technische universität
dortmund

Arbeit zur Erlangung des akademischen Grades
Master of Science

Online Reconstruction of Muon-Neutrino Events in IceCube using Deep Learning Techniques

vorgelegt von

Mirco Hünnefeld

geboren in Höxter

2017

Lehrstuhl für Experimentelle Physik Vb
Fakultät Physik
Technische Universität Dortmund

Erstgutachter:	Prof. Dr. Dr. Wolfgang Rhode
Zweitgutachter:	Prof. Dr. Kevin Kröninger
Abgabedatum:	3. November 2017

Abstract

The IceCube Neutrino Observatory is a Cherenkov detector deep in the Antarctic ice. Due to limited computational resources and the high data rate, only simplified reconstructions restricted to a small subset of data can be run on-site at the South Pole. However, in order to perform online analyses and to issue real-time alerts, fast and powerful reconstructions are desired.

Recent advances, especially in image recognition, have shown the capabilities of deep learning. Deep neural networks can be extremely powerful and their usage is computationally inexpensive once the networks are trained. These characteristics make a deep learning-based reconstruction an excellent candidate for the application on-site at the South Pole. In contrast to image recognition tasks, the reconstruction in IceCube poses additional challenges as the data is four-dimensional, highly variable in length, and distributed on an imperfect triangular grid.

In this thesis, a deep learning-based reconstruction method is presented which can significantly increase the reconstruction accuracy while reducing the runtime in comparison to standard reconstruction methods in IceCube.

Contents

1 Introduction

In 2013 the first evidence for a diffuse astrophysical neutrino flux was reported by IceCube [3]. With more years of data taking, the detection of an astrophysical neutrino flux was confirmed [4, 37]. These analyses use a veto region inside the IceCube detector to select high-energy starting events (HESE). A complimentary detection channel selecting charged-current muon-neutrino events further confirmed the measured astrophysical neutrino flux [2, 7, 25]. Even though IceCube has been able to detect a diffuse astrophysical neutrino flux in different detection channels, it has not been able to find any neutrino sources to date.

The limited pointing accuracy as well as the low number of detectable events are key challenges to detecting neutrino sources. Many efforts have been put forth to further increase the detector's sensitivity as well as to improve the reconstruction of neutrino events. To further increase the discovery potential, a realtime alert system was implemented on-site at the South Pole [1, 9] to enable a multi-messenger approach. Through the realtime alert system, alerts will be sent out to the community when interesting neutrino events are detected within IceCube. The goal being that simultaneous multi-messenger data will be collected through follow up observations. With the help of a multi-messenger approach the mystery of the origin of astrophysical neutrinos might be solved. Although the realtime alert system has been up and running since 2012, no detection could yet be made [1].

For the realtime-alert system, events must be filtered and reconstructed on-site at the South Pole. However, this is very difficult as resources are very limited. Therefore, only simple and fast event reconstructions can be run at the South Pole. Even though more sophisticated and powerful reconstruction methods exist, these can take minutes to hours to reconstruct a single event, rendering them intractable for the use at the South Pole. Another difficulty is that the runtime of these reconstruction methods is highly dependent on the event. High-energy events will deposit more light in the detector, which in turn will increase the runtime of the event reconstructions. This, however, is problematic since the high-energy events are, in most cases, the events of interest. Furthermore, the bandwidth is limited to send data north for further processing. The high data rate as well as the strict hardware and bandwidth limitations are key challenges to the success of the real-time follow-up framework. To further complicate matters, the data present is in a very

raw and low-level format. Typically, a reconstruction starts at the point where only the calibrated pulses per digital optical module (DOM) are given.

These limitations call for a powerful and fast reconstruction method which can handle raw data and has a runtime that is ideally independent of the event type.

Typically, machine learning methods are a good choice for demands like these. While methods such as a Random Forest[16] or a boosted decision tree (BDT) [22] have shown great results, they can not easily deal with raw data. Artificial neural networks, on the other hand, are able to handle raw data. Recent advances in image recognition [27] have shown the capabilities of deep convolutional networks. These networks belong to the class of representation-learning methods [41]. They are capable of processing raw data and of creating an abstract level of representation. Once the networks are trained, their usage is computationally inexpensive. The network performs a set amount of mathematical operations on the input data resulting in a very stable runtime that is essentially independent of the input. These characteristics make deep learning methods an excellent candidate for powerful and fast on-site reconstructions.

In this thesis, a first study on the feasibility of reconstruction methods based on deep convolutional networks is presented. The developed reconstruction methods are intended to be used on-site at the South Pole to improve the online filters as well as the realtime-alert system. Through further improvements in the realtime-alert system and follow-up observation framework, neutrino sources might finally be detected.

2 The IceCube Detector

IceCube is a neutrino detector located at the South Pole instrumenting a cubic kilometer of glacial ice. The detector as seen in figure 2.1, consists of 5160 digital optical modules (DOMs) installed on 86 vertical strings at depths between 1450 m and 2450 m. These strings are deployed on an approximately triangular grid with

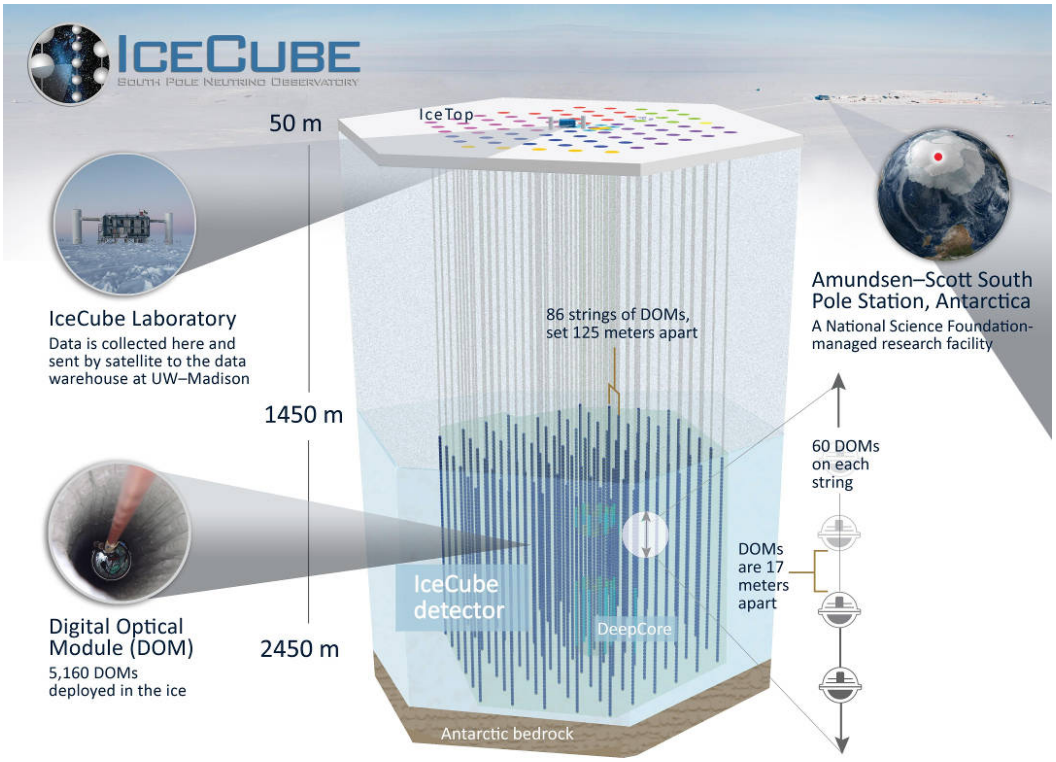


Figure 2.1: The IceCube detector consists of three detector parts: the main IceCube array, IceTop, and DeepCore. Data recorded by the digital optical modules (DOMs) is collected in the IceCube Laboratory. Online filtering and processing is applied limiting the data rate to about 100 GB per day [8, p. 61]. [32]

a string-to-string spacing of about 125 m. In order to detect neutrinos, IceCube

measures the Cherenkov photons produced by charged secondary particles resulting from neutrino interactions.

The primary goal of the IceCube detector is the detection of astrophysical neutrinos, which was achieved in 2013[3], as well as the identification and characterization of their sources. Although many efforts have been put forth to detect the sources of astrophysical neutrinos, to date, their origin remains a mystery. Ongoing work is put into the improvement of reconstruction methods to further increase the sensitivity of the detector. In addition, a multi-messenger approach is implemented to further increase the discovery potential through time-dependent analyses [1, 9]. Other objectives of the IceCube detector include indirect dark matter searches, searches for exotic particles as well as the study of neutrino oscillation physics. A detailed description of the IceCube detector can be found in [8].

The two primary detection channels consist of so called track-like events, induced by charged-current muon neutrino interactions of the type

$$\nu_\mu + N \rightarrow \mu + X, \tag{2.1}$$

as well as cascade-like events which result from charged current ν_e and ν_τ interactions in addition to neutral current interactions of all neutrino types [9, p. 6]. The pointing accuracy for track-like events is less than 1° and about 10 – 15° for cascades [9, p.6]. The focus in this thesis is on the reconstruction of track-like events.

2.1 Online Processing and Filtering

In order to enable real-time alerts and follow-up programs, an online processing and filtering (PnF) system is implemented at the South Pole. Triggered events undergo a base processing in which the raw DOM waveforms are calibrated and deconvolved to extract pulses consisting of the light arrival time and amplitude [5, pp.7-10]. After the base processing, further filters are applied.

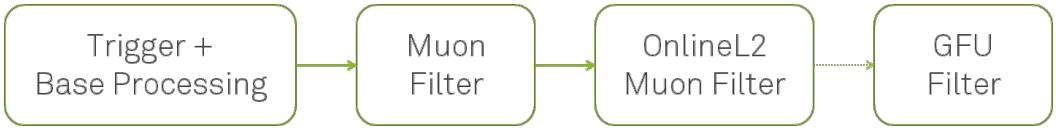


Figure 2.2: An example processing chain for the gamma-ray follow-up (GFU) filter is shown. The Muon Online L2 Filter is the basis for many different follow-up programs.

One of these filters is the Muon Filter [1, pp.6-7] which is the basis for many standard IceCube muon-neutrino analyses. The Muon Filter aims to select well reconstructed muons originating from any direction.

To further reduce the muon background, the Muon Online L2 Filter [1, pp.7-10] is introduced. It consists of additional advanced reconstructions as well as further event-selection cuts. The Muon Online L2 Filter is the basis of many follow-up programs such as the optical and X-ray follow-up (OFU) and gamma-ray follow-up (GFU) program [9, pp.13-18]. The GFU program searches for neutrino bursts on time scales of up to three weeks. In fig. 2.2, the basic processing chain for the GFU filter is depicted. The main focus in this thesis is on the Muon Online L2 Filter.

The trigger rate in IceCube is about 2.7 kHz resulting in data of approximately 1 TB/day [8, p.50]. After the Muon Filter is applied, the rate is reduced to about 45 Hz[1, p.7]. A further reduction in the event rate down to 2 Hz[1, p.9] in the up-going region is obtained through the Muon Online L2 Filter. The high data rate poses a key challenge for the PnF system as limited hardware capacities are available at the South Pole. Strict runtime requirements have to be fulfilled by reconstructions in order to prevent pileup. As a result, only basic and fast reconstruction methods can be run at the South Pole. A more in-depth description of the online processing and filtering system is given in [1, 8, 9]

2.2 Reconstruction Methods

Many different reconstruction methods exist within IceCube. In this thesis, the focus is on the relevant standard directional reconstruction (SplineMPE) as well as the standard energy reconstructions (Truncated Energy and MuEx). More sophisticated track reconstruction methods such as millipede[59] exist, but they are not further discussed, as they do not meet the hardware requirements imposed by the online processing and filtering system. Single events reconstructed by millipede can require hours of runtime.

2.2.1 Energy Reconstruction - Truncated Energy & MuEx

For through-going muons, the best achievable resolution on the reconstruction of the muon or neutrino energy is given by the muon energy at its entry into the detector. The standard energy reconstruction methods Truncated Energy[5, 11] and MuEx[5] exploit the linearity between the muon energy loss $\mathrm{d}E/\,\mathrm{d}x$ and its energy. This linearity, however, is only given for muon energies above approximately 1 TeV where the energy dependent stochastic energy losses begin to dominate over the

more or less constant ionization losses [36]. At lower energies, variables such as the measured track length in the detector provide better energy proxies. A summary of the energy reconstruction methods in IceCube is given in [5].

2.2.2 Directional Reconstruction - SplineMPE

Simple directional reconstructions such as Linefit or SPE fit are available at the Muon Filter level. These track reconstructions in addition to the MPE fit are described in [13]. The most advanced track reconstruction available online is SplineMPE. It is performed as part of the Muon Online L2 Filter.

SplineMPE maximizes a multi-photo-electron (MPE) likelihood [13, 55] given by

$$\mathcal{L}_{\mathrm{MPE}}\left(\vec{x}|\vec{\theta}\right) = \prod_{i}^{\text{1st hits}} n_i \cdot p(t_{\mathrm{res},i}|\vec{\theta}) \cdot \left(1 - P(t_{\mathrm{res},i}|\vec{\theta})\right)^{n_i - 1} \tag{2.2}$$

$$P(t_{\mathrm{res},i}|\vec{\theta}) = \int_{-\infty}^{t\mathrm{res}} p(t|\vec{\theta})\,\mathrm{d}t \tag{2.3}$$

where the probability density for an expected light arrival time $p(t_{\mathrm{res},i}|\vec{\theta})$ for a given track hypothesis $\vec{\theta}$ is obtained from spline tables instead of the parameterized Pandel function [48] as it is done in the MPE fit. A minimum ionizing muon with an infinite track length is used as the track hypothesis $\vec{\theta}$. The likelihood is built using the first pulse measured at a given DOM i as well as the total number of pulses n_i for the given DOM. Typically, the first light measured at a DOM (disregarding noise and prepulses) provides the most information as it is the least scattered light. More details on the SplineMPE reconstruction can be found in [55].

2.2.3 Uncertainty Estimation - Cramer Rao

In addition to the best fit of the track reconstruction, an estimate on the uncertainty is needed for point source searches and follow-up programs. Currently, the Cramér Rao and Paraboloid method are implemented in the online filters. However, the Paraboloid method[47] can only be run for a subset of low energy events due to its long run time. The Cramér Rao method is an analytical method that uses the Fisher information matrix and the Cramér Rao inequality [51]. The runtime of the Cramér Rao method is very fast and is subsequently run on all events. Both of the above mentioned track uncertainty estimators are heavily biased at higher muon energies, so that a bias correction needs to be performed. More information on the track uncertainty estimators in IceCube is given in [45, pp.36-47].

3 Artificial Neural Networks and Deep Learning

Artificial neural networks are not a new concept. During the 1970s and 1980s, the possibility of training simple multilayer neural networks with stochastic gradient descent and backpropagation was discovered independently by different groups [21, 39, 49, 60]. By the late 1990s, neural networks were mostly forsaken by the community which believed that their training was infeasible. Interest was revived in the early 2000s. Major breakthroughs in speech recognition [17, 46] as well as image recognition [27] have shown the capabilities of deep neural networks. This was made possible through the advent of fast GPUs[50], increasing datasets, as well as improvements in network architectures and training. A new regularization technique called dropout[56], ReLU[24] activiation functions, and data augmentation played a key role in the success story of deep neural networks.[41]

In comparison to shallow learners such as a Random Forest[16] or a boosted decision tree (BDT) [22], deep learning architectures do not rely on feature engineering. Deep learning architectures such as deep neural networks belong to the class of representation-learning methods. They are capable of processing raw data and of creating abstract levels of representation. Deep learning does not only consist of deep neural networks, but often these terms are used interchangeably. An overview of deep learning is given in [41].

