction, compare Figure 6.3c, where it is hard to improve on. The training of this task is therefore very slow.
In the Figures 6.15a and 6.15b the two normalized confusion matrices of this task are presented.

Single events are almost always correctly identified, their prediction is correct in 92% of the cases. Coincidences in contrast are only found, in 54% of all cases which is only slightly better as guessing. If a coincidence is predicted this is however correct in 99% of the cases. These results partly reflect the unequal distribution of labels. A possible solution for this problem would be an adapted loss function, that gives more weight to correct coincidence predictions than for single event predictions thus reinforcing their training. Another approach is the change of the label composition in the dataset. The optimal case for this task would be a nearly 50/50 split of single and coincident events. Nevertheless a 50/50 split will also influence the results of simultaneously learned task. For example, we expect that it is more difficult to predict the event topology correctly if a coincidence appears compared to an identical single event.

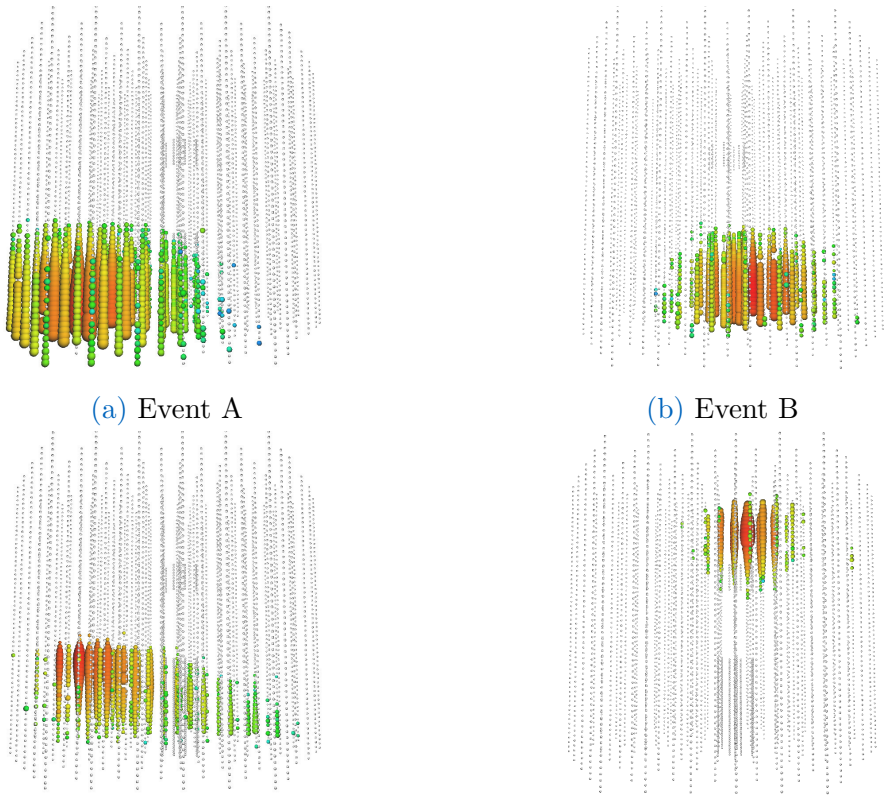


(a) ground truth normalized (b) prediction normalized

Figure 6.15: Confusion matrices of the coincidence identification once normalized on the ground truth of each class and once normalized on the predictions of each class are depicted.

6.6 Exemplary Events and their Predictions

In this section four misclassified example events as well as possible reasons for this results will be discussed. The events are presented in Figure 6.16.



(a) Event A (b) Event B

(c) Event C (d) Event D

Figure 6.16: Four exemplary event views are shown. Each event is misclassified.

Event A shows a true double bang caused by a tau neutrino with 6.2 PeV and a tau decay length of 28 m that is classified as a cascade. The effect that double bangs with small tau decay lengths get confused with cascades is shown in Figure 6.7.

Event B is a starting track with a track length inside the detector of 115 m and a inelasticity of 0.31. It is falsely classified as a cascade. The reason for this and the interplay of the two quantities and their effect on the prediction of starting tracks is illustrated in Figure 6.10.

Event C shows a track of 740 m in the instrumented volume which is classified as a starting track. This case is very rare as only 2% of true tracks are misclassified as starting tracks. This type of confusion is however still the main part of the total confusion for true tracks.

Event D shows an event with a coincidence that wasn't identified. The coincidence is located in the upper right corner and consists of only four hit DOMs. Simulta-

neously the event type was misclassified. While the true event type has been a double bang it was classified as a cascade, this is an effect of the small tau decay length as for event A.

More detailed information to each event can be found in the Table 6.1.

	Event A	**Event B**	**Event C**	**Event D**
True Event Type	Double Bang	Starting Track	Track	Double Bang
Pred. Event Type	Cascade	Cascade	Starting Track	Cascade
True Starting	Starting	Starting	Incoming	Starting
Pred Starting	Starting	Starting	Starting	Starting
True Coincidence	Single Event	Single Event	Single Event	Coincidence
Pred Coincidence	Single Event	Single Event	Single Event	Single Event
Tau Decay Length	28 m			24 m
Inelasticity	0.76	0.31	0.21	0.44
Track Length*		115 m	740 m	
# hit DOMs	855	649	662	313
Energy neutrino	6.2 PeV	2.4 PeV	3.1 PeV	6.1 PeV

Table 6.1: Detailed information about the falsely classified example events are shown. *= inside the detector

CHAPTER 7

Potential Applications in IceCube

To discuss potential applications of the classifier in IceCube we first need to take a look at the results under a realistic neutrino flux hypothesis. All results presented in Chapter 6 are shown on event-to-event basis. In contrast to this we now want to weight all events according to the IceCube best-fit neutrino flux. To conclude the chapter we present two potential applications of the classifier in IceCube: first we outline the use of the classifier in event selections. Second we discuss a possible way to search for tau neutrinos.

7.1 Weighted Results

We weight all events according to the IceCube best-fit neutrino flux which is composed of three parts: the atmospheric flux, the prompt flux and the astrophysical flux. We assume an atmospheric flux based on the model by Honda in 2006 [16]. Furthermore, the prompt neutrino flux suggested by Enberg in 2008 [9] was used. Finally, the astrophysical flux we exploit is described by the powerlaw

$$\Phi_{\nu+\bar{\nu}}(E_\nu) = c_0 \times 10^{-18}\,\mathrm{GeV}^{-1}\,\mathrm{cm}^{-2}\,\mathrm{sr}^{-1}\,\mathrm{s}^{-1}\left(\frac{E_\nu}{100\,\mathrm{TeV}}\right)^{-\gamma}, \tag{7.1}$$

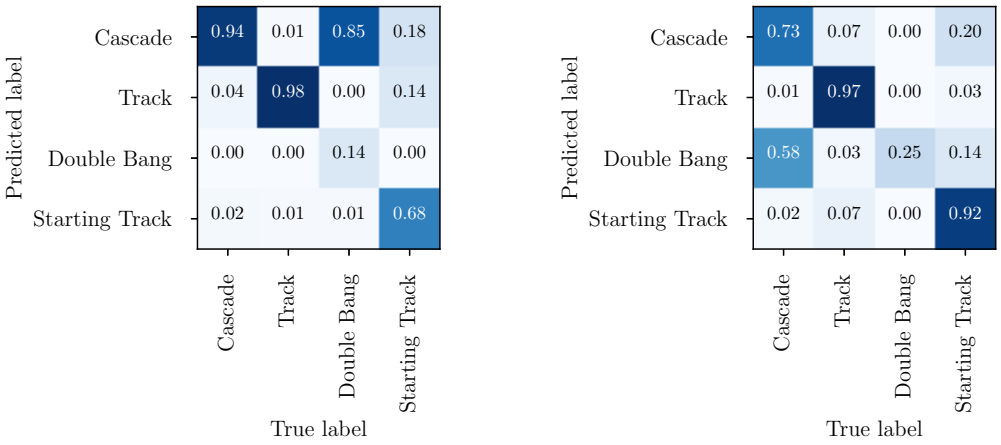
where we chose c_0 and the spectral index γ according to [13],

$$c_0 = 1.01 \qquad \gamma = -2.19. \tag{7.2}$$

As a result we expect for our data processing a total rate of around 9925.59 events per year. These consist of 9527.69 events of atmospheric origin, 217.67 events of prompt origin and 180.23 events of astrophysical origin. These rates split into the different event topologies as follows: in average we expect 962.26 cascades, 7575.05 tracks, 1386.26 starting tracks and 2.02 double bangs per year.

7.1.1 Event Type Classification

This section compares the event type classification results on event-to-event basis, presented in Section 6.3, to the ones resulting by weighting the events. The confusion matrices were shown in Figure 6.5. In Figure 7.1 they are shown again, but now based on the weighted events.



(a) ground truth normalized (b) prediction normalized

Figure 7.1: The weighted confusion matrices of the event type classification are shown. The left confusion matrix is normalized on the ground truth of each class while the right matrix is normalized on the predictions of each class.

The main changes introduced through the weighting are an accuracy reduction for starting tracks of 7%, a reduction of the precision for cascades by 13%, an accuracy reduction of 7% for double bangs and a larger reduction of the precision of double bangs from 57% to 25%. The precision loss of double bangs is explainable by their small expected rate or more precisely by the ratio of the event rates per type. Double bangs are expected much rarer compared to the other topologies, for example we expect about 3745 times more incoming tracks than double bangs. On event-to-event basis though the ratio between incoming tracks and double bangs was about 4.5. Hence by weighting, the fraction of events that get falsely identified as double bangs increases compared to the number of true ones, as the other event topologies are expected to be seen at higher rates. The precision of tracks improves by 4% and the precision for starting tracks by 3%.

In Figure 7.2 the absolute number of expected events of each possible truth and prediction combination is shown. Due to the high expected rate of incoming tracks and their good accuracy and precision we see this bin dominating the confusion matrix: we expect a total of 7422.93 correctly classified tracks each year. Correctly classified cascades are expected at a rate of 906.43 events per year, while only 52.84 true cascades are confused. Around the same rate of 939.17 correctly classified starting tracks are expected each year, while a lot more starting tracks are confused, 447.09. With 0.28 events per year, which equals 1 event every 3.57 years, correctly classified double bangs are expected least frequently. Most of the true double bangs, 1.72 per year, are falsely classified as cascades.

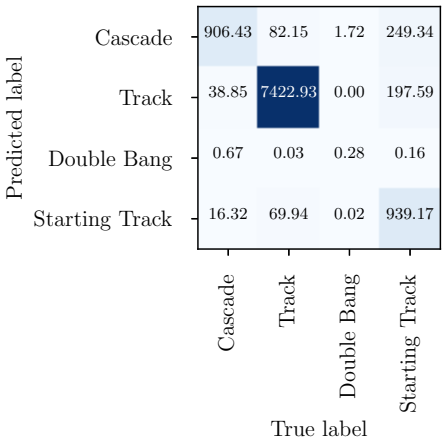
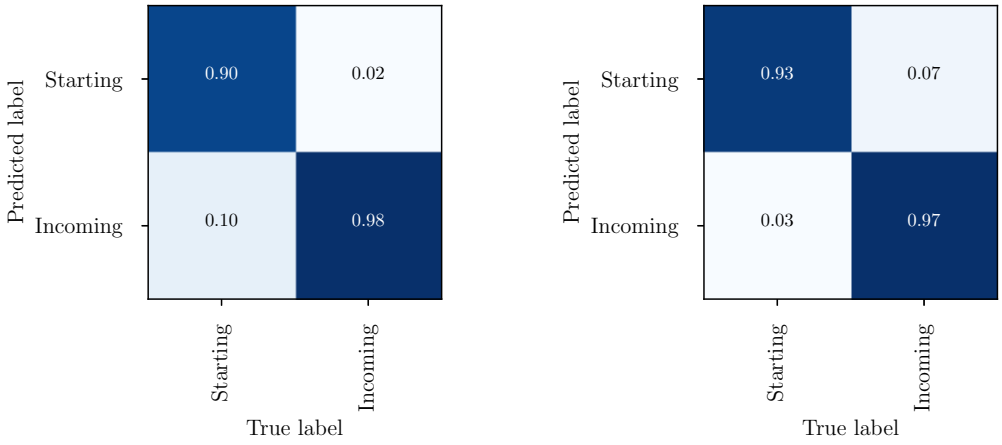


Figure 7.2: The weighted confusion matrix of the event type classification showing the number of absolute expected events per year is depicted.

7.1.2 Starting Event Identification

In this section the performance of the starting event identification for a realistic flux model is discussed. The corresponding confusion matrices are presented in Figure 7.3.



(a) ground truth normalized (b) prediction normalized

Figure 7.3: The weighted confusion matrix of the starting classification once normalized on the ground truth of each class and once normalized on the predictions of each class is depicted.

Compared to the event-to-event case in Figure 6.12, the accuracy and precision of starting events decreased while the precision of incoming events increased: the accuracy of the starting class decreased from 97% to 90% and the corresponding precision dropped by 6%. For the incoming class the accuracy does increase by 1%, while the precision also increases by 3%. This change results again from the

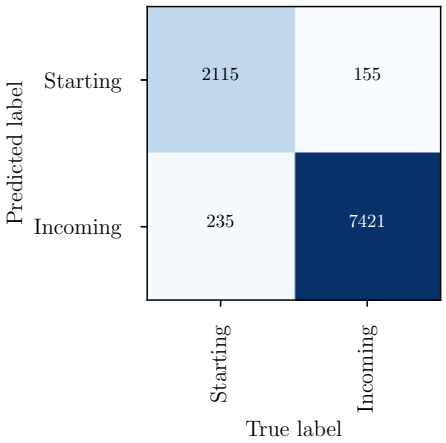
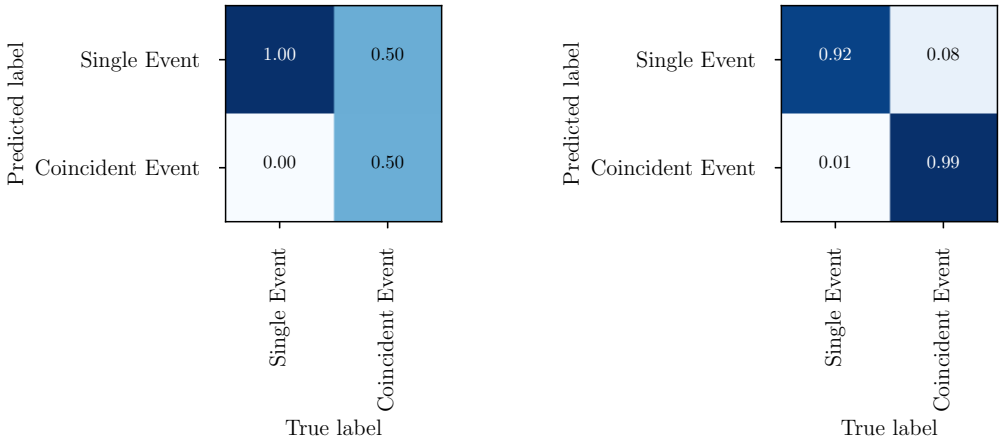


Figure 7.4: The weighted confusion matrix of the starting event identification in the number of absolute expected events per year is presented.

changing ratio of the classes: on event-to-event basis more cascades are in the dataset, due to the weighting this changes to relatively more tracks and starting tracks. The cascades are well predicted as starting events.
The confusion matrix with the absolute values for the starting event identification is shown in Figure 7.4. A rate of 2115 correctly identified starting events per year is expected, while 155 events are misclassified as starting. For incoming events 7421 events are classified correctly each year, while confusing 235 starting events as incoming.

7.1.3 Coincident Event Identification

Here the results of the coincidence identification if the IceCube best-fit neutrino flux is assumed are shown. In Figure 7.5 the corresponding confusion matrices are depicted.



(a) ground truth normalized (b) prediction normalized

Figure 7.5: The weighted confusion matrix of the coincidence classification once normalized on the ground truth of each class and once normalized on the predictions of each class is shown.

The accuracy of detecting coincidences decreases by 4% to 50%, which corresponds to pure guessing. However the precision of the prediction stays high at 99%. As a result 735 coincidences are correctly identified each year, while only 6 events are falsely classified as a coincidence. The confusion matrix with the absolute values is depicted in Figure 7.6.

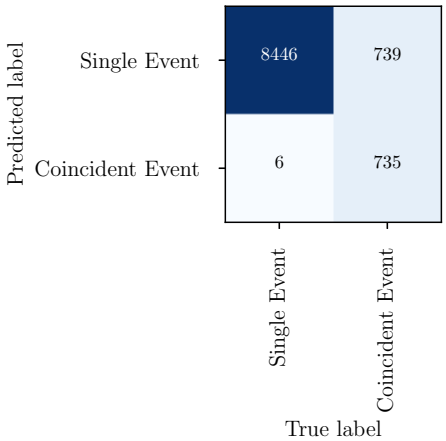
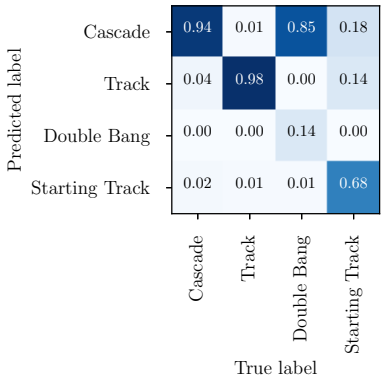


Figure 7.6: The weighted confusion matrix of the coincidence identification in the number of absolute expected events per year is depicted.

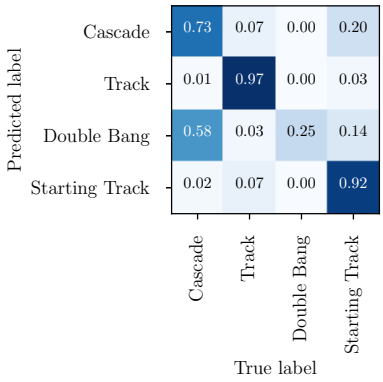
7.2 Usage for Event Selections

A first potential application of the classifier within IceCube is its usage for event selections. Based on the classification results events can be selected. If for example an analysis needs a dataset consisting of track-like events, only events classified as track or starting track are selected. Most of the time it is thereby very important to obtain a dataset that is as pure as possible. Hence it is beneficial to only use events that are predicted more certain. Depending on the later use case a balance between purity and the number of remaining events has to be found. We have shown in Section 6.3.1 that the p-score can be interpreted as a certainty of the classification results and can thus be used as a decision criteria.

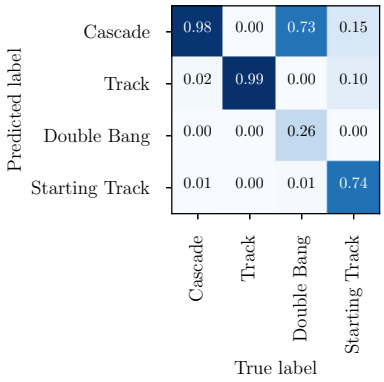
In the following the confusion matrix for the event type classification is shown three times in Figure 7.7: the first line shows the matrix with all events for comparison reasons. The matrices in the second line are based on events having a minimal p-score of 0.8. Lastly, the matrices in the third row are based on events having a p-score greater than 0.90.
The development over an increasing cut-off p-score illustrated in Figure 7.7 shows a continuous increase in accuracy and precision of each task. The largest confusion that remains after an increase of the cut-off p-score to 0.9 is the confusion of true double bangs with cascades. Also a minor confusion of true starting tracks with cascades and tracks remains as well. They are at a level of about 10% each. Further confusion matrices with a minimal p-score of 0.75, 0.85 and 0.95 can be found in the Appendix A.4.



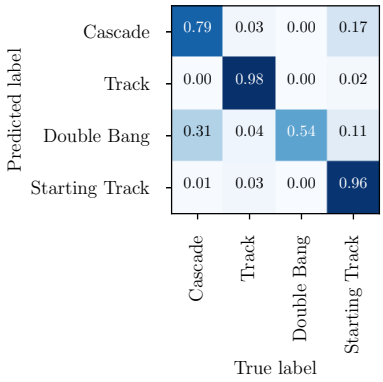
(a) ground truth normalized, $p > 0.0$



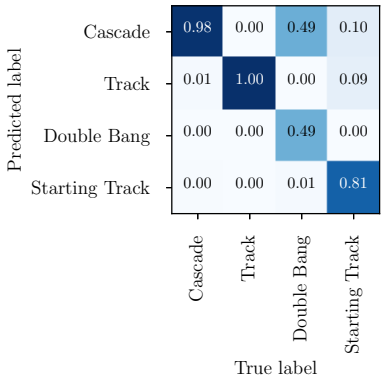
(b) prediction normalized, $p > 0.0$



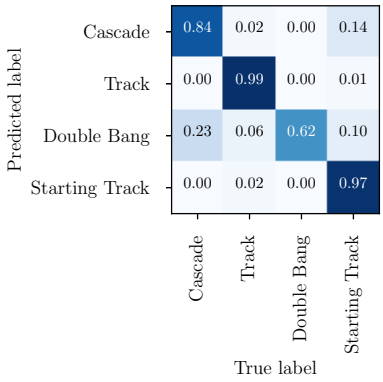
(c) ground truth normalized, $p > 0.80$



(d) prediction normalized, $p > 0.80$



(e) ground truth normalized, $p > 0.90$



(f) prediction normalized, $p > 0.90$

Figure 7.7: The development of the two normalized confusion matrices for the event type classification task if only events with an increasing minimal p-score are used is illustrated. Therefore the matrices are once shown based on all events, once only with events of a p-score greater than 0.80 and finally only based on events with a p-score greater than 0.90. p-score was shortened to p. Events are weighted to the IceCube best-fit neutrino flux.