3.1 Neural Network Architectures

Many of the neural network components and architectures mimic the human brain and visual system. In the following, important components and architectures are introduced.

Artificial Neuron (Perceptron)

A neural network is built out of small components called artificial neurons or perceptrons. These neurons are connected to other neurons as illustrated in fig. 3.1. In a basic feed forward network, information flows in one direction. An artificial neuron retrieves information over the inputs a_i. These can either be the outputs

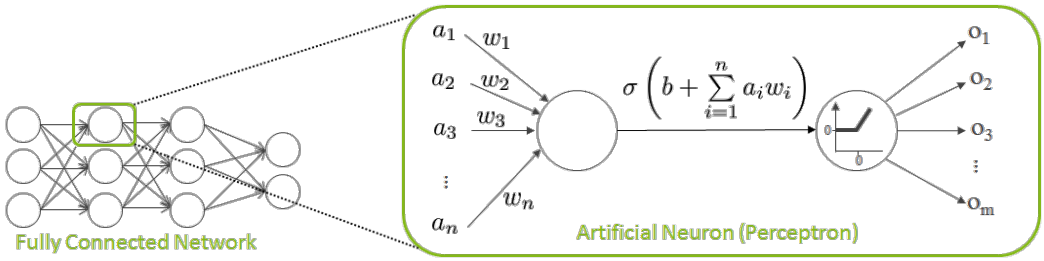


Figure 3.1: An artificial neuron (perceptron) is the basic building block of neural network architectures.

o_j of previous neurons, or the data which is fed into the network. A weighted sum is performed over the input and a bias term is added. Afterwards a non-linear activation function $\sigma(x)$ is applied. The n weights w_i and the bias b are free parameters which are tuned to the data during training.

Typically, rectified linear units (ReLU):

$$\sigma_{\text{ReLU}}(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \tag{3.1}$$

are used as activation functions. Previously, sigmoid functions were commonly applied. However, the use of ReLUs significantly facilitates and speeds up training of the neural network.[24]

Fully Connected Layers

When combining artificial neurons to build a neural network, the neurons are usually assorted into layers. On the left of fig. 3.1, a fully connected network with 4 layers is shown. The first layer is the input layer, while the last layer defines the output of the network. The intermediate layers are referred to as hidden layers. In a fully connected layer every neuron is connected to every neuron of the previous layer.

Every neuron in a fully connected layer has $n + 1$ free parameters, where n defines the number of neurons in the previous layer. For a given layer of k neurons, this results in $(n + 1) \cdot k$ free parameters. Therefore, fully connected layers do not scale very well due to the prohibitively high number of free parameters in big networks. Often convolutional layers are a better option.

Convolutional Layers

Convolutional layers exploit translational invariance and locality to greatly reduce the number of free parameters. Instead of a neuron being connected to every neuron in the previous layer, it is only connected to nearby neurons as illustrated in fig. 3.2. Finding features (edges, shapes, color gradients) in the bottom right corner of

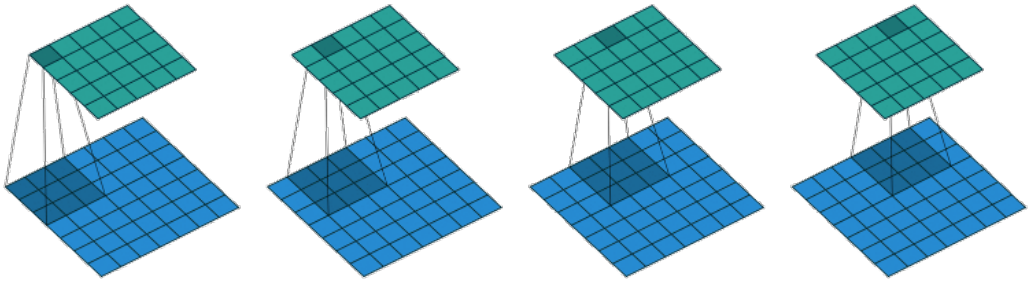


Figure 3.2: The computation of a 2-dimensional convolutional layer is shown. A kernel (set of shared weights) is shifted across the input layer (blue) to obtain the output (green). In order to obtain the output for the shaded neuron in the output layer, the shaded part of the input layer is weighted by the 3×3 kernel and summed up. Afterwards a bias is added and an activation function is applied. The green output layer is also referred to as a feature map.

an image is not dependent on the pixels in the top left corner. Therefore, these connections can be omitted. Additionally, the weights are shared between neurons. In doing so, translational invariance is exploited: a useful feature in one part of the image should also be useful in another part of the image.[43]

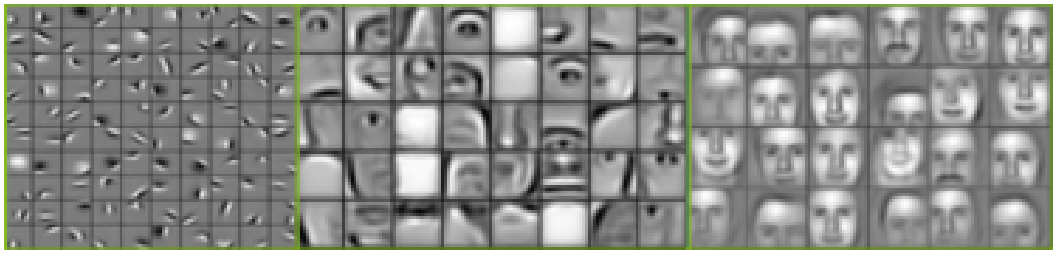


Figure 3.3: Feature maps of different layers of a face recognition model are shown. The feature maps on the left correspond to an earlier layer in the network, while the feature maps on the right originate from a deeper layer. Deep neural networks are able of extracting hierarchical representations from raw data.[44]

The shared weights are also referred to as convolution kernels or filters. In fig. 3.2

the convolution is illustrated by a single kernel which is shifted across the input layer (blue) to obtain the output (green). However, a convolutional layer is not limited to a single kernel. Every kernel of a convolutional layer creates an output layer (feature map). In fig. 3.3 feature maps for 3 different layers of a face recognition model are shown. Similarly, a convolutional layer can have more than one input layer. Within this thesis these are referred to as the channel input. As an example, a colored image of $n \times k$ pixels can be decoded into 3 images for the each of the RGB values. The result is a $n \times k \times 3$ tensor of pixels, where the last dimension is the channel input dimension.

Residual Nets

In residual nets the layers are reformulated to learn residual functions. Instead of the output $f(x)$ of a standard layer, a residual layer outputs

$$f_{\text{res}}(x) = x + f(x) \tag{3.2}$$

where x is the input into the layer. Residual nets ease the training of deep networks. A detailed description is given in [26].

Nonlinear Function

A neural network can be composed of any combination or modification of the previously described components (additional components are possible). In general, the network architecture defines a non-linear mapping of the input I to the output O:

$$f_\theta : I \rightarrow O \tag{3.3}$$

I : Model input
O : Model output
θ : Free parameters

The components of the network architecture define the free parameters θ. During training these parameters are optimized in a way that $f_\theta(x)$ maps the data.

3.2 Training of Neural Network Architectures

Supervised training of neural network architectures requires labeled training data. A loss function is defined which is minimized during training. See ch. 6 for details on the loss function used in this thesis. Once the loss function is defined, batches of n examples are drawn from the training data. The data is fed into the network to obtain the output $f_\theta(x)$. Afterwards the loss function $L\left(f_\theta(x), y_{\text{true}}\right)$ can be evaluated where y_{true} is the truth as defined by the label. This first step is illustrated on the left of fig. 3.4. Initially, the output of the network $f_\theta(x)$ will be random, due to the random initialization of the free parameters θ of the model.

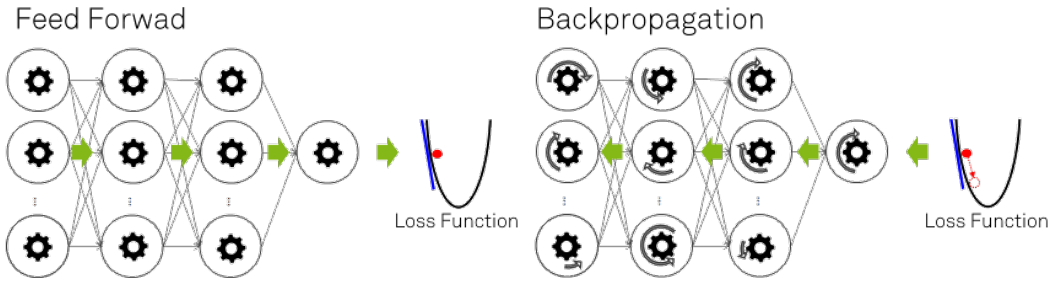


Figure 3.4: The data is fed into the network on the left to obtain the loss value (red dot). Backpropagation is then used to calculate the gradients for each of the free parameters. These are then adjusted via gradient descent. As a result, the loss value reduces.

In the second step, the free parameters of the model are adjusted to minimize the loss function. This is performed with stochastic gradient descent. The gradients of the loss function in regard to the free parameters $\frac{\partial L}{\partial \theta}$ are calculated for the given batch of data via backpropagation, which in principle is simply the application of the chain rule. These steps are repeated many times, until the loss converges in a minimum.[40, 41]

3.3 Batch Normalization

During training the distribution of the input into a given layer changes, which complicates the training procedure. This is also referred to as internal covariate shift. In practice it is helpful if the input of a layer is normalized with a zero mean and unit variance. Batch normalization normalizes the layer inputs in order to combat internal covariate shift.[33]

3.4 Dropout

Dropout is a regularization technique which randomly drops units in a given layer during training. In doing so, units are prevented from excessively co-adapting. Details are given in [56].

3.5 Frameworks and Datasets

Many different deep learning frameworks exist. Some examples are:

- TensorFlow[1]
- Theano[2]
- Keras[3]
- Caffe[4]
- Torch[5]
- MXNet[6]
- Microsoft Cognitive Toolkit[7]

An overview of the different frameworks is shown in [57]. Within this thesis TensorFlow 1.2[10] is used in combination with python 2.7. In order to evaluate the performance of deep learning architectures, benchmark datasets exist. Common image datasets are listed in tab. 3.1.

Dataset	Description	Image Size	Dataset Size	Per Class
MNIST [42]	Handwritten digits	28×28	60,000	6000
CIFAR 10 [38]	Color images in 10 classes	32×32	60,000	6000
CIFAR 100 [38]	Color images in 100 classes	32×32	60,000	600
Imagenet [19] (ILSVRC15 [54])	Color images in 1000 classes	Typically cropped: 256×256	$1,281,167$	732-1300

Table 3.1: Common benchmark datasets for image recognition tasks are shown.

[1] https://www.tensorflow.org/
[2] http://deeplearning.net/software/theano/
[3] https://keras.io/
[4] http://caffe.berkeleyvision.org/
[5] http://torch.ch/
[6] http://mxnet.incubator.apache.org/
[7] https://www.microsoft.com/en-us/cognitive-toolkit/

4 Data, Label Generation, Formatting, and Preprocessing

4.1 Monte Carlo Dataset

An IceCube muon-neutrino simulation dataset (11069) is used with a simulated neutrino spectrum of E^{-1} and an energy range from 100 GeV to 10 PeV. For the results shown in this thesis, the neutrino spectrum is reweighed to an unbroken power-law flux with $E^{-2.19}$ according to [25]. Only track-like events are used, where the muon resulting from the charged-current neutrino interaction comes within 60 m of the convex hull around the detector. Approximately 72,000 events are used for the validation set, while the training dataset consists of approximately 2 million events. An additional 480,000 events are held out during training and validation. These events are used as a test set to obtain the results shown within this thesis.

4.2 Label Generation

The deep convolutional neural network as presented in this thesis is a supervised machine learning approach, hence, labels for the training data must first be defined. A total of 22 labels are chosen. Some example labels are quantities such as the neutrino and muon energy, the direction of the particle, as well as the point of entry into the detector. The software used to extract the necessary labels from the Monte Carlo data is implemented within the Icetray[1] framework [20]. A complete list of all extracted labels as well as a brief description can be found in table A.1.

4.3 Data Input Format

Convolutional neural networks were first developed for the domain of image recognition[43]. Feeding image data into a neural network is straight forward, as it only has two dimensions with an optional channel input for the different color channels. The

[1] http://software.icecube.wisc.edu/documentation/

pixels of an image are arranged on an orthogonal grid and for typical image datasets (ch. 3.5) the pictures are all of the same size or can easily be cropped to the same size. In contrast, IceCube data is four dimensional, arranged on an approximately triangular grid and highly variable in size. These and other complications need to be addressed, before the data can be fed into a neural network. In the following, the IceCube data is described, core issues are explained, and possible solutions are presented.

4.3.1 IceCube Data

As mentioned in ch. 2, the IceCube detector consists of 5160 DOMs installed on 86 vertical strings. The geometry of the IceCube detector is illustrated in fig. 4.1, where an on-top view is shown. At trigger level, the data consists of recorded

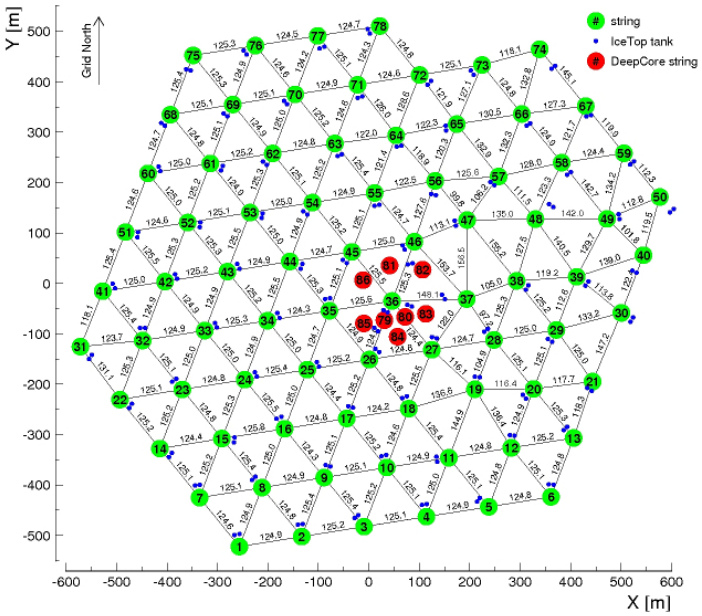


Figure 4.1: An on-top view of the IceCube detector is shown. The in green depicted 78 strings are on an approximately trinangular grid, while the DeepCore strings painted in red are installed in a denser configuration.

waveforms (see section 2.1). The recorded waveforms have a bin width of about 3.3 ns (ATWD waveform[8]). Each DOM can measure an arbitrary number of waveforms with variable starting times in a single event. While a typical event has a read out window of approximately 15 000 ns, this can vary depending on the

trigger [8, pp.56-59]. Single pulses can be extracted out of these waveforms. In the end, the data can be reduced to a pulse series of arbitrary length for each DOM, where each pulse has a time and charge associated to it. The input length is highly variable and different for each DOM. The 86 strings of the detector are divided into two detector parts. While the main IceCube array, consisting of the first 78 strings, is arranged on an approximately triangular grid, the 8 DeepCore strings do not follow this symmetry. They are installed in a denser configuration with variable distances to neighboring strings. The strings of the main IceCube array have an inter-string spacing of about 125 m. However, there are deviations as shown in fig. 4.1.

4.3.2 Core Issues

The architecture of a convolutional neural network must be predefined and is then kept fixed. In doing so, the input into the network has to be of the same format and dimensionality for every event. IceCube data, however, is highly variable in length as mentioned above. Therefore, the variable format of the data needs to be unified.

A naive approach might be to divide the event read out window into time bins and to pad with zeros, where no waveforms or pulses are measured. The issue with this approach lies in the high dimensionality of the problem. If a binning of 5 ns in time were to be used, this would result in approximately 8000 time bins for each of the 5160 DOMs in order to cover the whole event. This can be reduced to about 1200 time bins per DOM, if a time window of approximately 6000 ns is used, which is sufficient to cover the full time needed for a muon of 1 GeV to transverse the longest possible stretch within the detector. This still results in a minimal input size of $5160 \times 1200 = 6192000$ nodes. In comparison, typical image datasets (ch. 3.5) for image recognition tasks consist of input dimensions of $28 \times 28 = 784$ (MNIST [42]), $32 \times 32 = 1024$ (CIFAR 10/100 [38]), or are typically cropped to $256 \times 256 = 65536$ (Imagenet [19]). In addition, the data becomes extremely sparse which can cause problems for the deep learning approach. A fine uniform binning in time over the relevant part of the event is therefore not feasible. Increasing the width of the time bins introduces other issues as the directional reconstruction heavily depends on exact timing information.

Another issue compared to image data, is that IceCube data is four dimensional. In particular for the directional reconstruction, the time dimension is of the utmost importance. Reducing the dimensionality of the problem results in a loss of spatial and temporal relations that have to be compensated for in a different way.

Moreover, the IceCube detector is separated into two detector parts which do not share the same geometry or symmetries. The main IceCube array is arranged in an approximately triangular grid. However, pixels in an image are typically arranged on an orthogonal grid. Data storage, matrix and tensor operations are performed on orthogonal grids. Therefore, the IceCube data must first be transformed to an orthogonal grid.

IceCube data is very high-dimensional and variable. Handling of the data is rather complex compared to image data. As a first step, a viable solution to these issues needs to be found.

4.3.3 Possible Solutions

When regarding possible solutions to the above explained issues, it is important to first decide on a dimensionality of the data representation. Some options are presented in the following sections.

Spatial and Temporal Dimensions embedded in Architecture [4D]

In order to maintain both the spatial and temporal dimensions in the network architecture, a full four dimensional input is necessary. The charge of the measured pulses, or directly, the charge of the waveforms can be binned in time. For the binning in the spatial coordinates, one option is to divide the IceCube detector into bins along an orthogonal coordinate system, combining multiple DOMs in a single bin. Alternatively, an interpolation could be performed to sample the detector on an orthogonal grid. With these approaches, the triangular grid of the IceCube array is automatically dealt with.

However, the interpolation or up-sampling in the spatial dimensions will smear out the data. To avoid this, the DOMs themselves can be used as a bin. For this DOM-binning, the triangular grid of the DOMs needs to be transformed to an orthogonal grid. This can be achieved by the transformation described in fig. 4.2. The DeepCore part of the detector is handled separately, reducing the x and y-coordinates to a single dimension.

As a result, two input tensors of the shape $[10 \times 10 \times 60 \times n]$ and $[8 \times 60 \times n]$ are obtained for the main and DeepCore array, respectively, where n denotes the number of time bins. As mentioned above, the number of time bins has to be reduced in order to make this approach feasible. For this, the width of the time bins could be increased at the cost of timing resolution in order to cover the same time window

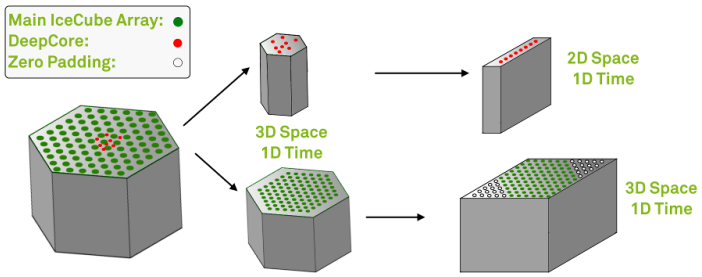


Figure 4.2: The main IceCube array and DeepCore strings are handled separately. Hexagonally shaped data on the left (green dots) can be transformed into an orthogonal grid on the right by padding with zeros (white dots) and aligning the rows. Every DOM defines a bin in the spatial coordinates (DOM-binning).

with less bins. With a four dimensional convolution over this input, translational invariance in space and time as well as locality can be exploited.

A different approach is to choose individual time windows for every DOM. In doing so, the time windows could be chosen smaller to only cover the relevant time of the measured waveforms of each DOM. This reduces the necessary number of bins while maintaining timing accuracy. However, the time bins between DOMs will no longer be aligned, making a 4D convolution problematic. Nevertheless, a 1D convolution could be performed over the time dimension for every DOM to extract useful values summarizing the distribution of the pulses/waveform. The extracted values could then be used as channel input into a 3D convolution over the spatial dimensions as described below.

Spatial Dimensions embedded in Architecture [3D + Channel]

In order to reduce the dimensionality, a four dimensional input format can be adopted where the last dimension is used as a channel input into a 3D convolution. The first three dimensions of the input tensor correspond to the three spatial dimensions of the DOMs, while the fourth dimension is used to summarize the pulse series of arbitrary length into a fixed amount of values. These values could for instance include the time of the first pulse, the total charge measured, or the spread of pulses in time.

As also for the fully 4D case previously described, the binning in the spatial dimensions can be performed through interpolation or up-sampling on an orthogonal grid as well as on a DOM-level as described in fig. 4.2. For a DOM-binning the resulting input tensors for the main IceCube array and the DeepCore part have the

shape $[10 \times 10 \times 60 \times c]$ and $[8 \times 60 \times c]$, respectively, where c is the number of channel values. Spatial relations such as locality and translational invariance in space are therefore exploited by the network architecture through the 3D convolutional layers.

Temporal Dimension embedded in Architecture [1D + Channel]

The measured pulses themselves can be used as input. For instance, one can use the n highest charged pulses and sort these in time. If an event has less than n measured pulses, a padding with zeros can be performed to maintain a fixed input dimension. In doing so, the two detector parts can be handled uniformly. The input tensor is of the dimensions: $[n, p]$, where n is the number of pulses and p is the number of parameters describing the pulse. These p parameters can for instance include:

- pulse coordinates x, y and z
- pulse charge
- pulse time
- charge of pulses in neighboring DOMs within a certain time window

Spatial information is not directly embedded in the network architecture for this approach, but included in the channel input. A 1D convolution can be employed to make use of temporal relations. In addition, timing information is included in the channel parameters as the pulses are not evenly distributed in time. Instead of a 1D convolution a locally connected architecture can be used to exploit locality, but reduce constraints for translational invariance.

An approach like this reduces sparsity in the input data as well as redundant operations and costly high-dimensional convolutions due to the smaller and lower dimensional input shape. The result is a faster, but possibly less accurate network. Spatial information has to be passed within the channel dimension and the network has to learn how the given spatial information can be used. The inclusion of prior knowledge of spatial relations is limited to passing more and pre-calculated information within the channel dimension. It cannot be directly embedded in the network architecture with this approach.

The gain from having spacial or temporal dimensions incorporated into the network architecture lies in the fact that a priori knowledge about spatial and temporal

relations can be exploited by the use of convolutional or locally connected layers. Therefore, these relations are somewhat hard coded into the network and do not have to be learned. Hence, the number of free parameters can be reduced. If the dimensionality of the input into the network is reduced, the loss of spatial and temporal relations have to be compensated for within the channel input. This can for instance be done by adding information about the neighboring pulses in the 1D convolution case, or in the case of the 3D convolution by adding a summary of the timing information of the pulse series.

The proposed solutions are just some examples of how the data could be transformed to feed into the convolutional network, but it is by no means exhaustive. Particularly in regard to the pulse series of arbitrary length, one might chose to use recurrent neural networks such as long short-term memory networks (LSTM)[23, 28], which can naturally handle input series of arbitrary length. There are many possible choices which remain to be investigated. The main focus in this thesis is on the 3D + channel approach which incorporates the spatial dimensions into the network architecture and convolution layers while reducing the time dimension to a number of values summarizing the distribution of the measured pulses.

4.4 Preprocessing

Deep neural networks can in theory process data of any range and scaling. However, the activation function's nonlinearity is centered around zero. Moreover, error contours are elliptical if input features have very different scaling. This causes practical problems in regard to gradient descent as demonstrated in fig. 4.3. In

Figure 4.3: The contours of an elliptical and circular error surface are shown. For a given point on the circular surface, the gradient vector points towards the minimum. This is not necessarily the case for elliptical contours.

addition, if input data is not normalized, small imbalances in the early layers of the network can cascade throughout neural networks causing gradients during backpropagation to explode. This effect is even stronger for deep neural networks.[35, 40]

It is therefore useful to normalize the input data and labels. Within this thesis, the data and labels are normalized to zero mean and unit variance by the following transformation:

$$X' = \frac{X - \overline{X}}{\sigma_X + \epsilon}, \tag{4.1}$$

where $\overline{X}$ is the mean of X, σ_X is the standard deviation of X, and $\epsilon = 10^{-8}$ is a small constant to prevent division by zero. For constant attributes, σ_X is set to 1 in equation (4.1).

In addition, steps are undertaken to maintain a standardized input in every layer throughout the network. This can be accomplished by the use of batch normalization, residual additions and unit variance maintaining layers as described in ch. 3.3, 5.3, 5.4, respectively.

Input features and labels which span over many powers of magnitude are first transformed via

$$X' = \log_{10}(C + X), \tag{4.2}$$

before they are normalized by equation (4.1). For variables with many values close to 0, the constant C is given by 10^{-4}, otherwise $C = 1.0$.

The values $\overline{X}, \sigma_X$ and $\overline{Y}, \sigma_Y$ in equation (4.1) for the transformation of the input data X and the labels Y are calculated on the training data. For the input data X, the channel values are treated equally across different DOMs. However, entries in the input data tensor that do not correspond to a real DOM are filled with zeros and handled separately. These entries are constantly 0 and remain 0 after the transformation in equation (4.1).

5 Implemented Software

For the realization of the deep learning based reconstruction as presented in this thesis, additional software is implemented. In the following, some of the implemented software is described.

5.1 Data Input Framework

In order to efficiently train the neural network, loading, preprocessing and feeding of the data into the network have to be highly optimized. The simulation dataset (about 1 TB of disk space) is first preprocessed within the IceTray framework [20]. During this preprocessing step, the labels and input variables are calculated for every event. Afterwards these files are written to disk in hdf5 files in a highly compressed format, discarding all information that is not needed for training. As a result, the necessary disk space is reduced by approximately 96 %. During training, reading and decompression of the hdf5 files is handled in parallel on multiple CPUs. Each hdf5 file contains about 350 events. Once read and decompressed, the input tensor X as well as the labels Y of these approximately 350 events are combined from the multiple processes and put in a single shared queue. An additional process dequeues the elements and produces batches of a given size, which are then fed into the neural network.

Due to the high dimensionality of the input and the restrictions given by the computing architecture, only a limited amount of files can be loaded into RAM at once. In order to reduce the amount of times files need to be read and decompressed, a given number of n files are loaded into memory at once. Of these approximately $n \cdot 350$ events, batches are randomly drawn until a specified amount of k epochs are completed. Afterwards, the next n files are loaded into memory.

5.2 Hexagonal Convolution Kernels

The convolutions within the TensorFlow framework are performed on orthogonal grids. Transforming the hexagonally shaped IceCube data into an orthogonal grid

as illustrated in fig. 4.2 results in convolution kernels shaped as parallelograms in the detector. To instead obtain convolutional kernels which are shaped according to the IceCube geometry, the convolution kernels within TensorFlow are adjusted by setting the corner elements to zero as illustrated by the red points on the right of fig. 5.1. As a result, hexagonally shaped convolution kernels in the detector are

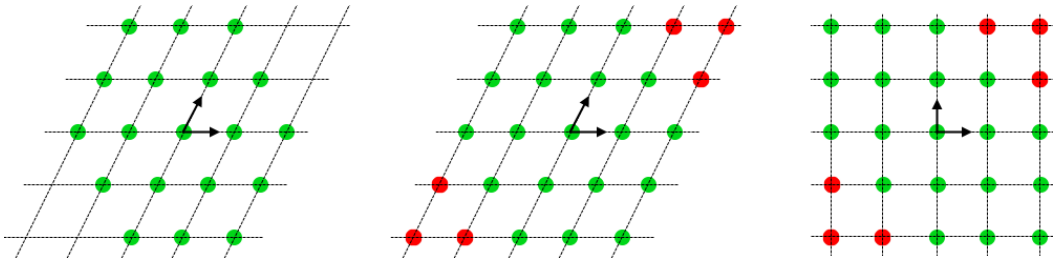


Figure 5.1: Hexagonally shaped convolutional kernels on the left (green dots) can be transformed into an orthogonal grid on the right by padding with zeros (red dots) and then shifting the rows, so that they align.

obtained. The obtained kernels are defined by a tuple of size and orientation as illustrated in fig. 5.2.

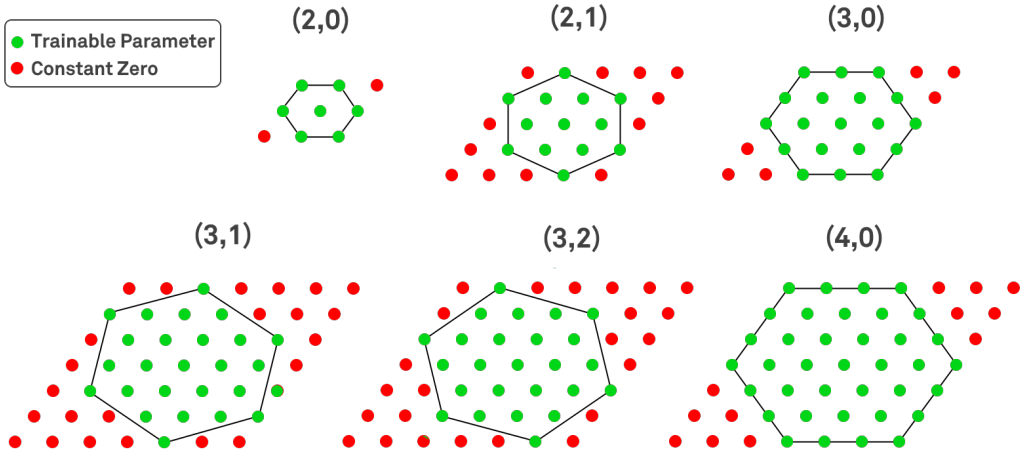


Figure 5.2: A hexagonally shaped kernel is defined by a tuple of size s and orientation o: (s, o). The size defines the number of edge-points of an aligned hexagon (orientation $o = 0$).

5.3 Residual Additions

As described in ch. 3.1, residual nets [26] can greatly facilitate the training of deep neural networks. The concept is adopted here with slight modifications applied.

Layers without residual additions return the output of the activation function X_{output}. In residual layers as defined in this thesis, this output is modified to X'_{output} according to

$$X'_{\text{output}} = X_{\text{input}} + s \cdot X_{\text{output}} \tag{5.1}$$

where s is a scale factor defining the contribution of the residual and of the given layer. If $s = 0$, the output of the layer is equal to the input: $X'_{\text{output}} = X_{\text{input}}$. The scale factor s is initialized to very small random values. As a result, the given layer will initially solely pass the input along. During training, the scale factor s is optimized and the contribution of the residual increases.