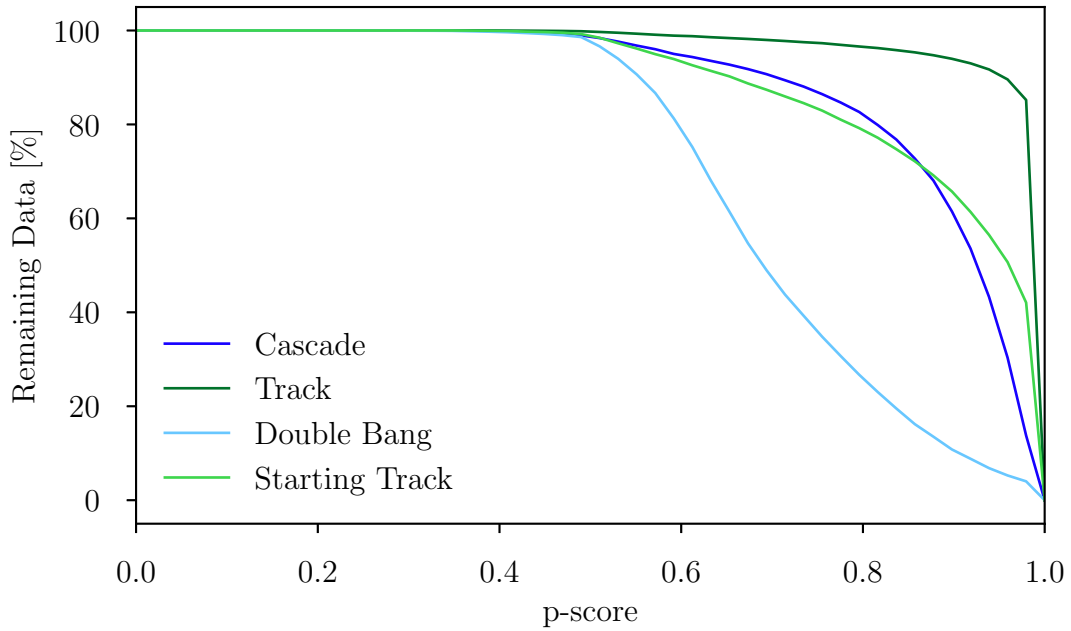


Figure 7.8: The fraction of data that remains as a function of the p-score cut-off is shown. Events are weighted to the IceCube best-fit neutrino flux.

The disadvantage of only using events that got predicted with a p-score higher than a certain threshold is that only a fraction of the data will meet the imposed requirement. Thereby the number of events in the sample is reduced. Figure 7.8 presents the fraction of data that remains as a function of the p-score threshold. For a given p-score the remaining fraction of data for tracks is the largest of all classes. It only starts to drop significantly at a p-score of over 0.95. In general tracks are therefore classified with the highest certainty, nevertheless this alone isn't a measure for how well they get predicted. For the class double bang on the contrary, the remaining fraction of data drops very early, starting at p-scores of 0.50. For a p-score of less than 0.70 already more than half of the events get dropped. The classes of cascades and starting tracks are in between those two extremes. Details can be found in Figure 7.8.

Coming back to the example made in the beginning of this section, a dataset of only track-like events is wanted. We now want to look at this selection process in more detail. In Figure 7.9 the joint precision and remaining data fraction of tracks and starting tracks as a function of $1 -$ p-score is sketched on a log-scale. The p-score increases to the left side. The precision is high over the full spectrum. However, for a track event selection a confusion of some sub-percentages can already be crucial. On the log scale the remaining data fraction decreases nearly linearly starting at a minimal p-score of around 0.5.

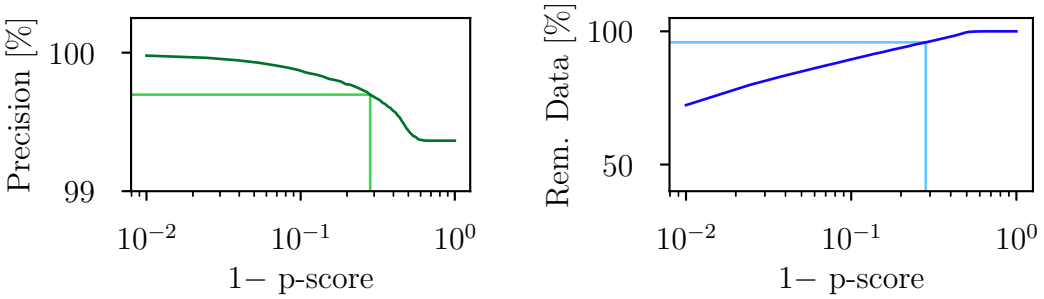


Figure 7.9: For a track-like event selection the joint precision and remaining data percentage of tracks and starting tracks as a function of $1 - \text{p-score}$ on a log-scale is sketched. The remaining data fraction is shown on the right, the precision is shown on the left. The corresponding values for a forced precision of about 99.7% are highlighted.

Already without applying any cut a remarkable precision of 99.36% is obtained. If for example a precision of 99.7% should be obtained a minimal p-score of 0.72 needs to be forced. This cut results in dropping 4.12% of the events with the prediction track or starting track. Thus 95.88% of the data remains. The respective points are highlighted in Figure 7.9.

This principles of tuning the accuracy and precision by forcing a p-score over a certain threshold also applies to the starting events identification and coincidence identification tasks. As a short overview the evolution of their confusion matrices as a function of a increasing minimal p-score are shown in the Appendix A.4.

7.3 Double Bang Detection

The identification of events originating from tau neutrino interactions is an important task in IceCube. As a double bang structure can only be caused by tau neutrinos, the search for taus can be done by the identification of the double bang topology. A reliable identification of the latter one through the classifier could thus assist in the identification of tau neutrinos.

With the current version of our classifier 0.28 double bangs are identified correctly each year, with a corresponding background of 0.86 events per year, see Figure 7.2. The background equals the number of falsely identified events per year. About 1.72 events of the total 2.02 double bangs expected each year are classified as different classes.
The signal to background ratio can be tuned by applying cuts. Hereby a balance with the remaining signal has to be found.

To see which kind of double bangs we find the accuracy as a function of the number of minimal hit DOMs and minimal p-scores is shown in Figure 7.10. In more detail,

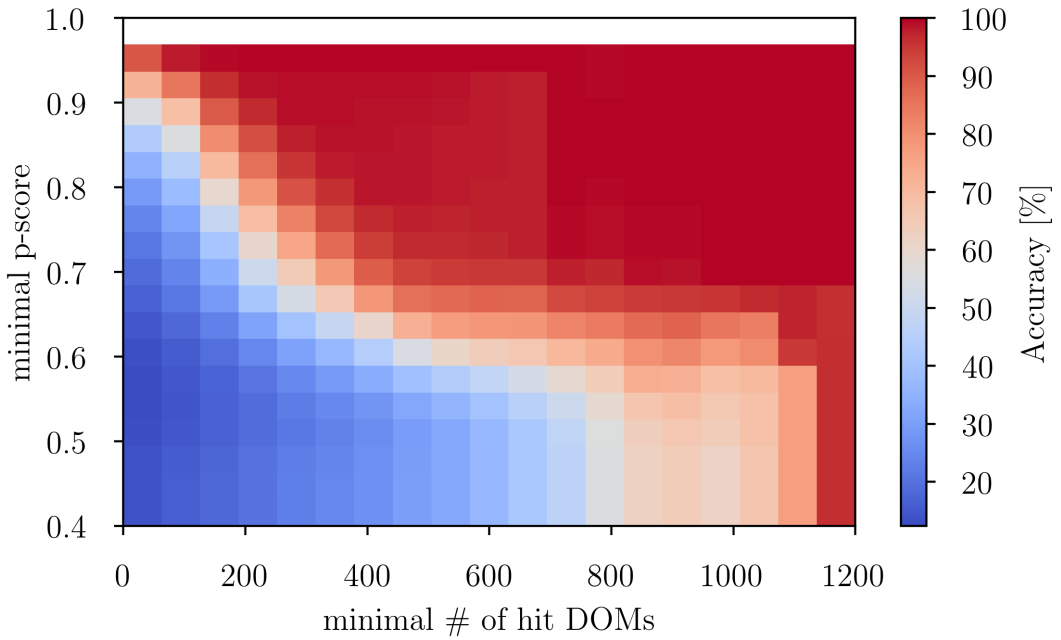


Figure 7.10: The accuracy of true double bangs as a function of the minimal number of hit DOMs and the minimal p-score is plotted. Events are weighted to the IceCube best-fit neutrino flux.

the x-axis shows the number of minimal hit DOMs, this means, if the number of minimal hit DOMs has a certain value k all events with k or more hit DOMs are considered. The same principle applies to the minimal p-score, all events with a p-score equal or higher than the minimal p-score are used. As expected the accuracy rises for higher numbers of hit DOMs as well as for higher p-score thresholds. No events have a p-score higher than 0.966, hence the top bin is white.

As for the accuracy in Figure 7.10 we sketch the precision as a function of the number of minimal hit DOMs and minimal p-scores in Figure 7.11. The expected tendency of better predictions for higher minimal p-score is confirmed again. However the unexpected trend of larger precisions for increasing numbers of hit DOMs seems counterintuitive. The highest precisions are achieved for only few hit DOMs and for a minimal p-score of over 0.90. A part of the explanation could be the underlying event distribution, more events are expected in the area of few hit DOMs than for many. Another part could be that possibly the double bangs get to big to be well identified by the classifier.

Using the precision, two linear decision boundaries can be determined by choosing a respective point in Figure 7.11. Thereby a specific signal to background ratio could be obtained. Nevertheless this still results in two straight cuts and they do not lead to an optimal solution. For a final double bang detection analysis here a further processing step should start, that determines more advanced decision boundaries.

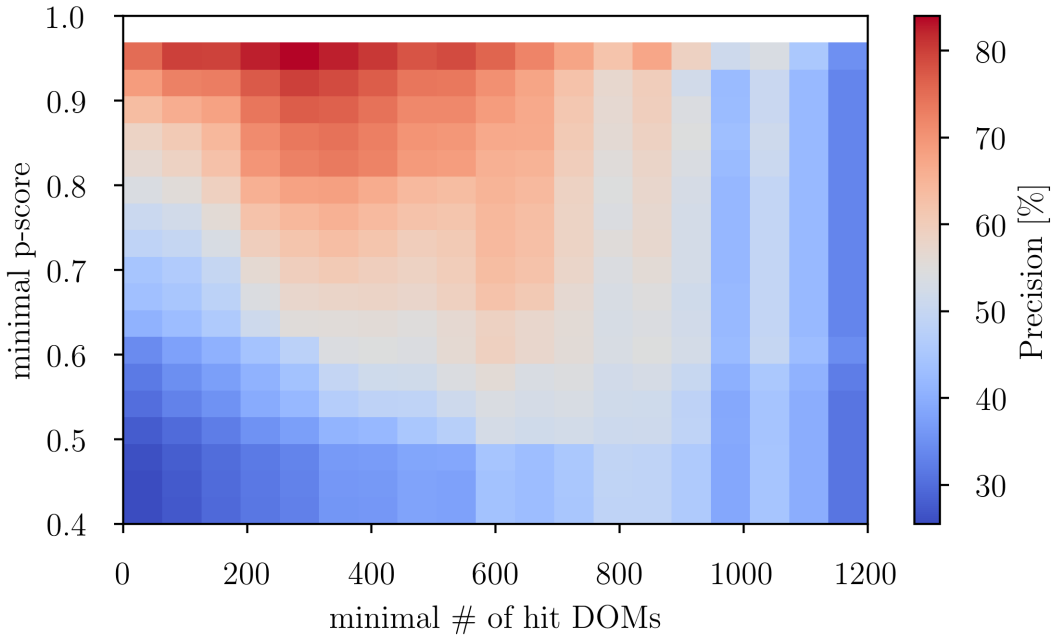


Figure 7.11: The precision of double bang predictions as a function of the minimal number of hit DOMs and the minimal p-score is illustrated. Events are weighted to the IceCube best-fit neutrino flux.

The final interesting question does not concern ratios, but absolute numbers. Therefore Figure 7.12 shows the absolute number of events correctly and falsely classified as double bangs per year as a function of the number of minimal hit DOMs and minimal p-scores.

As expected the loser the cuts are, the higher the rates are. The maximal values, if no cuts are applied, of 0.28 correctly identified events per year and the background of 0.86 events per year can be found in the bottom left corner of the upper subfigure in Figure 7.12 and respectively in the subfigure at the bottom. Further, we see a higher decrease in the number of events per year in the regime for few hit DOM for falsely classified double bangs than for correctly classified ones if we increase the minimal required p-score. Taking the ratio of the events in the two absolute plots gives the precision, shown in Figure 7.11. The two plots can be used to easily see witch signal rate corresponds to with background rate.

Additionally three histograms can be found in the Appendix A.5 showing the number of true double bangs, the number of predicted double bangs and the number of correctly identified double bangs per year binned in number of hit DOMs and p-score.

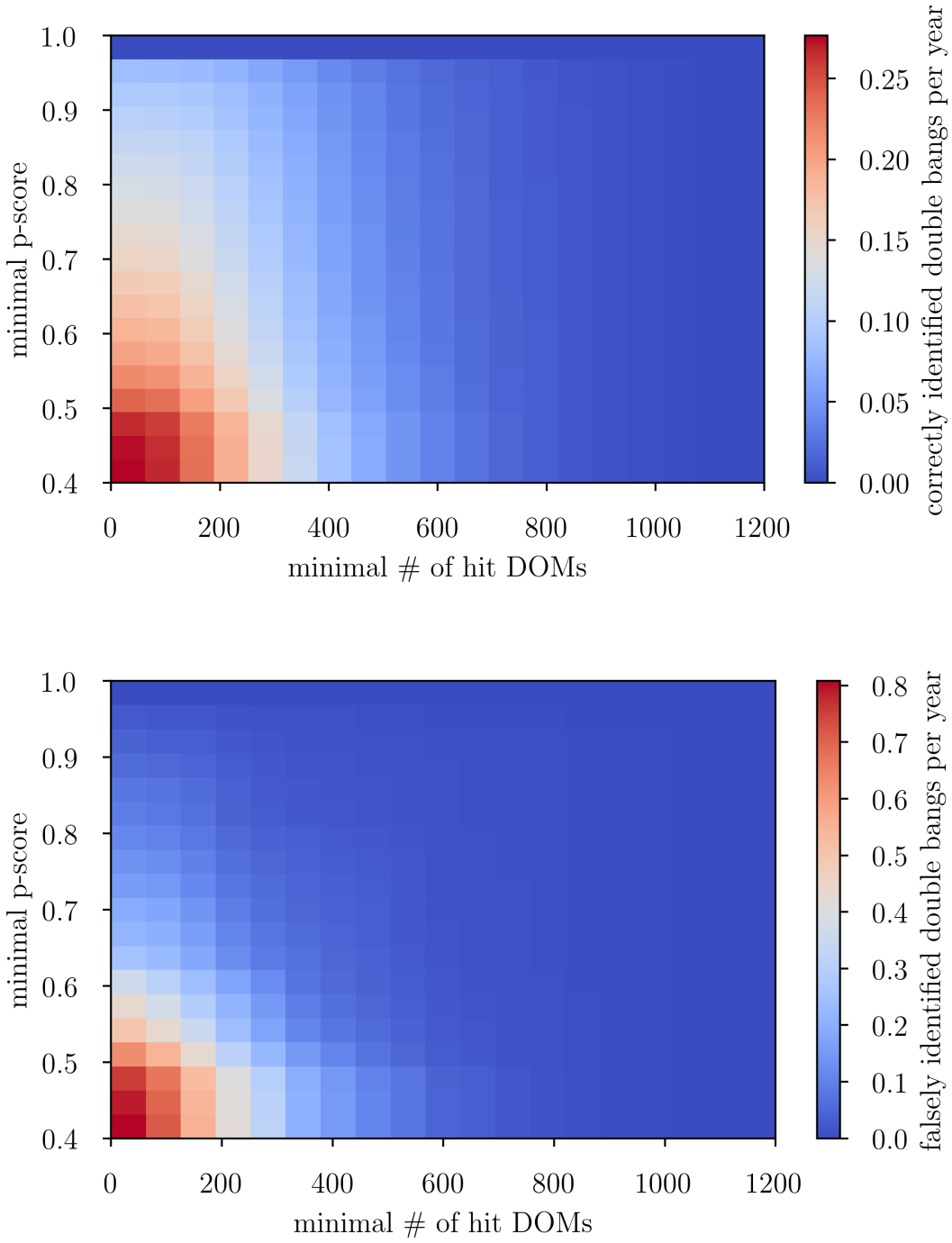


Figure 7.12: Number of double bang predictions per year as a function of the minimal number of hit DOMs and the minimal p-score. On the top the number of correctly identified events per year is shown. In contrast the bottom graph shows the number of double bang predictions per year that are false. The different orders of magnitude for the two graphs should be taken into account. Events are weighted to the IceCube best-fit neutrino flux.

CHAPTER 8

Conclusion and Outlook

8.1 Conclusion

This thesis evaluated the use of deep neural networks for event type classifications in IceCube. The training and evaluation of the neural networks were based on Monte Carlo simulations by the IceCube collaboration.

Firstly, we have shown that special care needs to be taken to generate a consistent dataset. Label definitions should be as clear as possible to enable the network to learn the different event topology characteristics. Further the ratio between the different classes should be carefully chosen to achieve a balanced dataset. We have shown that equally distributed labels prevent biases towards specific classes. Additionally we suggested to adjust the distributions of each individual class further to avoid biases here. An similar distribution in number of hit DOMs among all classes seems promising.

As the available four dimensional input for one single event would have been too large and too complex to train on, we proposed a reduced input size. It is stacked of three dimensional features, containing charge quantiles and the total measured charge per DOM. Furthermore, we used a compressed grid representation by rearranging the strings.

The architecture of the presented neural network is based on the findings of a broad exploration of parameters including the general structure of the architecture, the network's respective size, different loss functions, optimizers and the extend of using different regularization methods. The final classifier presented in this thesis is based on Google's InceptionResNet. It uses the Adam optimizer and dropout as well as batch normalization for the regularization of the training. Furthermore, it utilizes multi-task learning to enable better generalization and to have a more generic application. Each intermediate version of the classifier had been carefully evaluated by checking its results against our physical expectations.

The resulting classifier was able to distinguish the four topologies, cascades, tracks, double bangs and starting tracks, fairly well. 87.9% of its predictions are correct. The confusion of double bangs with cascades remains the largest open issue. Starting events can be distinguished from incoming ones in more than 90% of the cases. The coincidence identification detected around half of all occurring coincidences, while if a coincidence is predicted the classifier is mostly correct. This needs further investigation.

Finally we outlined two potential applications for the classifier in IceCube: event selections and searches for tau neutrinos. However, further investigations have to be done to validate the consistency of the neural network. A fine-tuning of the classifier to the respective task is expected to further improve its performance.

In general we see the achieved results of event classification through our proposed architecture as a proof of principle. We have shown that it is possible to classify event topologies in IceCube on a competitive level through the usage of deep neural networks.

The framework for developing deep learning applications in IceCube partly created within this thesis is publicly available at the following git repository: https://github.com/MaxiKro/DeepIceLearning.

8.2 Outlook

Many open questions and starting points for improvement remain: on the one side the classifier itself can further be optimized while a more profound understanding of the internals of the classifier could be gained. On the other side its usage should be evaluated in more detail for the various applications in IceCube.

One of the biggest constraints of the neural network was our decision to use a special architecture based on three dimensional convolutional neural networks. There are other architectures which may have beneficial aspects for our classification task. For sure it is worth investigating them further. A first potential step could be the use of four dimensional convolutions, as the raw IceCube data is four dimensional as well. This architecture would match the data more natural as well as make the calculation of specific input features obsolete. An alternative approach would be the use of recurrent neural networks. These are able to deal more naturally with the time component of the waveform. Graph neural networks on the other hand could offer the possibility to deal with the irregular geometry of the detector.
Lets assume we stick with the general architecture of our neural network. Additional potential lays in the further optimization of the hyperparameters through a structured hyperparameter scan.
Based on the discussion on the significant influence of the dataset composition in section 5.6, we argue that the biggest potential for improvement for the current architecture is hidden in the dataset. We therefore suggest to train the classifier again on a further optimized dataset. An equal distribution for the number of hit DOMs among the classes seems promising here.
DeepCore increases the resolution of the detector by extending the energy range in which IceCube can measure effectively. However the information measured by DeepCore's DOMs has not yet been included in our classifier. The challenge is an effective implementation in the current input shape of the neural network. We expect an improved performance through its inclusion.
The classifier has so far only been trained on a specific set of simulations that all

have the same ice properties. We have not yet tested the network on systematic datasets, which should be done to evaluate how robust the classifier is and in which way its performance is affected.
A big step in any machine learning application that was trained on simulated data is its application on experimental data. As often slight disagreements between simulated and experimental exist, the applications can show different performances. This test is an important step, which still needs to be done. Especially it has to be checked, weather the neural network has trained on specific features inherent to the Monte Carlo simulation.
Some additional ideas that we want to mention but don't want to outline further are the inclusion of dropped DOMs in IceCube, the use of hexagonal kernels, an event dependent loss function and the visualization of the neural network feature maps. Additionally more tasks, like a direction reconstruction, can be added.

Some of the possible applications in IceCube of such an classifier were already shortly outlined in Chapter 7. We further see the possibility to increase the active volume for starting events to the vicinity of the instrumented volume. This would allow for a detailed study on the performance of the starting events identification with different volume sizes. If this works fine, this would allow to increase the starting event datasets.
Neural networks are extremely fast applied, in an order of milliseconds, once they are trained. Additionally they don't need many computational resources. Therefore the classifier could be used at the South Pole by allowing fast event topology predictions online. The classifier could furthermore be used to perform a flavor ratio analysis based on the predictions of the neural network.
Independent of which application will be targeted, the classifier should further be fine-tuned to this specific task. We thereby expect a further increase in performance such that the classifier becomes competitive as an alternative approach to the present methods applied in IceCube.

Acknowledgements

This thesis would not have been possible on this level without the support of many people. That's why I want to sincerely thank all of them.
First of all, I want to thank my supervising professor, Elisa Resconi, who made it possible for me to write this thesis on such an interesting topic in the first place. She gave me the opportunity to be a part of her amazing working group at the Technical University Munich and IceCube over the last year. Additionally I want to thank her for sending me to several conferences where I had the chance to meet different scientists from all over the world.
I would like to thank Susanne Mertens for agreeing to be the second reviewer of my thesis.
A big thanks to my direct supervisor, Theo Glauch, which supported me during my whole thesis. He somehow survived all my questions and always had great advise and stunning ideas, that changed my work to the better.
A special thanks goes to all my colleges with whom I shared many discussions about many different topics while having great coffee and surprisingly often cake. It was a pleasure to share the last year with you all.
In particular I want to thank Elisa, Theo and Matthias for proofreading this thesis and giving valuable feedback.
Finally I want to thank my parents, without whom my whole studies would not have been possible and I would have never even reached the point to write such an thesis. Thanks for all the support and love.
A special thanks goes to Friedrich, without him my physics studies would have already ended in its first year. Thanks for the last five years of friendship.
A last and special thanks to my girlfriend Antonia, who supported me heavily over the last year, encouraged me over and over again and who was always able to cheer me up.