In convolutional layers, the number of input channels and output channels are often different, hence, the dimensions of X_{input} and X_{output} do not match. In this case, only the channels which align are treated as residuals according to eq. (5.1).

5.4 Unit Variance Maintaining Layers

As already mentioned in ch. 4.4, there are great benefits to keeping the input and output normalized within the neural network. This is also the goal of batch normalization [33], which explicitly normalizes the input into a specific layer. Another possibility for maintaining a normalized throughput, is to make sure that layers maintain unit variance of the input. With the assumption of a zero centered input with unit variance, the layers of the neural network can be modified by a term controlling the variance of the output.

The addition of independent random variables with variances s_i^2, increases the variance to

$$s_{\text{ges}}^2 = \sqrt{\sum_i s_i^2}. \tag{5.2}$$

To compensate for this, the result of the convolution in a convolutional layer is divided by the square root of the number of input channels: $\sqrt{\text{num input channels}}$. In a fully connected layer, the result of the matrix multiplication of the input X and the weights W is divided by the square root of the number of inputs: $\sqrt{\text{num inputs}}$.

In addition to the modifications described above, weights, biases and scale factors are initialized in a way that the layers of the neural network initially perform an identity operation. This initialization scheme in combination with the unit variance maintaining layers as well as the normalization of input and labels as described in ch. 4.4 greatly speeds up and facilitates the training procedure. As a result, the initialized but untrained neural network can immediately reproduce a reconstruction accuracy obtained by guessing the label given its distribution.

6 Training of Neural Networks

The dataset is divided into three parts as mentioned in ch. 4.1. Training is performed on the training set while the validation set is used to validate the results and to check for overfitting. The test set is not touched until everything is fully optimized and completely trained. It is used to create the result plots shown in this thesis.

The residuals between predicted and true label for a mini batch of B events is given by

$$\Delta Y = |Y_{\text{true}} - Y_{\text{pred}}|, \tag{6.1}$$

where ΔY is a $B \times L$-matrix with L as the number of labels. The residuals are weighted by an importance vector $\vec{C}$ via

$$(\Delta Y_w)_{b,l} = \vec{C}_l \cdot (\Delta Y)_{b,l}\,. \tag{6.2}$$

The mean squared error (MSE) and the weighted mean squared error for a label l is defined by

$$\left(\overrightarrow{\text{MSE}}\right)_l = \frac{1}{B}\sum_{b=0}^{B} (\Delta Y)^2_{b,l} \tag{6.3}$$

and

$$\left(\overrightarrow{\text{MSE}_w}\right)_l = \frac{1}{B}\sum_{b=0}^{B} (\Delta Y_w)^2_{b,l} \overset{(6.2)}{=} \vec{C}_l^2 \cdot \left(\overrightarrow{\text{MSE}}\right)_l, \tag{6.4}$$

respectively.

All labels are trained at once in a single neural network with one optimizer. The importance vector $\vec{C}$ can be used to distribute importances to single labels. This is useful, because some labels are easier to reconstruct as others. In addition, the importance vector can be updated every N optimization steps, to ensure that each label contributes to the loss function according to the importance assigned to it. This compensates for the fact that different labels are trained at different speeds. The importance vector is updated via:

$$\left(\vec{C}'\right)_l = \left(\vec{C}_0\right)_l \cdot \max\left(\frac{1}{\sqrt{\left\langle\overrightarrow{\text{MSE}}\right\rangle_l}}, 1\right), \tag{6.5}$$

where $\vec{C}_0$ is the unmodified, original importance vector and $\overrightarrow{\langle \mathrm{MSE} \rangle}$ is given by

$$\overrightarrow{\langle \mathrm{MSE} \rangle} = \frac{1}{N} \sum_{n=0}^{N} \overrightarrow{\mathrm{MSE}}_n \tag{6.6}$$

with the mean squared error $\overrightarrow{\mathrm{MSE}}_n$ for the n-th mini batch.

Optionally, the events can additionally be weighted by the physical weight of the simulation. This changes the weighted residuals from equation (6.2) to

$$(\Delta Y_w)_{b,l} = \vec{W}_b \cdot \vec{C}_l \cdot (\Delta Y)_{b,l} \,, \tag{6.7}$$

where $\vec{W}$ are the physical weights for the B events in a mini batch. The weighted mean squared error from equation (6.4) is then used to construct the loss function

$$\mathrm{loss} = \sum_{l=0}^{L} \left(\overrightarrow{\mathrm{MSE}}_w \right)_l \,. \tag{6.8}$$

In order to monitor the learning process independently of the chosen importance vector $\vec{C}$ and weighting scheme $\vec{W}$, a benchmark value is defined as the sum of the root mean squared errors (RMSE) over all labels:

$$\mathrm{Benchmark} = \sum_{l=0}^{L} \sqrt{\left(\overrightarrow{\mathrm{MSE}} \right)_l} \tag{6.9}$$

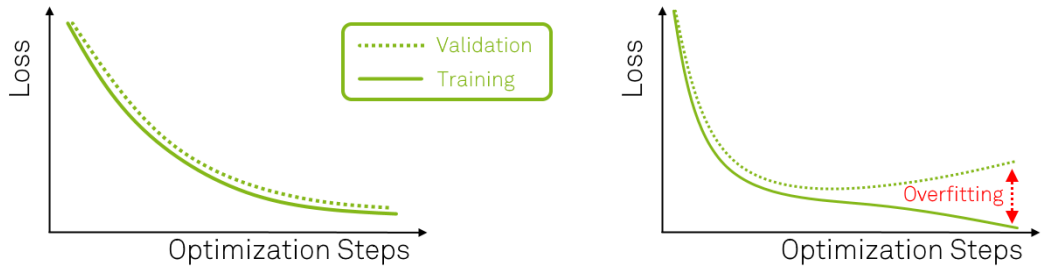


Figure 6.1: Overfitting of the training data can be monitored by evaluating the loss curves for the training and validation set. On the left, a typical loss curve is shown in the case of no overfitting, whereas the case for overfitting is shown on the right.

The loss function (6.8) is minimized with the ADAM-optimizer [34] within the TensorFlow framework [10]. The general learning scheme is to start training with a high dropout rate, forcing the network to learn robust features. Over time the

dropout and learning rate are reduced. While reducing the dropout rate, the training process is carefully monitored for signs of overfitting. When the network starts to overfit the training data, a gap will develop between the training and valdiation loss curves as shown in fig. 6.1. In this case the training is stopped early.

The mean squared error heavily punishes outliers. In general, this is useful as the network is forced concentrate on learning to better reconstruct the badly mis-reconstructed events. However, the difficulty of the reconstruction greatly varies between events. While the reconstruction of some events might be straight forward, others can be ambiguous. At a certain point, the network can not possibly learn to better reconstruct the outliers. Yet, due to a loss function such as the mean squared error, it is still forced to focus on these events. Consequently, the reconstruction of events, which the network could in principle reconstruct more accurately, is not further optimized. A more outlier-robust loss functions such as the absolute error, huber loss [29], or tukey loss [15] function can be utilized to counteract this effect. These loss functions are illustrated in fig. 6.2.

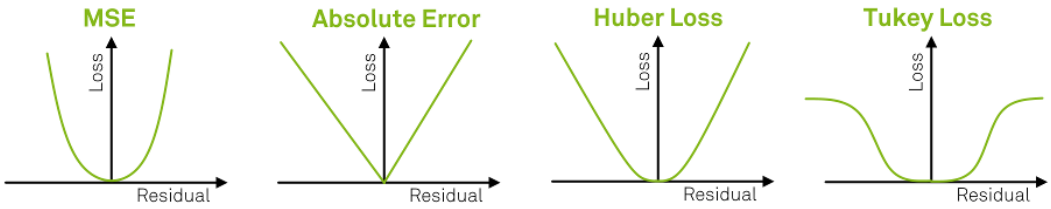


Figure 6.2: A diagram of different loss functions is displayed. The residuals are plotted on the x-axis and the corresponding loss on the y-axis.

7 Neural Network Architectures

In the following sections, the different network architectures used within this thesis, including their training procedures are explained.

7.1 All-rounder Model

As stated in ch. 4.3.3, the main focus in this thesis is on a four dimensional [3D + channel] input format with three spatial coordinates and one dimension for parameters summarizing the pulse series. The architecture of the all-rounder model described here uses this data representation with a DOM-binning as illustrated in fig. 4.2. The pulse series of a DOM are summarized in the following nine values:

- Sum of pulse charges[1]
- Sum of pulse charges within 500 ns of first pulse[1]
- Sum of pulse charges within 100 ns of first pulse[1]
- Time of first pulse
- Time at which 20 % of charge is deposited in DOM
- Time at which 50 % of charge is deposited in DOM
- Time of last pulse
- Charge weighted mean time of pulses
- Charge weighted standard deviation of pulse times

As a result of the DOM-binning and the above mentioned 9 input values, the input tensor is of the form $[10 \times 10 \times 60 \times 9]$ and $[8 \times 60 \times 9]$ for the main IceCube array and DeepCore, respectively. The network architecture is summarized in fig. 7.1. A detailed description can be found in the appendix A.1.1.

The architecture of the described neural network model is used to reconstruct the labels in table A.1. It is designed to be an all-rounder model capable of reconstructing

[1] Transformation (4.2) with $C = 1$ is performed before normalization

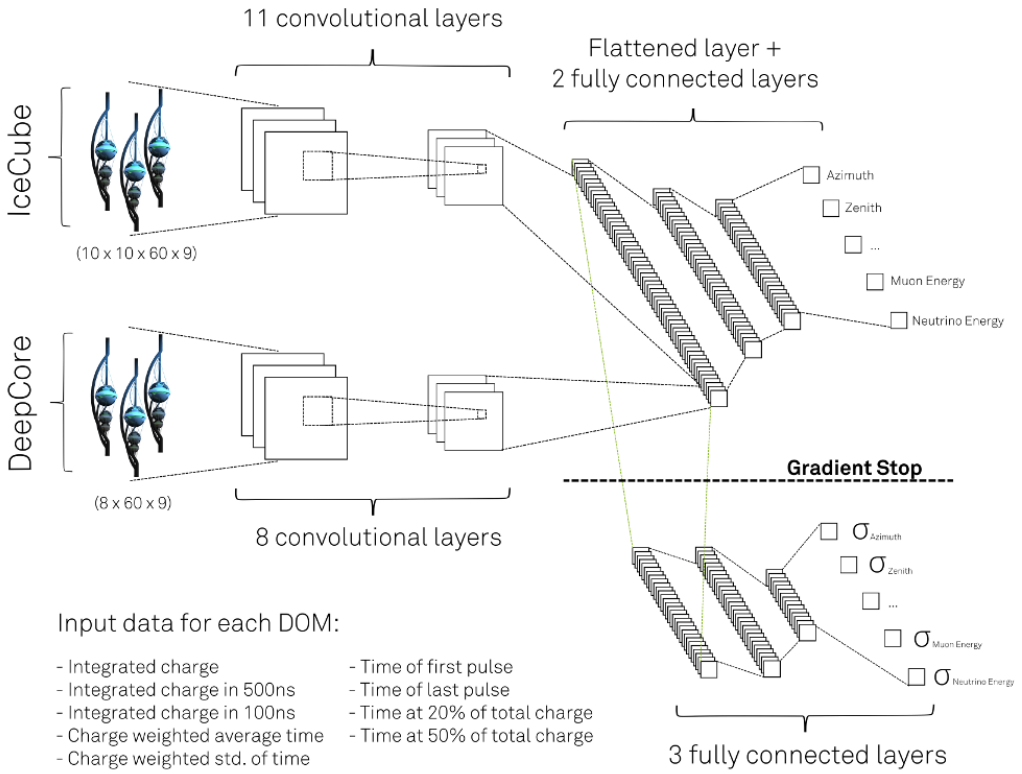


Figure 7.1: The architecture of the all-rounder model is illustrated. The main IceCube array and DeepCore input are treated seperately with independent convolutional layers. The result is flattened, combined and used as an input into the fully connected layers.

energy related variables as well as directional information. The neural network is trained in five steps as depicted in tab. A.5. Training takes about 1 to 2 days on a Tesla P40 and the resulting loss curves are plotted in fig. 7.2.

Figure 7.2: The training and validation loss curves are shown. A total of 87500 optimization steps are performed. This corresponds to about 11 data epochs.

7.1.1 Fine-Tuned Models

The all-rounder model is trained on events passing the Muon Online L2 Filter. To further improve the reconstruction accuracy of the neural network for different filters, the all-rounder model can be fine-tuned to the specific use case. In the following, some examples are given. The first three training steps as described in tab. A.5 are kept, while the last two training steps are replaced as noted below.

Tuned to GFU Filter

The GFU filter is an online filter designed for online analyses as described in ch. 2.1. It is applied after the Muon Online L2 Filter. To further improve the resolution of the deep learning approach on the GFU filter, the all-rounder model can be fine tuned by exchanging the last two training steps in tab. A.5 with the steps listed in tab. A.6.

Tuned to Final Level

Events in the point source final data level are a subset of the events passing the Online Muon L2 filter. These events are directly used in point source analyses.

Similarly to the GFU filter, the all-rounder model can be fine tuned to the point source final level by exchanging the last two training steps in tab. A.5 with the steps listed in tab. A.7.

Tuned with Tukey Loss

The all-rounder model is trained with a mean squared error loss function. To decrease the sensitivity to outliers, a more robust loss function such as the tukey loss [15] can be used in training. The first three training steps in tab. A.5 are kept, while the last two steps are replaced with the steps listed in tab. A.8.

7.2 Track Uncertainty Model

In order to estimate the uncertainty on the standard track reconstruction, SplineMPE (ch. 2.2.2), the all-rounder model as described in ch. 7.1 is slightly modified. The network architecture of the track uncertainty model is the same as the all-rounder model. However, in addition to the previously mentioned 9 input values of the all-rounder model, 4 additional input values are calculated by using the SplineMPE reconstruction as the event hypothesis:

- Expected charge[2]
- Distance of DOM to origin of Cherenkov light
- Residual time between actual pulse time and earliest possible time for Cherenkov light to reach the DOM
- Log-Likelihood given the event hypothesis

The labels which are reconstructed by the track uncertainty model are listed in tab. A.1. Details on the training of the neural network are given in appendix A.5 and the loss curves are plotted in fig. 7.3.

[2]Transformation (4.2) with $C = 1.1$ is performed before normalization

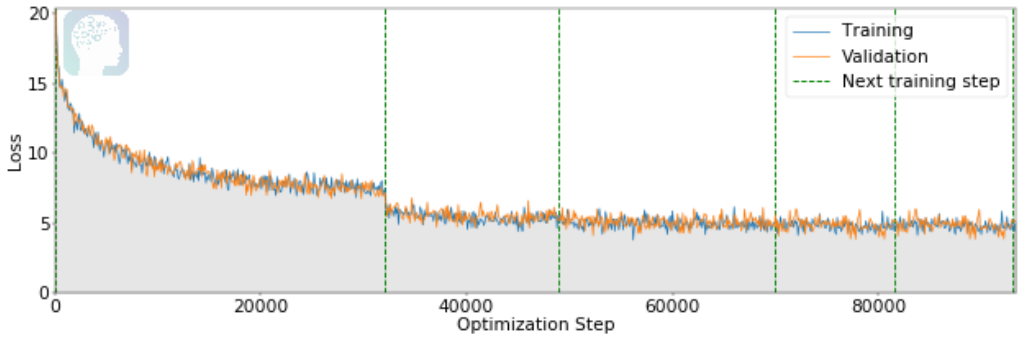


Figure 7.3: The loss curves of the validation and training dataset for the track uncertainty model are shown.