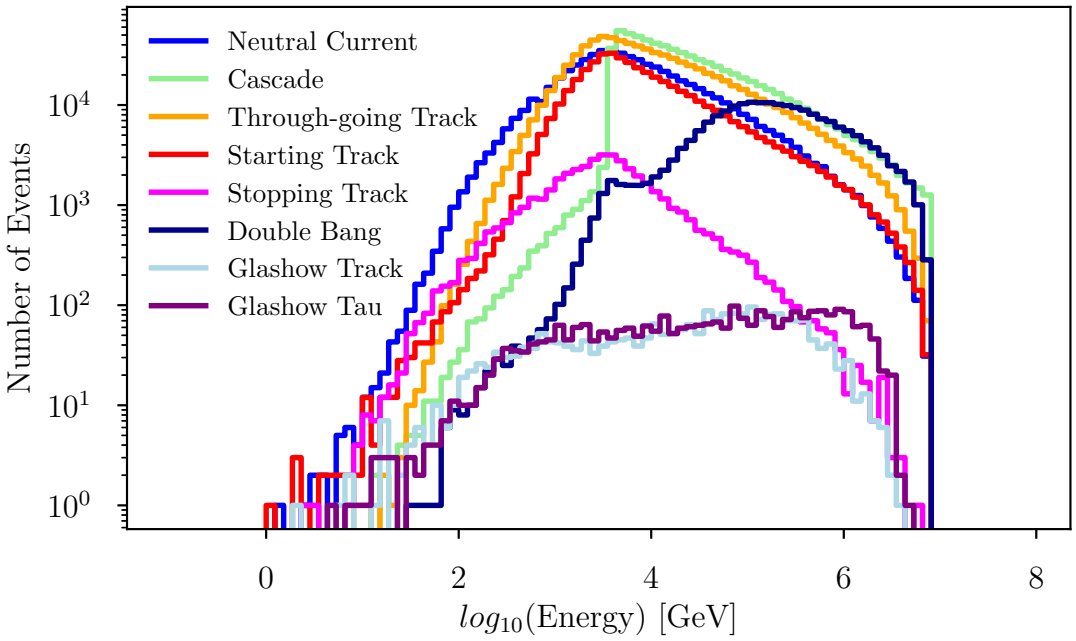
Appendix A

A.1 Monte Carlo Simulation

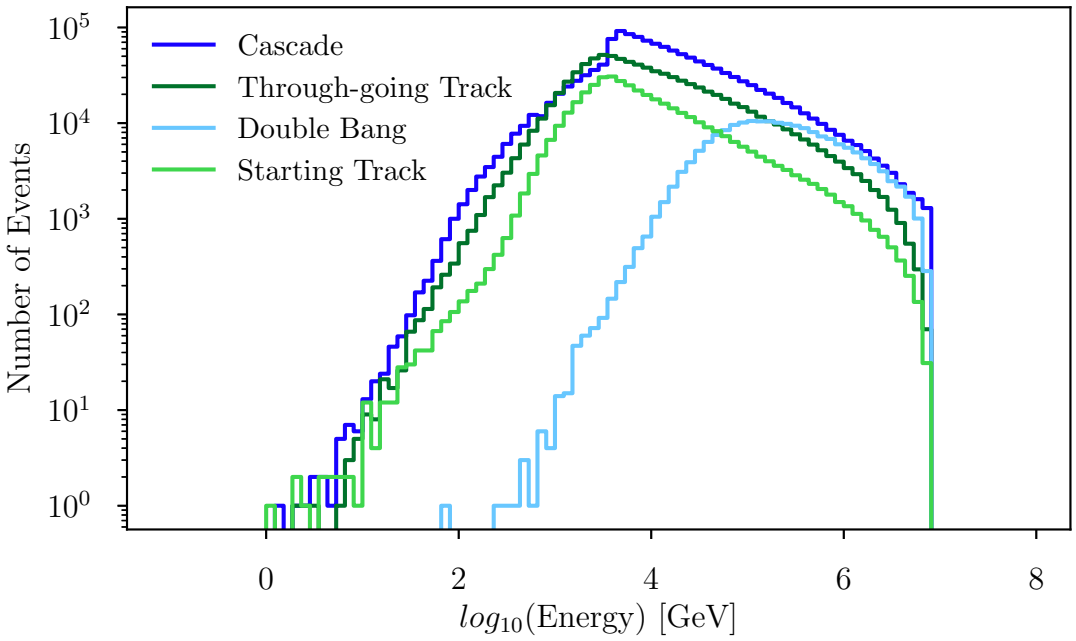
The directories of the used simulations on the cluster of the IceCube collaboration in Madison are listed below:

- /data/ana/Cscd/StartingEvents/NuGen_new/NuE/
  medium_energy/IC86_flasher_p1=0.3_p2=0.0/l2
- /data/ana/Cscd/StartingEvents/NuGen_new/NuMu/
  medium_energy/IC86_flasher_p1=0.3_p2=0.0/l2
- /data/ana/Cscd/StartingEvents/NuGen_new/NuTau/
  medium_energy/IC86_flasher_p1=0.3_p2=0.0/l2

A.2 Additional Distributions of Physics Parameters

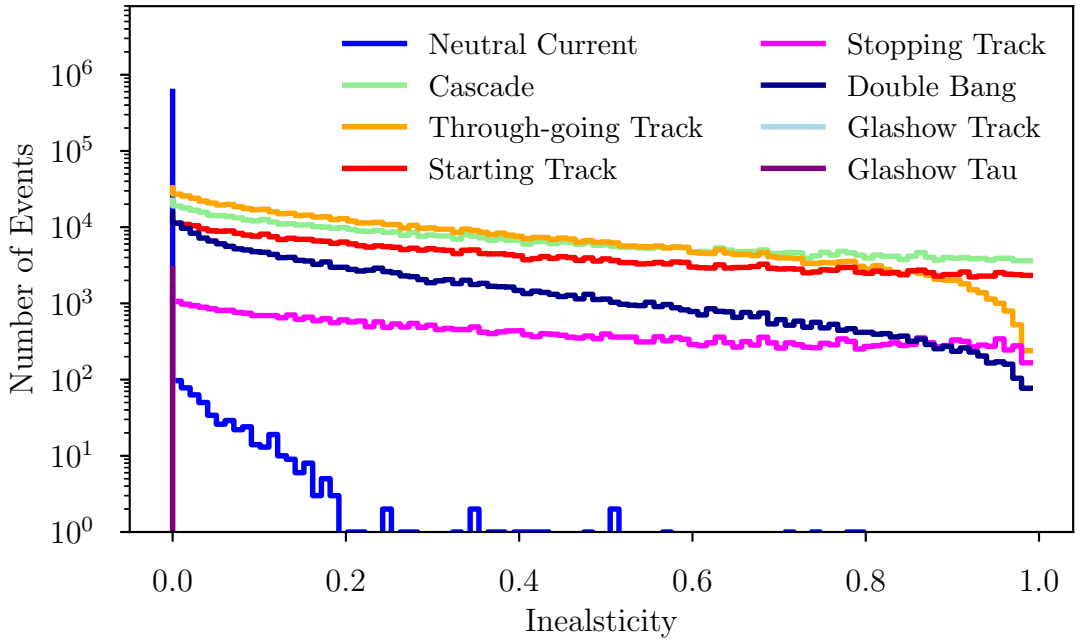


(a) All Event Types

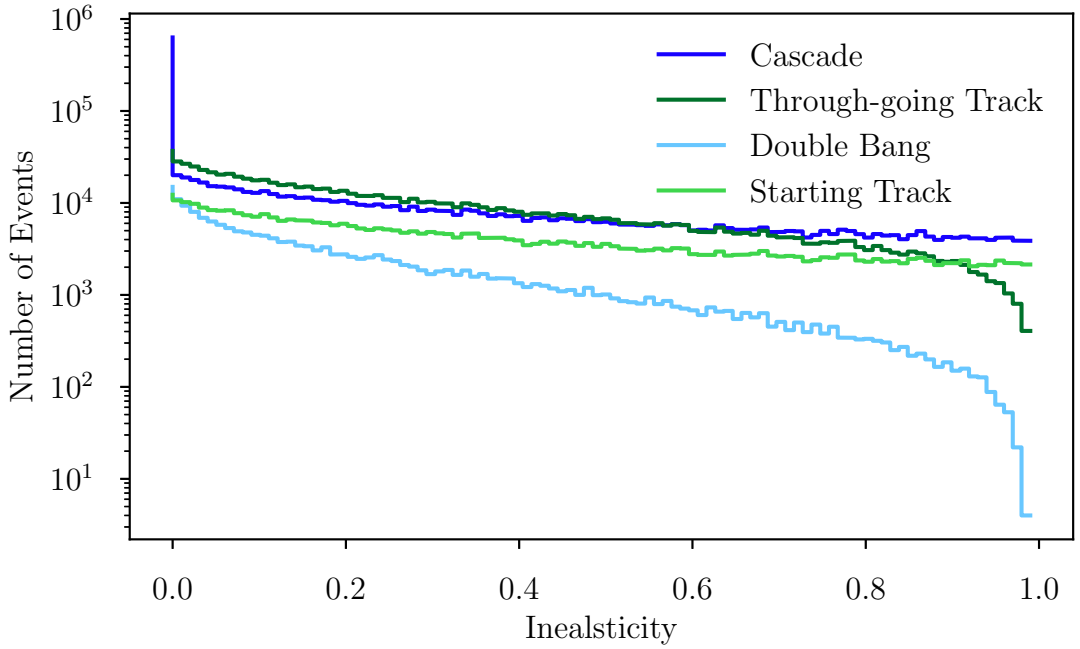


(b) Event Types after relabeling

Figure A.1: Distributions of the whole dataset against the distributed energy once split for all event types and once for the event types which should be classified are shown.

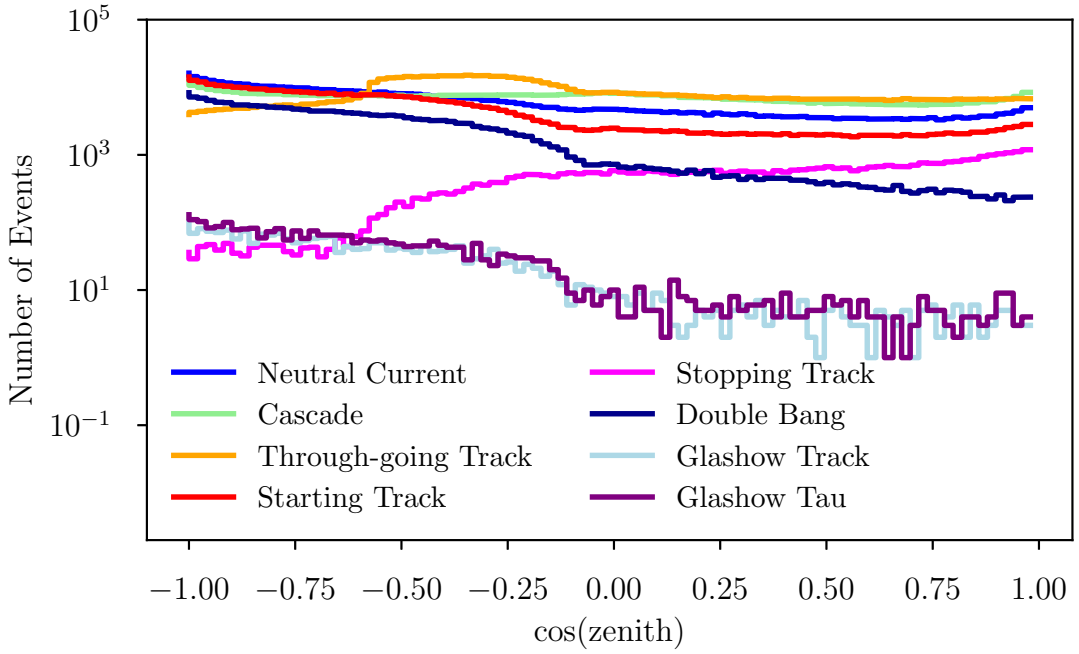


(a) All Event Types

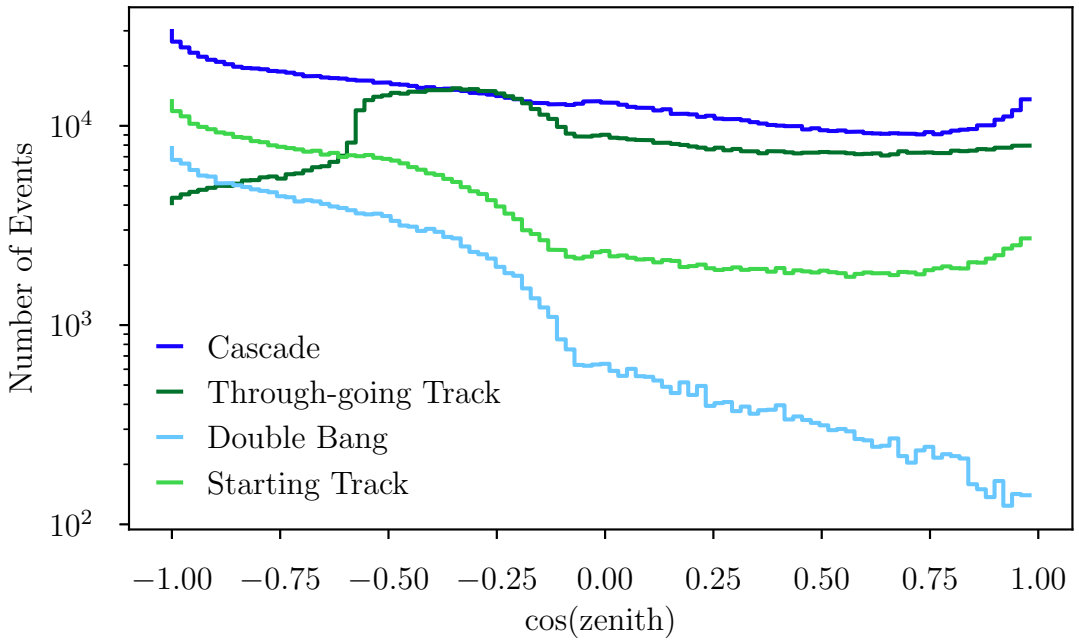


(b) Event Types after relabeling

Figure A.2: Distributions of the whole dataset against the inelasticity once split for all event types and once for the event types which should be classified are depicted.

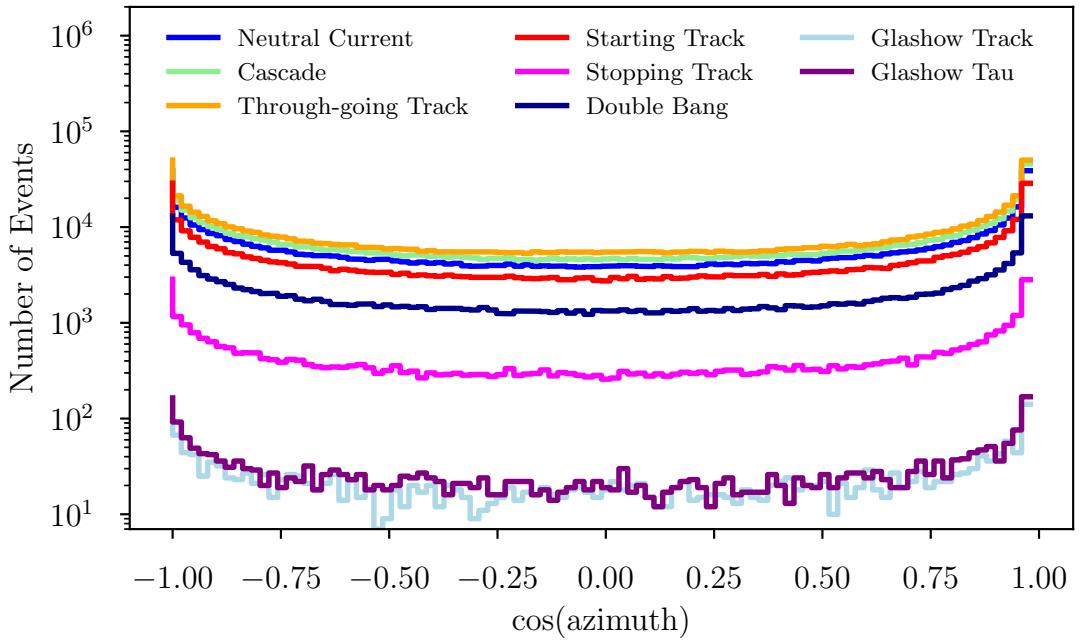


(a) All Event Types

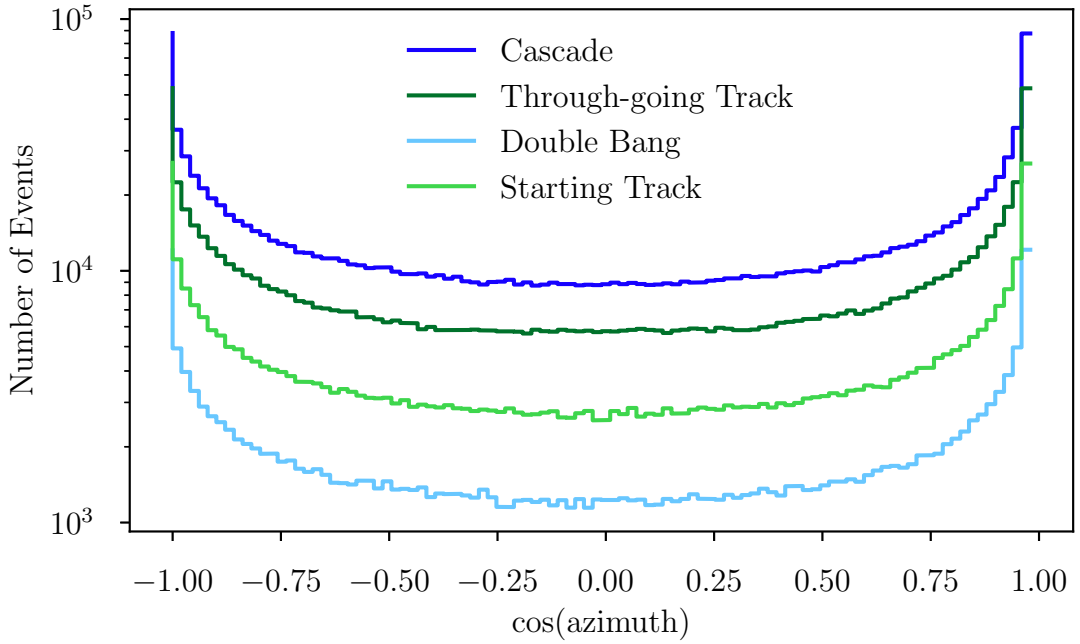


(b) Event Types after relabeling

Figure A.3: Distributions of the whole dataset against the cousin of the zenith once split for all event types and once for the event types which should be classified are presented.

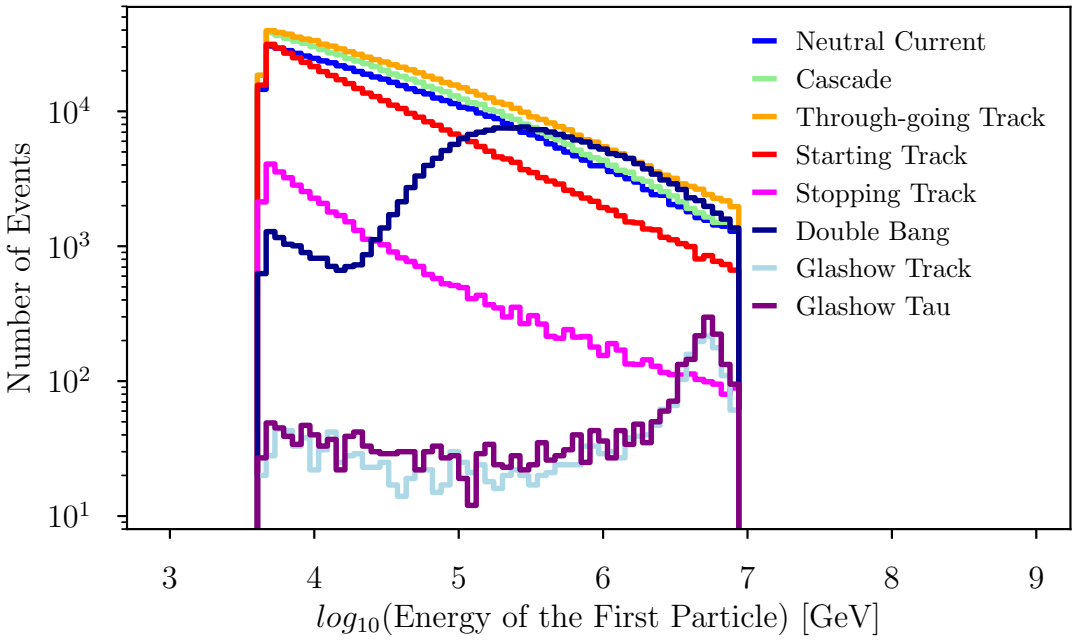


(a) All Event Types

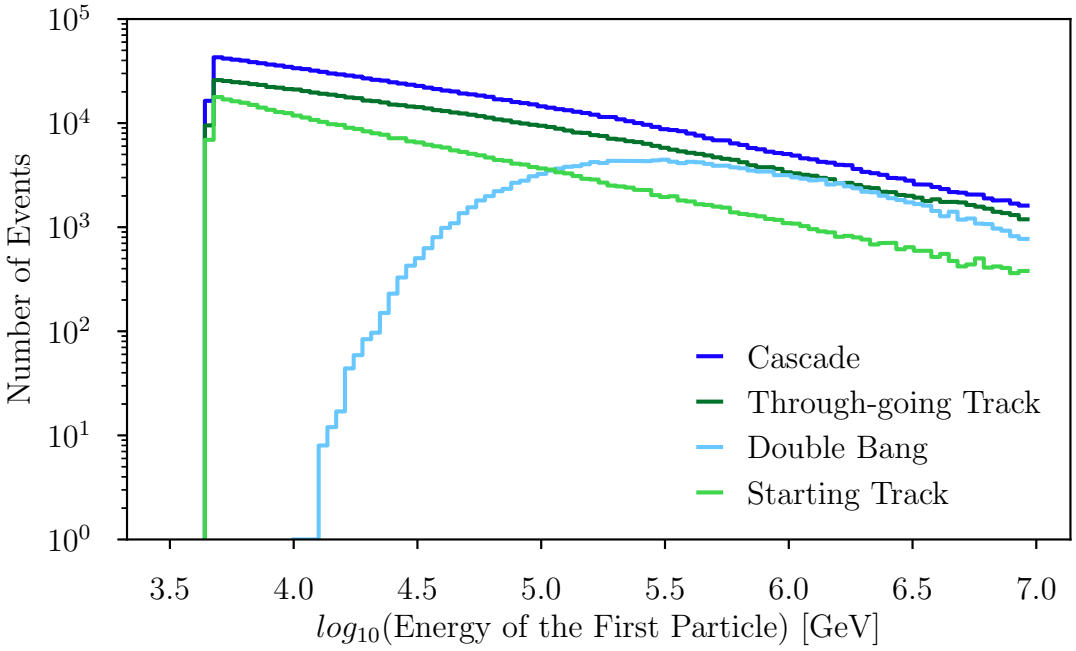


(b) Event Types after relabeling

Figure A.4: Distributions of the whole dataset against the cousin of the azimuth once split for all event types and once for the event types which should be classified are plotted.