7.3 Fast Model

The fast model is a trimmed version of the all-rounder model. As a result, the runtime of the prediction is greatly reduced as stated in ch. 12.1. The number of filters, neurons per layer, and number of layers are reduced in comparison to the all-rounder model as shown in appendix A.6. In addition, the reconstructed labels are reduced to the muon energy at detector entry and its direction vector as noted in tab. A.1. The loss curves are plotted in fig. 7.4.

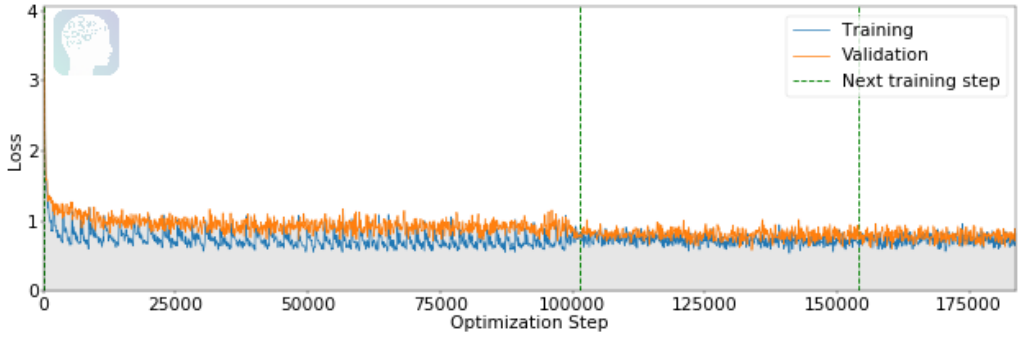


Figure 7.4: The training and validation loss curves are shown for the training of the fast model. A total of 185500 optimization steps are performed which corresponds to approximately 47 data epochs.

8 Reconstruction Performance Measures

There are many different measures to quantify the performance of a reconstruction method. While some measures are affected by bias or rely on a one-to-one relation between proxy variable and true label, others do not. It is therefore important to understand the limitations of the different performance measures.

8.1 Overall Performance Measures

The overall performance of an estimator can best be visualized in a scatter plot between the true label y_{true} and the proxy variable y_{pred} (correlation plot) as seen in the top left panel of fig. 8.1. For a good resolution, the distribution in label values

$$P(y_{\text{true}}|\hat{y}_{\text{pred}}) \tag{8.1}$$

for a given proxy value $\hat{y}_{\text{pred}}$ as well as the distribution of proxy values

$$P(y_{\text{pred}}|\hat{y}_{\text{true}}) \tag{8.2}$$

for a given label value $\hat{y}_{\text{true}}$ have to be narrow. This does not require a one-to-one relation between true label and proxy variable.

Under the requirement of a one-to-one relation between proxy variable and true label, the mean absolute error (MAE) and the root mean squared error (RMSE) are viable options to quantify the performance. While the RMSE is more sensitive to outliers, both the MAE and RMSE punish bias. Another measure that can be used is the width of the distribution of the residuals between y_{true} and y_{pred} expressed in the standard deviation. The standard deviation of the residuals is insensitive to a constant bias as shown in fig. 8.2b. This holds true for the pearson correlation coefficient as well. In addition, the pearson correlation coefficient is insensitive to the scaling between true label and proxy variable. It punishes deviations from a linear relation. To further loosen the requirement of a linear relation, the spearman correlation coefficient can be used. The spearman correlation demands a monotonic relation between true label and proxy variable to certify good resolution. However, this is not necessary for a good resolution.

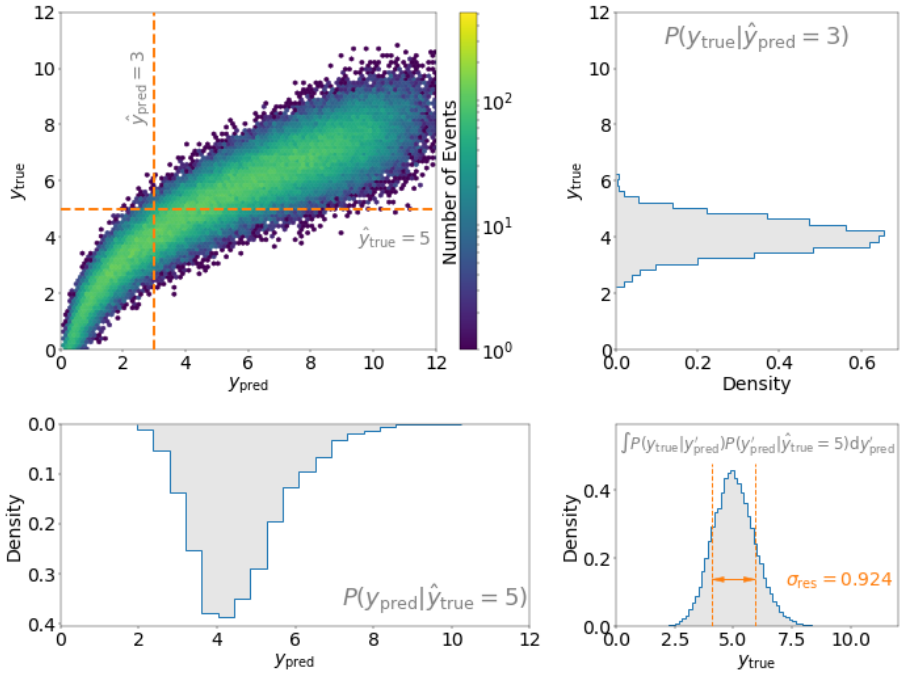
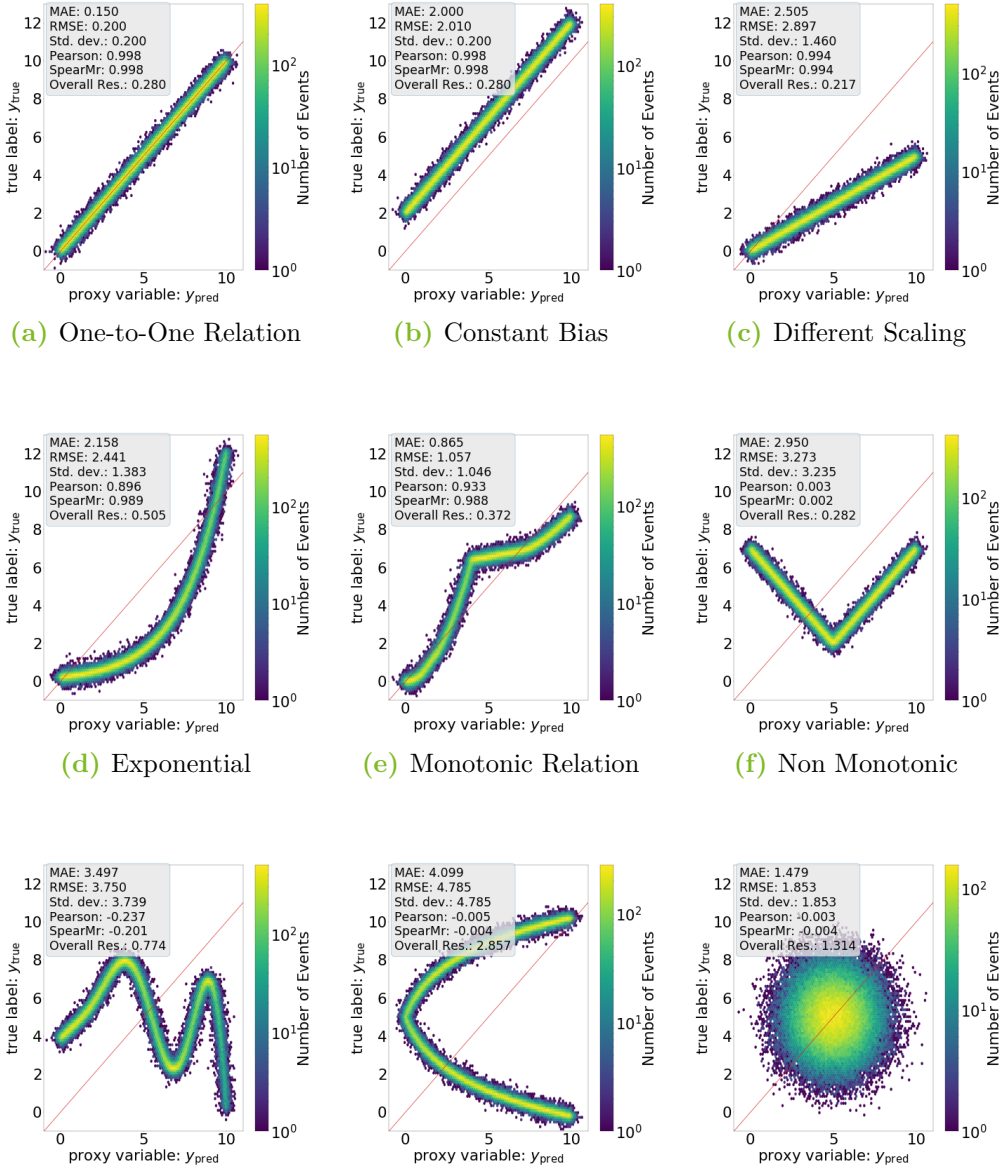


Figure 8.1: A correlation plot between true label y_{true} and proxy variable y_{pred} is shown in the top left panel. For a given proxy value $\hat{y}_{\text{pred}} = 3$, the distribution $P(y_{\text{true}}|\hat{y}_{\text{pred}} = 3)$ can be obtained as illustrated in the top right panel. Similarly, the distribution $P(y_{\text{pred}}|\hat{y}_{\text{true}} = 5)$ can be obtained (bottom left). The resolution σ_{res} of the proxy variable for a given true energy $\hat{y}_{\text{true}} = 5$ is obtained by the width of the distribution $\int P(y_{\text{true}}|y'_{\text{pred}})P(y'_{\text{pred}}|\hat{y}_{\text{true}} = 5)\,\mathrm{d}y'_{\text{pred}}$ as shown in the bottom right plot. (Adopted from [5, p.18])

The example shown in fig. 8.2f illustrates how all of the above measures show a poor resolution of the proxy variable, even though the label can be well reconstructed (a simple calibration can be performed to obtain a one-to-one relation). In order to obtain a more general statement of the performance of an estimator, another measure incorporating the width of the distributions in equations (8.1) and (8.2) is defined. The definition of the performance measure is given in the following chapter 8.2. An overview of the different performance measures and their limitations are illustrated in fig. 8.2.



(a) One-to-One Relation (b) Constant Bias (c) Different Scaling

(d) Exponential (e) Monotonic Relation (f) Non Monotonic

(g) Complex Relation (h) Ambigious Relation (i) Gaussian Noise ($\sigma = 1.3$)

Figure 8.2: The results of various performance measures are illustrated for different relations between the true label and proxy variable. The sensitivity towards bias, scaling, or monotony greatly depends on the given performance measure.

8.2 Energy dependent Resolution

The performance of the energy and directional reconstruction in IceCube is energy dependent. In general, the resolution is better for higher energetic events. It is therefore useful to calculate the resolution for a given energy.

A general way of obtaining an energy dependent resolution is to take all events within $\hat{y}_{\text{true}} \pm \Delta y$ for a given true energy $\hat{y}_{\text{true}}$. One of the previous mentioned performance measures such as the standard deviation of the residuals or the RMSE can then be used to obtain a resolution for the given energy $\hat{y}_{\text{true}}$. For the directional reconstruction, the median angle between reconstructed and true track is typically used.

As previously shown, events can be reconstructed with a high resolution, while some performance measures indicate otherwise. This could for instance be due to bias in the reconstruction. However, this can be corrected for through calibration. A more general performance measure is desired. One option is to include the width of the distributions in equations (8.1) and (8.2) as proposed in [5, p.18].

The resolution of a proxy variable y_{pred} for a given energy $\hat{y}_{\text{true}}$ can be obtained from the width of the distribution

$$\int_{y'_{\text{pred}}} P(y_{\text{true}}|y'_{\text{pred}}) P(y'_{\text{pred}}|\hat{y}_{\text{true}}) \, \mathrm{d}y'_{\text{pred}} \tag{8.3}$$

as illustrated in fig. 8.1 [5, p.18].

In order to evaluate equation (8.3), the integral and the distributions in equations (8.1) and (8.2) are discretized. $P(y'_{\text{pred}}|\hat{y}_{\text{true}})$ is a scalar value, while $P(y_{\text{true}}|y'_{\text{pred}})$ is a vector. The result of equation (8.3) is therefore a distribution in y_{true} as shown in the bottom right panel in fig. 8.1. The width of this distribution is the resolution of the proxy variable for a given energy $\hat{y}_{\text{true}}$. This has to be done for all energy bins $\hat{y}_{\text{true}} \pm \Delta y$.

The resolution for each energy bin $\hat{y}_{\text{true}} \pm \Delta y$ can then be used to obtain an overall measure of the resolution by calculating an average weighted by the number of events in each energy bin.

9 Energy Reconstruction

The energy reconstruction does not rely as heavily on timing information as the directional reconstruction. The chosen data representation with a 3D convolution and timing information summarized in the channel input is therefore a reasonable option. A total of 5 energy labels are reconstructed:

- Neutrino energy
- Energy deposited in detector
- Muon energy
- Muon energy at the point of entry into the detector
- Muon energy at the closest approach point

of which the muon energy at the point of entry into the detector is the most widely used in analyses. Unless stated otherwise, the all-rounder model as explained in ch. 7.1 is used to obtain the results shown in this chapter.

9.1 Muon Energy at Detector Entry

The muon energy at the point of entry in the detector is the most widely used energy proxy for the muon-neutrino energy. Analyses will therefore directly benefit from an improvement in the energy resolution.

The current standard energy reconstruction methods which are performed for the online L2 Muon Filter are Truncated Energy and MuEx. These are described in ch. 2.2.1. In the following, the results of the deep learning approach are evaluated and compared to the current standard.

The performance of an estimator can be measured in many different ways (see ch. 8). For a high resolution, a low standard deviation of the residuals and a high correlation between reconstruction and Monte-Carlo-Truth is expected, as well as a low mean absolute error (MAE). These values in addition to the overall resolution as defined in ch. 8.2 are calculated and included in the following correlation plots shown in fig. 9.2.

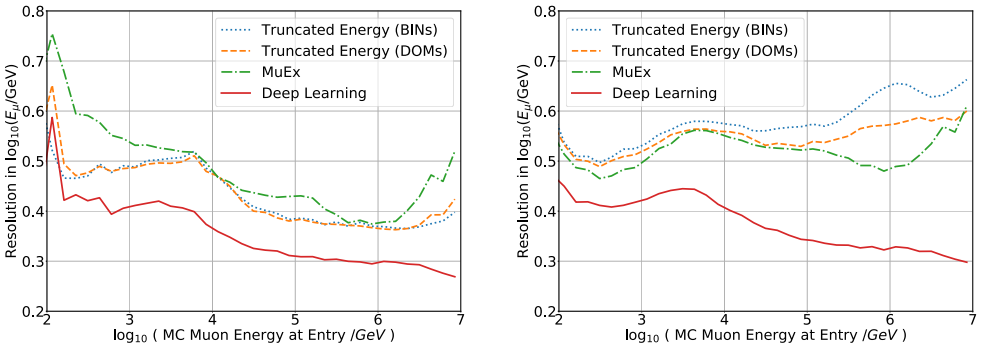


Figure 9.1: The resolution of the proxy variable is shown for a spectral index of $\gamma = -1.00$ and $\gamma = -2.19$, respectively.