(a) All Event Types



(b) Event Types after relabeling

Figure A.5: Distributions of the whole dataset against the energy of the first particle once split for all event types and once for the event types which should be classified are shown.

A.3 Details to the Classifiers Setup

A.3.1 Stem

Position	Layer Type	Number of Filters	Filter Size	Stride
1	3D Conv. with bn*	32	(2,2,3)	1
2	3D Conv. with bn*	32	(2,2,3)	1
3	3D Conv. with bn*	64	(2,2,3)	1
4	MaxPooling		(1,1,2)	1
5	3D Conv. with bn*	80	(2,2,3)	1
6	3D Conv. with bn*	96	(2,2,3)	1
7	MaxPooling		(2,2,3)	(1,1,2)

Table A.1: Details of the different layers of the stem are presented.
*= three dimensional convolution with a batch normalization layer

For all convolutions we used a same padding and a ReLu activation function.

A.3.2 Training Parameter

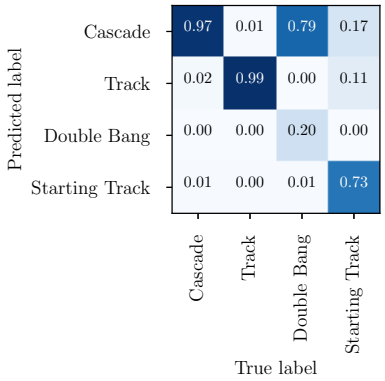
The hyperparameters we used in the training process were chosen as follow:

Parameter	Choice
Patience	20
Verbose	1
Delta	0
Max Queue Size	3
Learning Rate	0.001
Optimizer	Adam
Single Gpu Batch Size	16

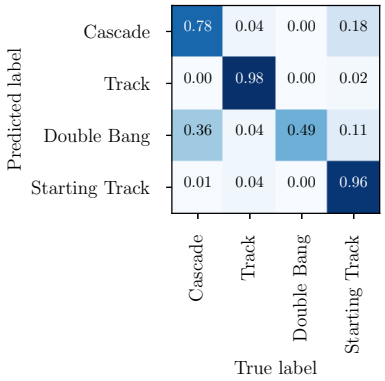
Table A.2: Choices for the parameters used in the training process are listed.

A.4 Confusion Matrix p-cut

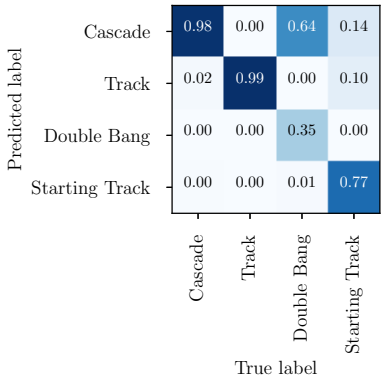
A.4.1 Event Type Classification



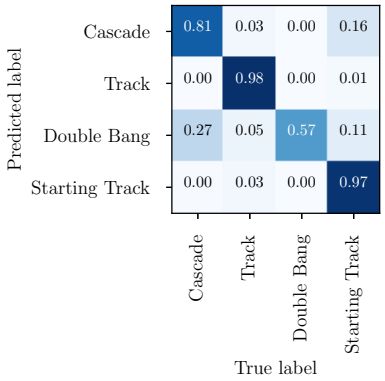
(a) ground truth normalized, $p > 0.75$



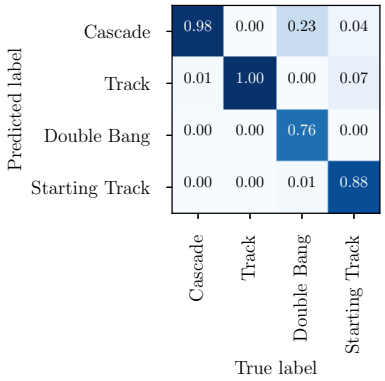
(b) prediction normalized, $p > 0.75$



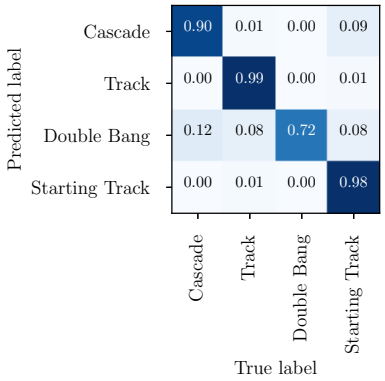
(c) ground truth normalized, $p > 0.85$



(d) prediction normalized, $p > 0.85$



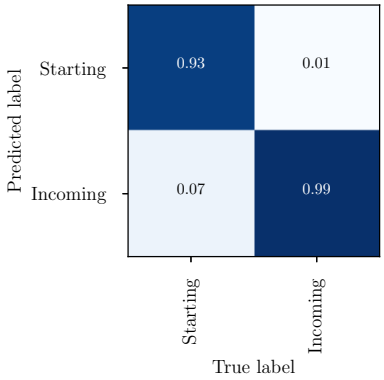
(e) ground truth normalized, $p > 0.95$



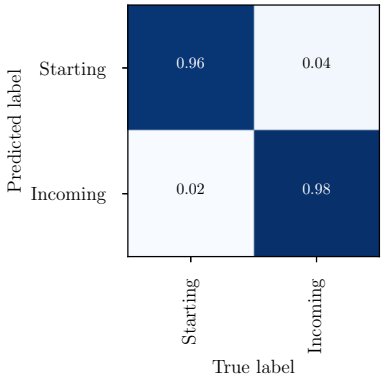
(f) prediction normalized, $p > 0.95$

Figure A.6: The development of the two normalized confusion matrices on an event-to-event basis for the event type classification task if only events with an increasing minimal p-score are used is illustrated. Therefore the matrices with an minimal p-score of 0.75, 0.85 and 0.95 are shown. The values are based on the weighted events.

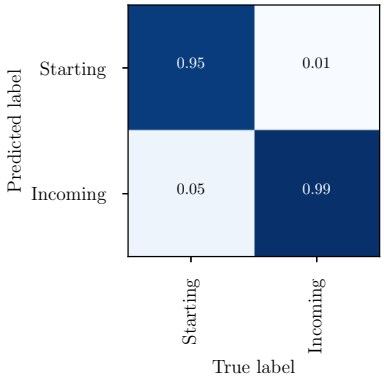
A.4.2 Starting Events Identification



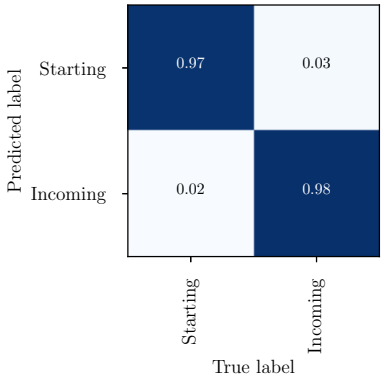
(a) ground truth normalized, $p > 0.75$



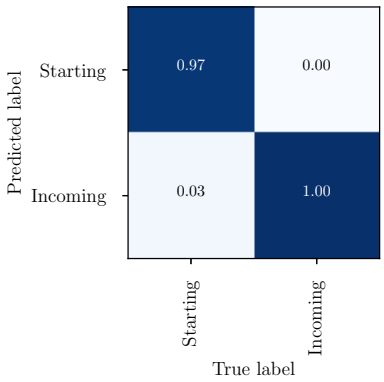
(b) prediction normalized, $p > 0.75$



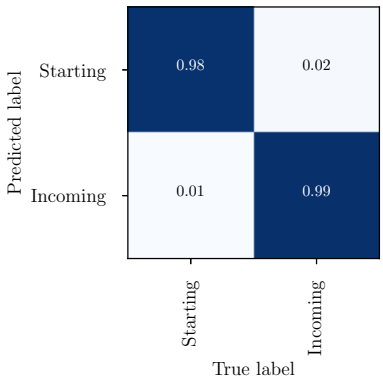
(c) ground truth normalized, $p > 0.85$



(d) prediction normalized, $p > 0.85$



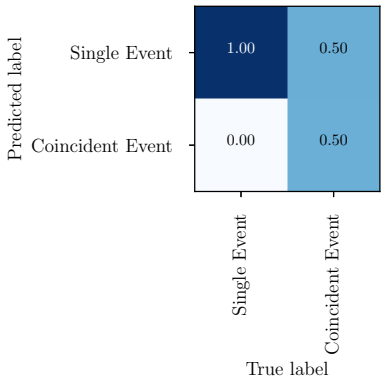
(e) ground truth normalized, $p > 0.95$



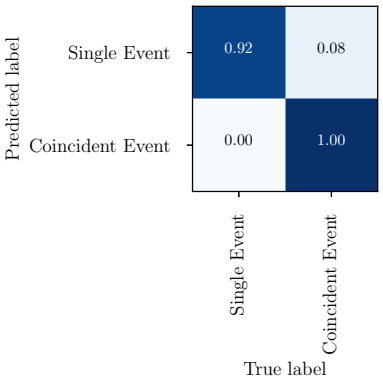
(f) prediction normalized, $p > 0.95$

Figure A.7: The development of the two weighted normalized confusion matrices for the starting events identification task if only events with an increasing minimal p-score are used is illustrated. Therefore the matrices with an minimal p-score of 0.75, 0.85 and 0.95 are shown. The values are based on the weighted events.

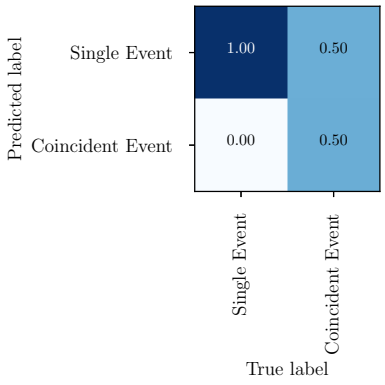
A.4.3 Coincidence Identification



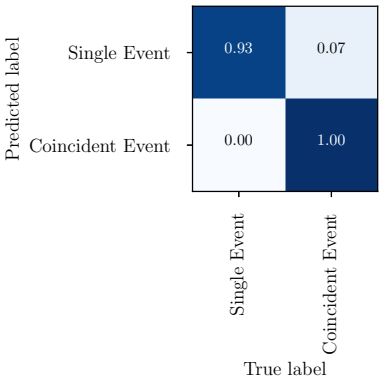
(a) ground truth normalized, $p > 0.75$



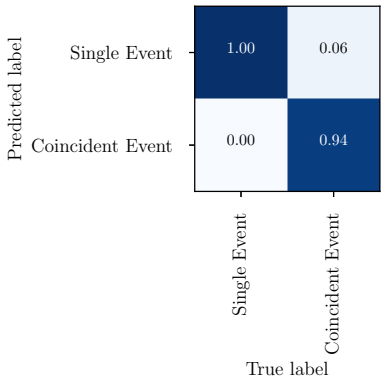
(b) prediction normalized, $p > 0.75$



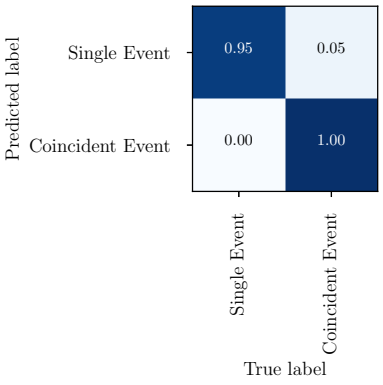
(c) ground truth normalized, $p > 0.85$



(d) prediction normalized, $p > 0.85$



(e) ground truth normalized, $p > 0.95$



(f) prediction normalized, $p > 0.95$

Figure A.8: The development of the two normalized confusion matrices based on weighted events for the coincidence identification task if only events with an increasing minimal p-score are used is illustrated. Therefore the matrices with an minimal p-score of 0.75, 0.85 and 0.95 are shown. The values are based on the weighted events.