In fig. 9.1, the energy dependent resolution is shown for two different spectral indices γ of an unbroken power-law flux E^γ. The resolution curve is calculated as described in ch. 8.2. The deep learning approach can greatly improve the resolution over all performance measures. The results are summarized in table 9.1.

	DOMs		**BINs**		**MuEx**	
Performance Measure	all	> 1 TeV	all	> 1 TeV	all	> 1 TeV
Std. dev.	32.4 %	25.8 %	33.6 %	28.0 %	33.9 %	11.2 %
MAE	36.8 %	23.5 %	35.8 %	22.6 %	39.3 %	40.2 %
Resolution	27.1 %	32.0 %	31.6 %	36.9 %	24.6 %	29.0 %
Pearson corr.	8.2 %	5.6 %	8.9 %	6.1 %	7.1 %	3.4 %
Spearman corr.	6.1 %	4.8 %	7.2 %	6.0 %	4.8 %	3.8 %

Table 9.1: The overall improvement is shown of the deep learning approach compared to the standard energy reconstructions for various performance measures. Charged-current events passing the Online Muon L2 filter are used to calculate the performance.

In comparison to the results presented in [31], the energy resolution is further improved by a few percent. This is accomplished through the combination of changes in the network architecture, of changes in the variables used as input, and through the utilization of residual additions (ch. 5.3) as well as variance maintaining layers (ch. 5.4).

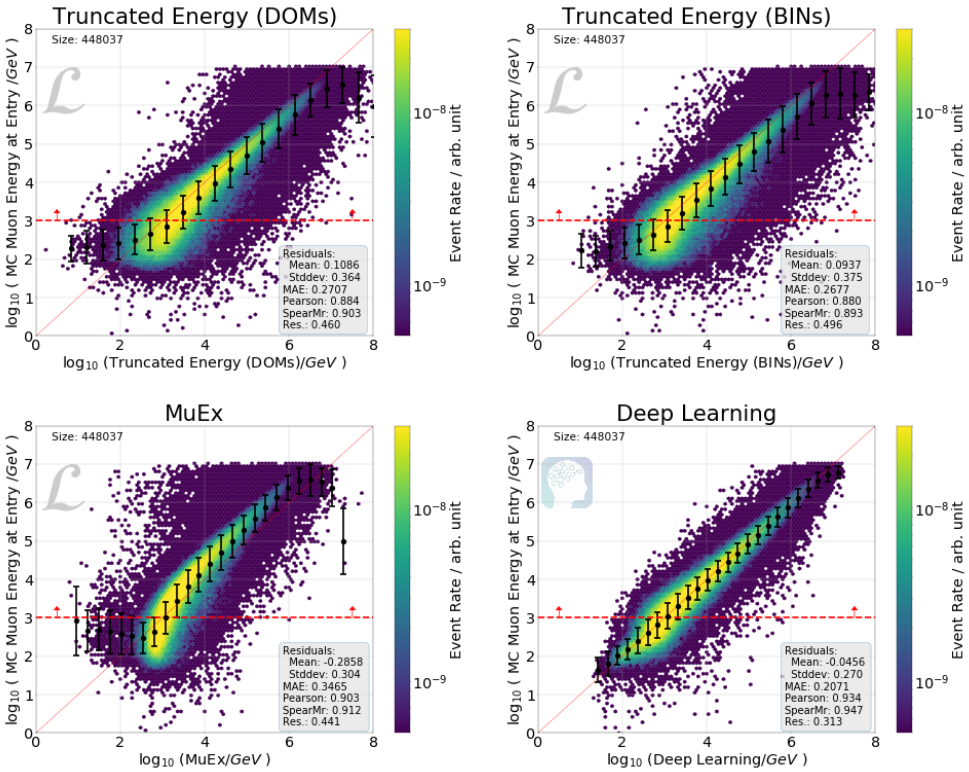


Figure 9.2: The correlation plots for different energy estimators are shown for charged current muon-neutrino events passing the Online Muon L2 filter. A spectral index of $\gamma = -2.19$ is used. Only events above 1 TeV indicated by the red dashed line are used to calculate the correlation and resolution. The deep learning approach is able to greatly improve the resolution across all performance measures.

9.1.1 Whole Energy Range

The standard energy reconstruction methods shown in the correlation plots in fig. 9.2 exploit the proportionality between the muon energy and its energy loss $\mathrm{d}E/\mathrm{d}x$ as described in ch. 2.2.1. This proportionality, however, is only valid for muon energies above approximately 1 TeV. Therefore, the correlation and resolution is calculated for events with muon energies above this threshold to obtain a fair comparison.

For lower energies, the energy loss flattens out and is more or less constant. In this case, the track length of the muon is a good indicator for its energy. This, however, only works well when most of the track is contained in the detector. Energy estimators for the low energy region exist within IceCube [6, 12], but are not included in the simulation dataset used for the study presented within this thesis. Additional low energy simulation datasets need to be added for a comparison. In any case, even if perfect energy estimators for different muon energy ranges exist, it is difficult to know when to use which estimator, since the true muon energy is not known. The deep learning approach does not have this limitation. It is capable of reconstructing the muon energy over the whole energy range of the simulation dataset.

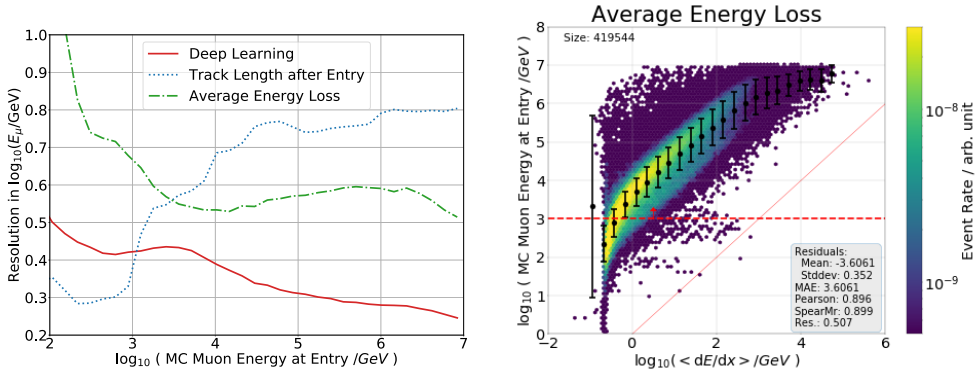


Figure 9.3: The resolution curves of the deep learning approach as well as of the true remaining muon track length after its entry into the detector and the average energy loss are shown on the left. A correlation plot between the muon energy and its average energy loss is depicted on the right. Above 1 TeV the energy loss $\mathrm{d}E/\mathrm{d}x$ becomes a better proxy for the muon energy in comparison to its track length. In this transition region, a bump in the resolution curve of the deep learning approach can be identified.

The transition region in which the energy loss becomes a better proxy for the muon energy compared to the track length is shown in fig. 9.3. At around $10^{3.3}\,\mathrm{GeV} \approx 2\,\mathrm{TeV}$ a bump is visible in the resolution curve of the deep learning approach. In

principle, both of the resolution curves for the muon track length and its energy loss should be lower than the curve of the deep learning approach, since they are calculated with the Monte Carlo truth. However, the energy loss plotted is only an average energy loss:

$$< \mathrm{d}E/\,\mathrm{d}x >= \frac{E_{\text{entry}} - E_{\text{center}}}{\Delta x}, \tag{9.1}$$

where Δx is the distance between the point of entry into the detector and the closest approach point of the muon to the center of the detector. In addition, the muon energy loss is a stochastic process. A simple average as applied in eq. (9.1) is therefore not necessarily the best choice. Truncated Energy, for instance, calculates a truncated mean over the energy losses in each bin. Nevertheless, it suffices to illustrate the transition region. A similar bump is seen in [30], where an energy reconstruction method based on a k-nn regression is investigated.

9.1.2 When Standard Reconstructions Fail

The standard reconstruction methods do not always succeed in reconstructing an event. This can for instance happen when the minimizer fails to find a minimum or when certain criteria are not fulfilled. For Truncated Energy, for example, a minimum number of bins is required. As a result, analyses which rely on these energy reconstructions will have to discard these events. With the help of the deep learning approach, these events do not have to be discarded.

The deep learning approach can reconstruct all events. Events previously discarded are typically events which are hard to reconstruct to begin with. However, with the uncertainty estimation as described in ch. 11 the neural network can pick well reconstructed events. In fig. 9.4 the correlation plot for all events where any one of the standard reconstructions failed is shown on the left. On the right, the subset selected by the neural network is shown. The subset chosen are the 20 % of events with the lowest predicted uncertainty. The threshold can be raised or lowered depending on the desired resolution.

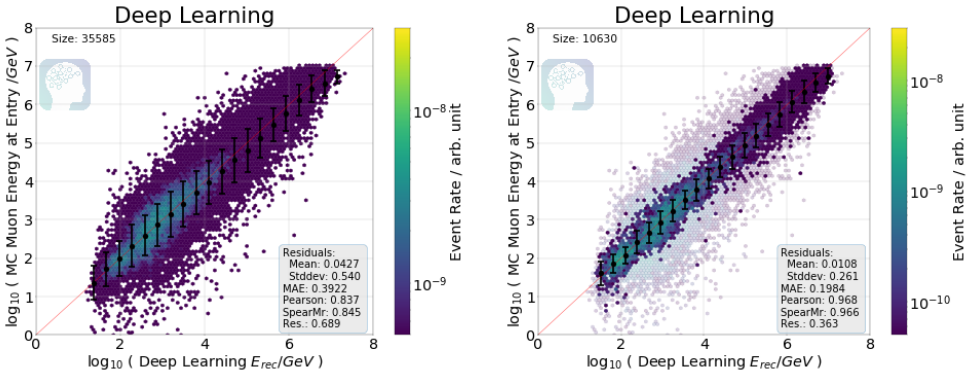


Figure 9.4: The deep learning approach is able to reconstruct all events. On the left, the correlation plot is shown for all charged current muon-neutrino events passing the Online Muon L2 filter where any one of the three standard energy reconstructions failed to produce an energy reconstruction. A spectral index of $\gamma = -2.19$ is used. A subset of 20 % of events is chosen for the correlation plot on the right. The events are chosen by the neural network by taking the events with the lowest predicted uncertainty (see ch. 11).

9.1.3 GFU Level

The GFU filter is an online filter applied after the Muon Online L2 Filter as described in ch. 2.1. It is used for online analyses and therefore relies on accurate reconstructions. In the following, the results of the deep learning approach are compared to the standard energy reconstructions performed for the GFU filter.

The deep learning approach is designed to work well for events and computational restrictions of the Online Muon L2 filter. However, it can improve the energy resolution of the GFU filter as well as seen in fig. 9.5 and fig. 9.6. The deep learning approach is able to improve the overall resolution by about 15 % to 20 %. If desired, the neural network can be fine tuned to the GFU filter through additional training. This is shown in appendix A.2.

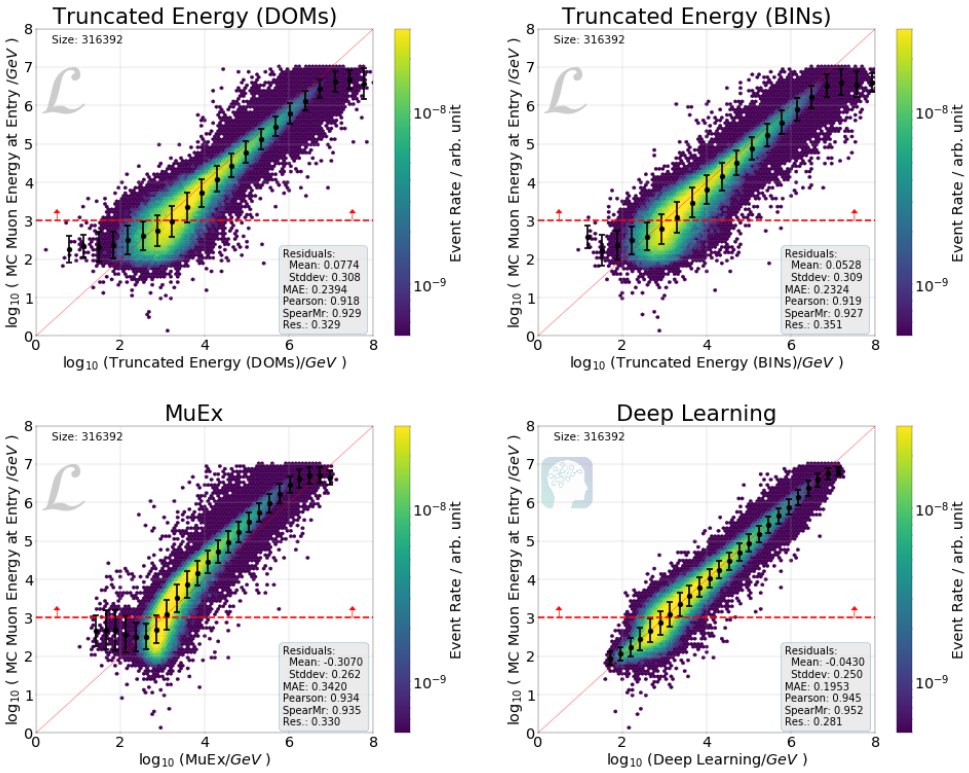


Figure 9.5: The correlation plots of the standard energy reconstructions as well as the deep learning approach are shown for events passing the GFU filter for a spectral index of $\gamma = -2.19$. Only charged-current muon-neutrino events with muon entry energies above 1 TeV indicated by the red dashed line are used to calculate the correlation and resolution. The deep learning approach is able to outperform the standard energy reconstructions.

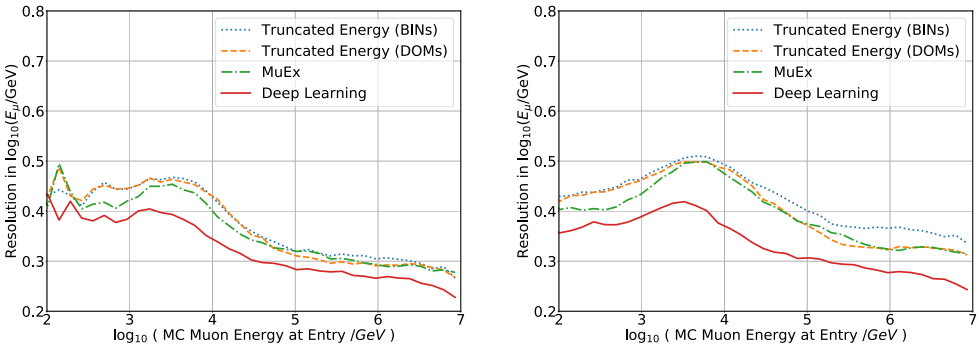


Figure 9.6: The resolution of the proxy variable is shown for a spectral index of $\gamma = -1.00$ and $\gamma = -2.19$, respectively, for events passing the GFU filter.

9.1.4 Final Data Level

The deep learning approach as presented in this thesis is designed to work on-site at the South Pole. Hence, the complexity of the network architecture is kept low to maintain a fast runtime. However, the deep learning approach is still able to improve the energy reconstruction for the final data level of point source analyses datasets. For the final data level, the standard energy reconstructions are rerun with improved track reconstructions and settings optimized for reconstruction accuracy instead of speed. The output of these reconstructions is then directly used in the analyses.

As shown in the correlation plots in fig. 9.7, the deep learning approach outperforms the current standard reconstructions even for the final data level. The resolution is better over all energy ranges as seen in fig. 9.8. Physics analyses which rely on the energy estimation of muons can therefore directly benefit from this improvement.

Moreover, the model can be fine tuned to the final data level through additional training to further increase the performance by about 5 %. The training steps and results are shown in appendix A.3.

The performance of the deep learning approach in comparison to the standard reconstruction methods for the various filters is summarized in table 9.2. An improvement is possible through the deep learning approach in all filters, while the biggest relative improvement is obtained in the Muon Online L2 Filter for which the model is designed. As shown in appendix A.4 and fig. 12.2, even further improvements can be obtained through the use of robust loss functions such as the tukey loss [15](ch. 6).

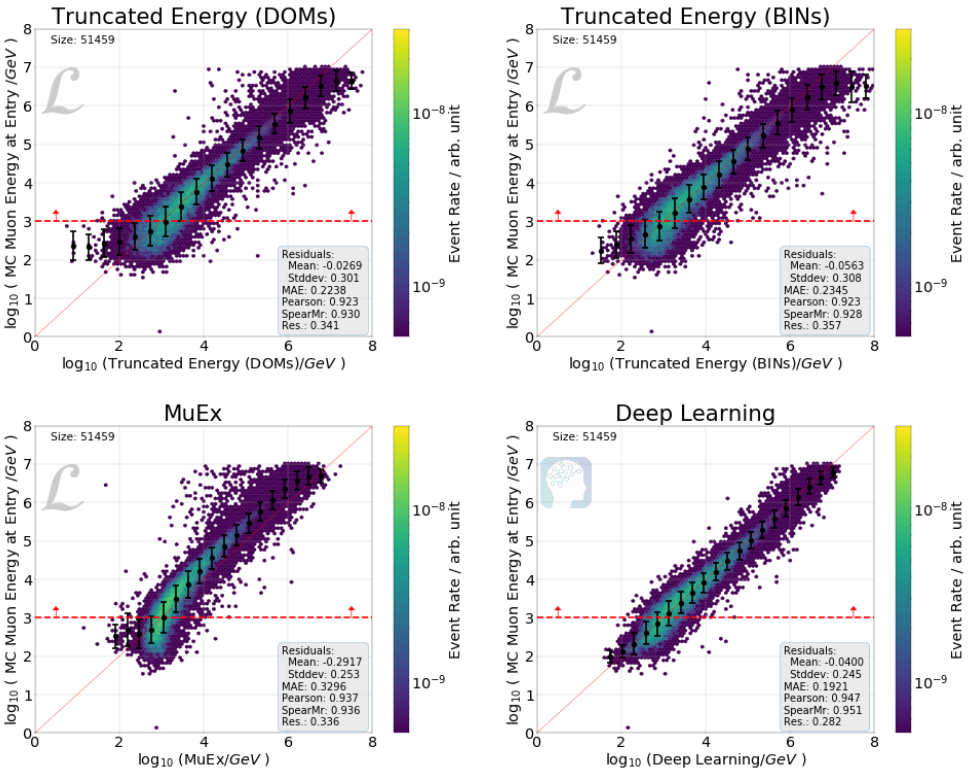


Figure 9.7: Depicted are the correlation plots of the current standard energy reconstructions for the point source final data level. A spectral index of $\gamma = -2.19$ is used and only charged-current events with muon energies above 1 TeV indicated by the red dashed line are used to calculate the correlation and resolution. Even though the deep learning approach is designed to comply the limitations given on-site at the South Pole, it is still able to improve the resolution for the final data level.

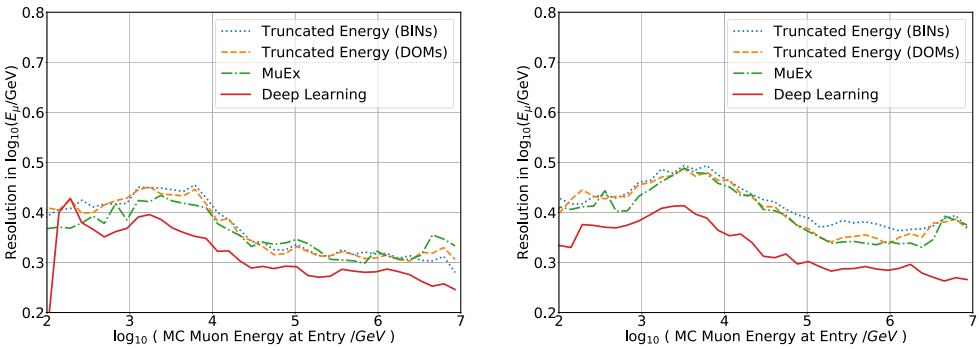


Figure 9.8: The resolution of the proxy variable is shown for a spectral index of $\gamma = -1.00$ and $\gamma = -2.19$, respectively, for events of the final point source data level.

Filter	DOMs	BINs	MuEx
Online L2 $> 1\,\mathrm{TeV}$	32.0 %	36.9 %	29.0 %
Online L2	27.1 %	31.6 %	24.6 %
GFU $> 1\,\mathrm{TeV}$	14.6 %	19.9 %	14.8 %
GFU	19.0 %	21.6 %	18.0 %
PS final $> 1\,\mathrm{TeV}$	17.3 %	21.0 %	16.1 %
PS final	19.9 %	21.5 %	18.0 %

Table 9.2: The overall improvement in resolution for different event filters is shown. With the help of the deep learning approach, the resolution can be greatly improved.

9.2 Other Energy Labels

In addition to the muon energy at its point of entry into the detector, other energy related labels are reconstructed. This includes the muon energy at the point of closest approach to the center of the detector as shown in fig. 9.9.

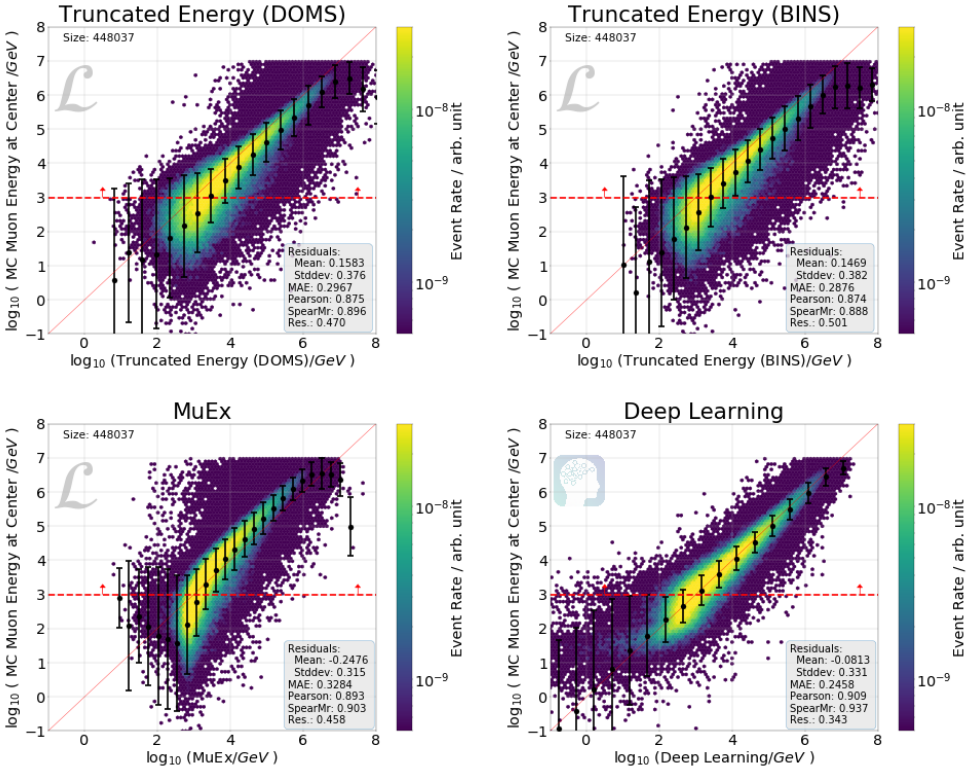


Figure 9.9: The correlation between reconstructed and true muon energy at the closest approach to the center of the detector is shown. The deep learning approach is able to improve the resolution in comparison to the standard energy reconstruction methods. A spectral index of $\gamma = -2.19$ is used. Only events above 1 TeV indicated by the red dashed line are used to calculate the correlation and resolution.

The correlation plots for the energy deposited in the detector by the neutrino as well as the muon energy can be seen in fig. 9.10.

The reconstruction of neutrino energy is an inherently difficult task. For through-going muons, only the energy of the muon at its point of entry into the detector can

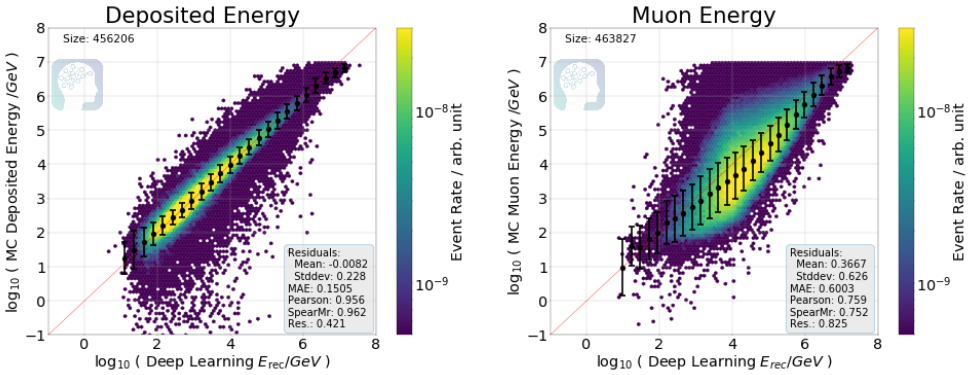


Figure 9.10: On the left, the correlation plot is shown for the energy deposited in the detector by the neutrino. The correlation plot for the muon energy is shown on the right. Events passing the Online Muon L2 filter with a spectral index of $\gamma = -2.19$ are used.

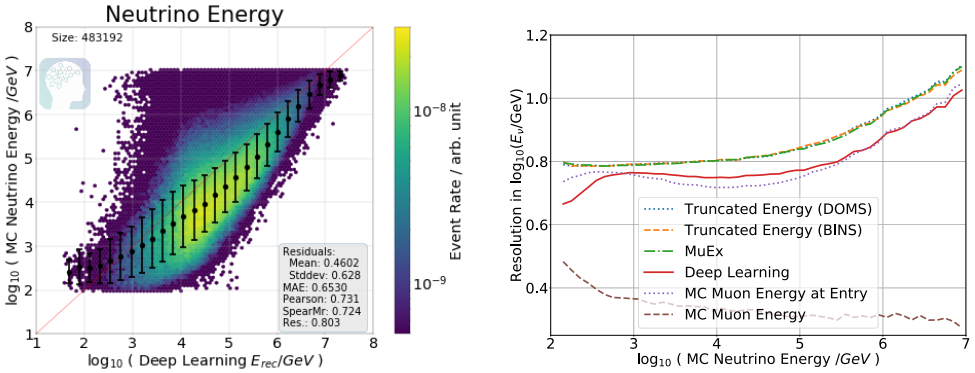


Figure 9.11: The correlation between true neutrino energy and the result of the deep learning approach is illustrated on the left for events passing the Online Muon L2 filter. A spectral index of $\gamma = -2.19$ is used. Reconstructing the neutrino energy is an inherently difficult task as shown in the resolution plot on the right. For through going tracks, the best achievable energy resolution while disregarding earth attenuation and zenith dependent effects is given by the true muon energy at the point of entry (MC Muon Energy at Entry).

be measured. In theory, the true muon energy would provide an even better proxy for the neutrino energy, however, it is not possible to determine how much energy the muon has lost before its entry into the detector.

The resolution on the reconstruction of the neutrino energy is therefore limited by the muon energy at its point of entry. This can be seen on the right of fig. 9.11. The resolution of the deep learning approach is very close to the limit given by the true Monte Carlo energy of the muon at its point of entry. For starting tracks, the resolution can be reduced by including the energy of the initial cascade. In addition, taking the zenith dependence of the neutrino spectrum as well as earth attenuation effects into account can further improve the resolution. At lower energies, the deep learning approach seems to have a better resolution than the theoretical limit. Starting events might contribute to this, but the dominating effect is most likely the boundary of the Monte Carlo dataset. The simulation dataset stops at 10^2GeV, introducing an artificial cutoff.

9.3 Conclusion and Outlook

The energy reconstruction by the deep learning-based approach significantly surpasses the resolution of the current standard methods in all filters and data levels. The relative improvement varies from $\approx 15 - 35\%$ depending on the event selection and data level. Even though the deep learning approach is intended for the on-site reconstruction, it outperforms even the energy reconstructions applied offline for the final data level. The developed reconstruction method is currently the best available method in IceCube for the reconstruction of the muon and neutrino energy of charged-current muon-neutrinos.

In addition, the deep learning-based method is able to reconstruct the energy of all events and over the whole energy range covered by the training data. The standard energy reconstructions are only valid in a certain energy regime.

As shown for the energy resolution of the neutrino energy in fig. 9.11, the deep learning-based method provides a resolution which is close to the theoretical lower limit. Taking all effects into account, an exact theoretical limit can be obtained. In future work, this limit can be used to evaluate possible further improvements of the deep learning-based reconstruction.

10 Directional Reconstruction

Directional reconstruction is performed with the all-rounder model described in ch. 7.1. Focus is laid on the directional reconstruction of the muon, which is the best possible proxy for the neutrino direction. Three possible approaches to reconstruct the muon direction are described in the following.

Direct Reconstruction of Azimuth and Zenith

A naive approach to reconstruct the azimuth and zenith angle is to directly use these as labels. This, however, is problematic in practice. The azimuth angle is poorly defined for very high or very low zenith angles. In addition, the 2π-periodicity needs to be accounted for. A proper definition of the gradient is therefore difficult, which is needed for gradient descent during training.

Reconstruction of the Direction Vector

Instead of directly reconstructing the zenith and azimuth angle, the three coordinates of the direction vector are reconstructed. This resolves the previously mentioned difficulties. In principle, the norm of the direction vector could be constrained to 1 during training. However, this is not performed in the results shown here.

Reconstruction over Entry and Exit Point

A third example of how the direction can be reconstructed is by using the entry and exit point of the muon into and out of the detector. Once these points are reconstructed, the direction vector $\vec{d}$ is calculated as

$$\vec{d} = \frac{\overrightarrow{EX}}{|\overrightarrow{EX}|}, \tag{10.1}$$

where E defines the point of entry and X the exit point of the muon. In the case of a starting muon, the entry point E is defined as the vertex. Similarly, for a stopping muon, the exit point X is defined as the stopping point.

10.1 Results

All three of the above mentioned methods are used to reconstruct the direction of the muon. The results of these methods are compared to the standard directional reconstruction SplineMPE (ch. 2.2.2). In fig. 10.1 the results are shown for the azimuth angle. The correlation plots for the zenith angle are plotted in fig. 10.2.