A.5 Tau Analysis

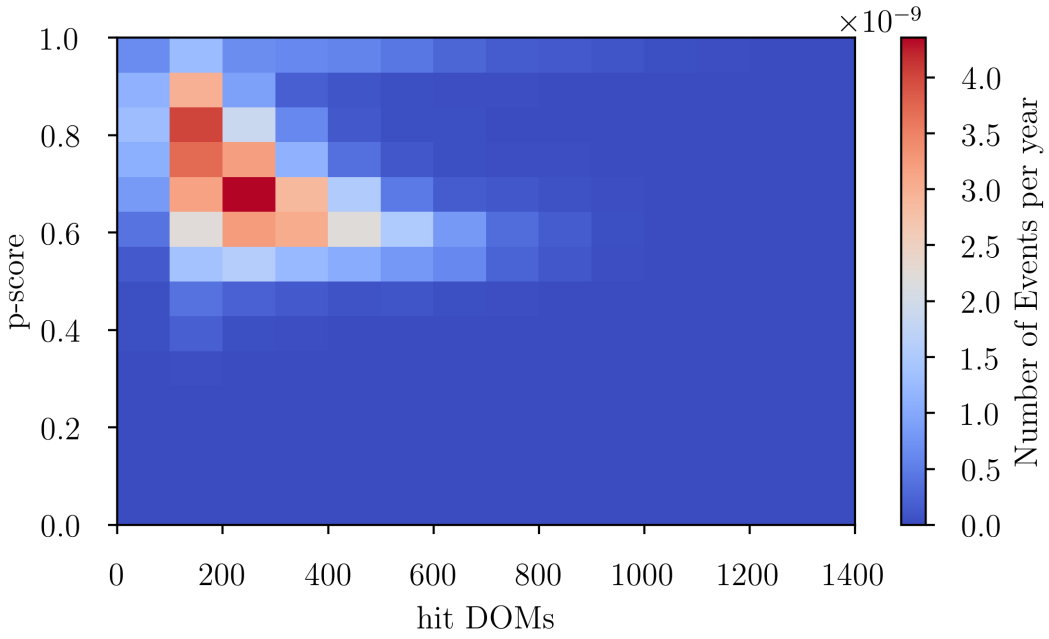


Figure A.9: All true double bangs depended on their number of hit DOMs and there p-score are shown in form of a histogram. Events are weighted to the IceCube best-fit neutrino flux.

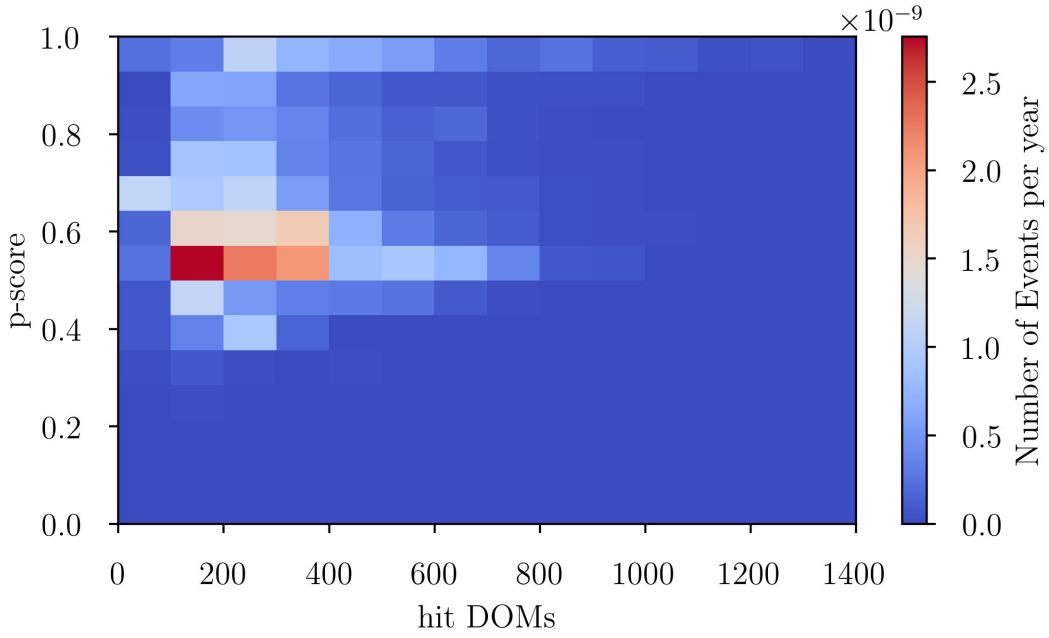


Figure A.10: All predictions of double bangs depended on their number of hit DOMs and there p-score are shown in form of a histogram. Events are weighted to the IceCube best-fit neutrino flux.

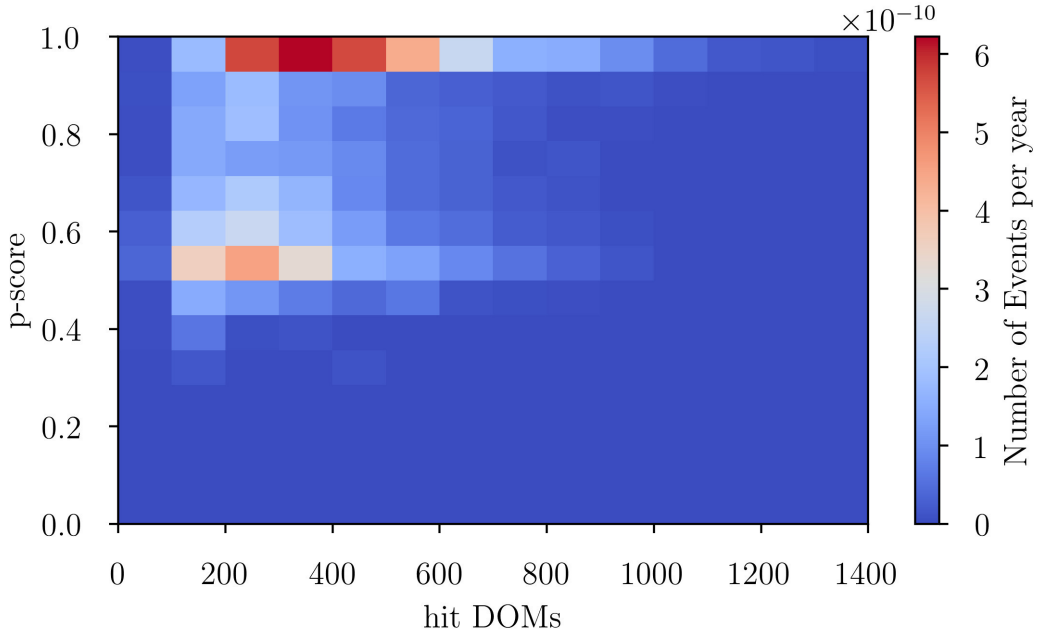


Figure A.11: All correct predictions of double bangs depended on their number of hit DOMs and there p-score are shown in form of a histogram. Events are weighted to the IceCube best-fit neutrino flux.

Bibliography

[1] M. G. Aartsen et al. Evidence for High-Energy Extraterrestrial Neutrinos at the IceCube Detector. *Science*, 342:1242856, 2013.

[2] M. G. Aartsen et al. Measurement of South Pole ice transparency with the IceCube LED calibration system. *Nucl. Instrum. Meth.*, A711:73–89, 2013.

[3] M. G. Aartsen et al. Search for Astrophysical Tau Neutrinos in Three Years of IceCube Data. *Phys. Rev.*, D93(2):022001, 2016.

[4] M. G. Aartsen et al. The IceCube Neutrino Observatory: Instrumentation and Online Systems. *JINST*, 12(03):P03012, 2017.

[5] M. G. Aartsen et al. Neutrino emission from the direction of the blazar TXS 0506+056 prior to the IceCube-170922A alert. *Science*, 361(6398):147–151, 2018.

[6] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018.

[7] J. Beringer et al. Review of particle physics. *Physical Review D - Particles, Fields, Gravitation and Cosmology*, 86(1), 7 2012.

[8] Richard Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

[9] Rikard Enberg, Mary Hall Reno, and Ina Sarcevic. Prompt neutrino fluxes from atmospheric charm. *Phys. Rev. D*, 78:043005, Aug 2008.

[10] J. A. Formaggio and G. P. Zeller. From eV to EeV: Neutrino Cross Sections Across Energy Scales. *Rev. Mod. Phys.*, 84:1307–1341, 2012.

[11] Raj Gandhi, Chris Quigg, Mary Hall Reno, and Ina Sarcevic. Ultrahigh-energy neutrino interactions. *Astropart. Phys.*, 5:81–110, 1996.

[12] Sheldon L. Glashow. Resonant scattering of antineutrinos. *Phys. Rev.*, 118:316–317, Apr 1960.

[13] Christian Haack and Christopher Wiebusch. A measurement of the diffuse astrophysical muon neutrino flux using eight years of IceCube data. *PoS*, ICRC2017:1005, 2018.

[14] Francis Halzen and Spencer R. Klein. IceCube: An Instrument for Neutrino Astronomy. *Rev. Sci. Instrum.*, 81:081101, 2010.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[16] Morihiro Honda, T. Kajita, K. Kasahara, S. Midorikawa, and T. Sanuki. Calculation of atmospheric neutrino flux using the interaction model calibrated with atmospheric muon data. *Phys. Rev.*, D75:043006, 2007.

[17] Mirco Huennefeld. Deep Learning in Physics exemplified by the Reconstruction of Muon-Neutrino Events in IceCube. *PoS*, ICRC2017:1057, 2018.

[18] P. A. Čerenkov. Visible radiation produced by electrons moving in a medium with velocities exceeding that of light. *Phys. Rev.*, 52:378–379, Aug 1937.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[20] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *CoRR*, abs/1702.05659, 2017.

[21] Johannes Kager. Investigation on Applying Deep Learning Methods for Event Reconstruction in IceCube. 2017.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[23] Kai Krings. Search for Galactic and Extra-Galactic Neutrino Emission with IceCube. 2018.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[26] Yann LeCun, Y Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[27] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[28] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *PSYCHOLOGICAL REVIEW*, pages 65—-386, 1958.

[29] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[30] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis, editors, *Artificial Neural Networks – ICANN 2010*, pages 92–101, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[31] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[32] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[34] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

List of Figures

List of Tables

List of Abbreviations

ATWD analog transient waveform digitizer

CC charged current (interaction)

DOM digital optical module

fADC fast analog digital converter

MTL multi task learning

NC neutral current (interaction)

PMT photon multiplier tube

ReLu rectified linear unit

RNN recurrent neural network

Declaration

I, Maximilian Kronmueller, herewith declare that I have composed the present thesis myself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The thesis in the same or similar form has not been submitted to any examination body and has not been published. This thesis was not yet, even in part, used in another examination or as a course performance. Furthermore I declare that the submitted written (bound) copies of the present thesis and the version submitted on a data carrier are consistent with each other in contents.

Munich, 13.12.2018 _____