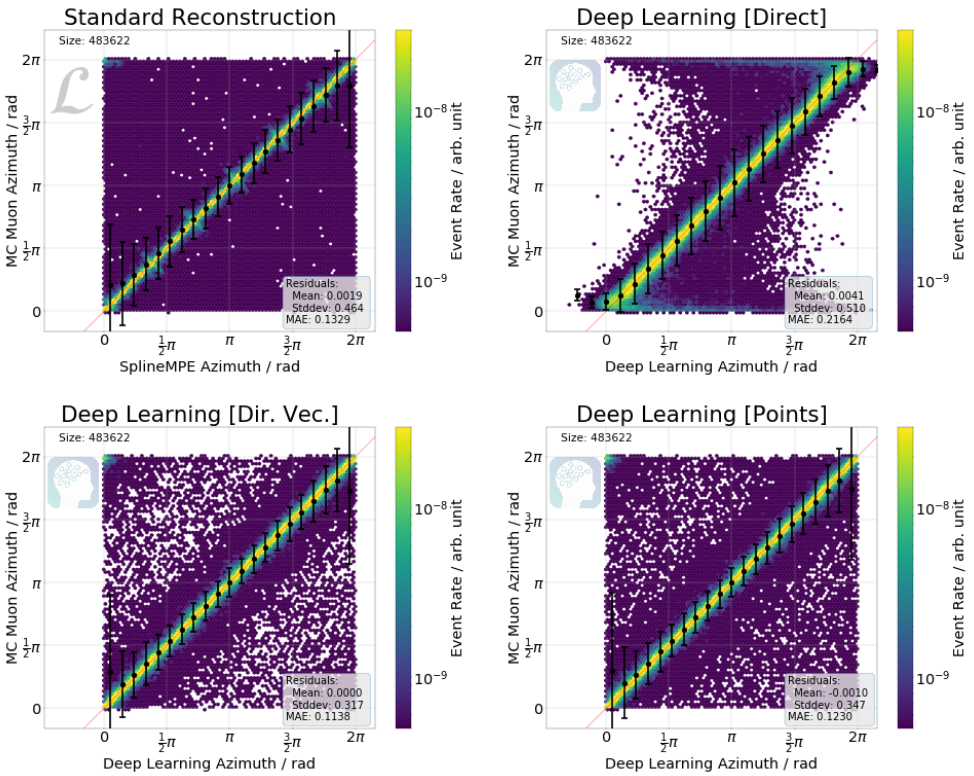


Figure 10.1: The correlation plots of the various reconstruction methods for the muon azimuth angle are shown for charged current muon-neutrino events passing the Online Muon L2 filter. A spectral index of $\gamma = -2.19$ is used. For the calculation of the residual values and MAE, the 2π-periodicity is taken into account. Artifacts of the hexagonal shape of the IceCube detector can be found in the SplineMPE azimuth correlation plot.

The deep learning approach is able to improve the average angular resolution. However, when only looking at the median angular resolution, the standard likelihood method SplineMPE outperforms the deep learning approach. With the deep learning approach, events which are badly mis-reconstructed by the standard reconstruction

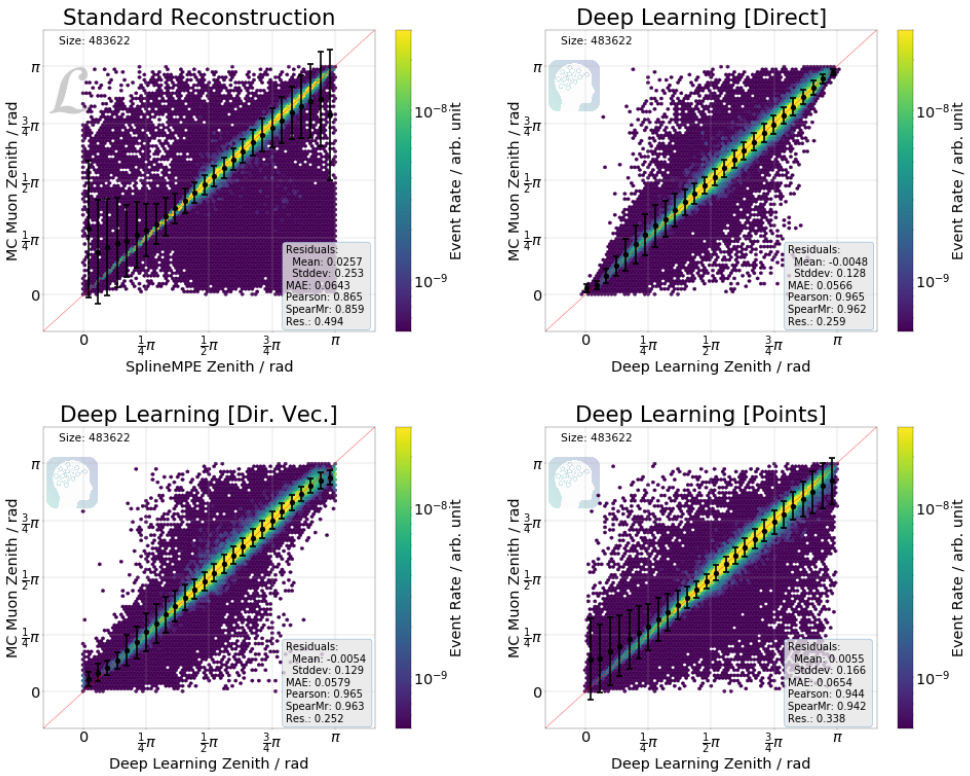


Figure 10.2: Shown are the correlation plots for the muon zenith angle for the different reconstruction methods. Charged current muon-neutrino events passing the Online Muon L2 filter are used with a spectral index of $\gamma = -2.19$.

can be greatly reduced at the cost of loosing resolution for the well reconstructed events. This might be an effect due to the chosen loss function (MSE) which heavily punishes outliers as described in ch. 6. An attempt to reduce the weight of outliers by utilizing a more robust loss function such as the tukey loss[15] yields an improvement as shown in fig.10.4 in comparison to fig. 10.3. Fine-tuning with the tukey loss improves the median angular resolution by about 1°. The energy resolution also improves by about 5 % in comparison to the results shown in ch. 9. In fig. 12.2, the resolution of different deep learning models is compared for the muon energy at entry and the directional reconstruction. The results of the tukey fine-tuned all-rounder model are summarized in appendix A.4.

The SplineMPE reconstruction is a likelihood based reconstruction as mentioned in ch. 2.2.2. An infinite track of a minimum ionizing muon is assumed for the track hypothesis. In the cases where this is a valid assumption, the likelihood minimization will result in the best possible reconstruction, assuming the global minimum is found. However, the likelihood will provide a poor reconstruction for events where this is not the case. This effect is exceptionally well visible in the azimuth correlation plot on the top left of fig. 10.1. While most events are well reconstructed and distributed along the diagonal of the correlation plot, others are badly mis-reconstructed and evenly distributed in azimuth angles. For these mis-reconstructed events, a random number generator would provide a similar reconstruction accuracy.

In addition, the likelihood exhibits difficulties to describe high energy events as seen in fig. 10.3 for the SplineMPE reconstruction. The brighter an event is (the higher energetic a muon is), the more information is available to reconstruct the event. Hence, the angular resolution should improve with increasing energy. Nonetheless, the median angular resolution of the SplineMPE reconstruction seems to show a resolution floor and the average resolution on the right of fig. 10.3 even greatly increases. Studies are currently underway within IceCube to investigate this issue.

While likelihood-based methods provide the best possible reconstruction in theory, they are often difficult to parameterize or typically limited in complexity and dimensionality due to practical reasons such as minimization difficulties. In contrast, a deep learning-based approach can in principle directly learn a likelihood in a high-dimensional feature space from the given data.

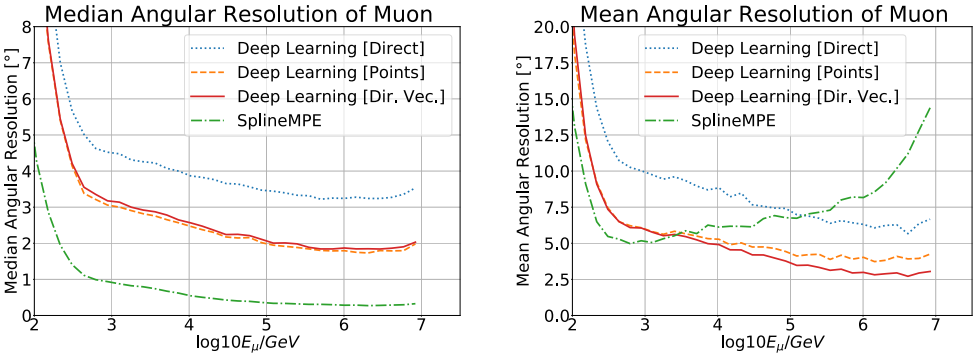


Figure 10.3: The angular resolution of the various reconstruction methods are shown for a spectral index of $\gamma = -2.19$. On the left, the median angular resolution is shown, while the average angular resolution is shown on the right. The deep learning approach can reduce badly mis-reconstructed events.

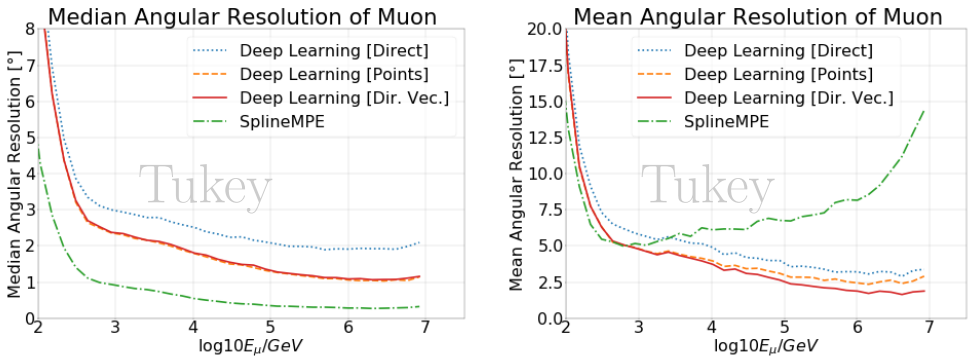


Figure 10.4: The median and mean angular resolution is shown for the standard reconstruction and the deep learning approach fine-tuned with a robust tukey loss. A spectral index of $\gamma = -2.19$ is used. Reducing the effect of outliers in the loss function by applying a robust tukey loss improves the overall resolution.

10.2 Reduction of Outliers

Many analyses in IceCube searching for neutrinos use the earth as a shield to eliminate the atmospheric muon background. It is nearly impossible for an atmospheric

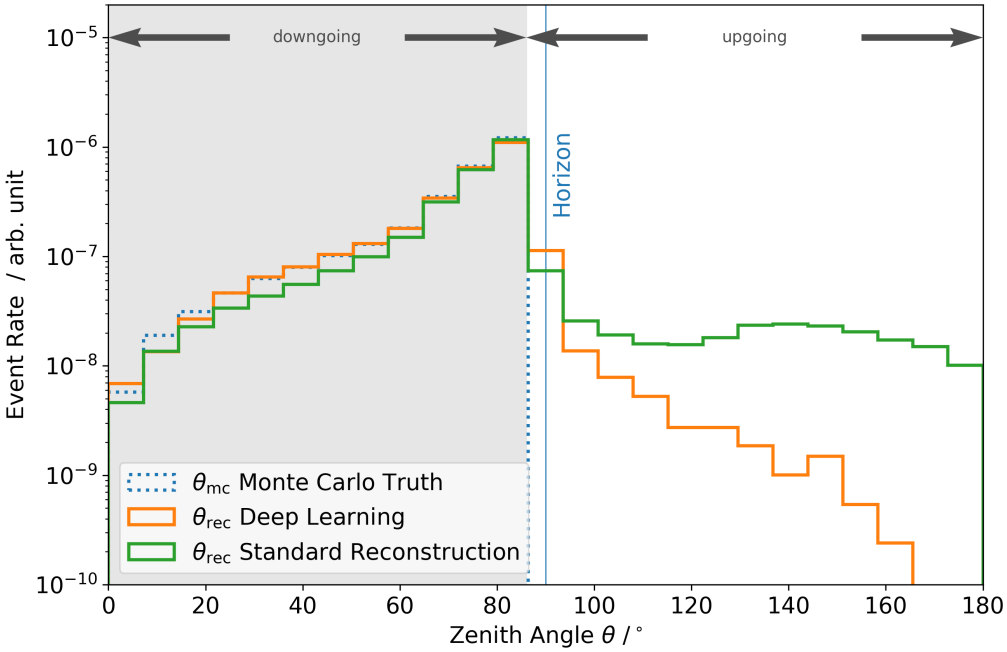


Figure 10.5: The distribution of background events (down going muons) is shown for the true zenith angle as well as the zenith angles reconstructed by the standard reconstruction and the deep learning approach. A cut in the zenith angle reconstructed by the deep learning approach is more effective in reducing the background than a cut in the zenith angle reconstructed by the standard reconstruction.

muon to penetrate the earth. Most of the atmospheric background can therefore be eliminated by discarding all downgoing events. In practice, this is done by applying a cut in the reconstructed zenith angle at about 86° discarding all events with zenith angles below this threshold. In principle, this should discard all background events. However, muons are occasionally mis-reconstructed, so that down-going muons are falsely classified as up-going. This poses a key challenge in obtaining pure data samples.

The standard directional reconstruction, SplineMPE, is very accurate in most cases,

but it has a high number of outliers as shown in the correlation plot on the top left of fig. 10.2. With the help of the deep learning approach, these outliers can be greatly reduced. As a result, purer data samples with higher efficiency can be obtained.

To illustrate this, the charged-current events of the neutrino simulation dataset are separated into up and down going muons according to the true simulated zenith angle. The down going muons are then assumed to be background. This is an approximation for atmospheric muons. For a more proper investigation, this study has to be performed on corsika simulation datasets. The distribution of the down going muons (background) is plotted in fig. 10.5 for the true zenith angle and the reconstructed zenith angles for the standard method and the deep learning approach.

A cut in the true zenith angle at 86° discards all down going muons per definition. For the reconstructed zenith angles, this is not the case. The background distribution in the standard zenith reconstruction is rather flat with increasing zenith angle, while the distribution in the zenith angle reconstructed by the deep learning approach falls off sharply. A cut in the deep learning reconstructed zenith angle is therefore more effective at reducing the background.

10.3 Conclusion and Outlook

In comparison to the standard directional reconstruction, the deep learning-based approach can significantly reduce the number of mis-reconstructed events. This is of great importance for event selections as shown in fig. 10.5. With the help of the deep learning approach, purer data samples with higher efficiency can be generated.

The median resolution of the deep learning approach cannot yet compete with the resolution of the standard reconstruction. However, there is still room for improvement. The directional reconstruction heavily relies on the timing information which the current network architecture might not optimally use. As mentioned in ch. 4.3, many different options exist which remain to be investigated. Simple adjustments such as the fine-tuning with the tukey loss result in a significant jump in the resolution curve as shown in fig. 10.3 and 10.4. This indicates that solutions exist, which can further improve the deep learning-based method.

Another option is to combine the strengths of the standard and deep learning-based reconstruction. An additional machine learning method such as a random forest or BDT can be used on top of the results produced by the standard and deep learning-based approach. Especially in regard to the uncertainty estimate (ch. 11) available

on every reconstructed quantity, the machine learning method should be able to interpolate between the best reconstructions to further improve the resolution.

Moreover, the standard reconstructions often get stuck in a local minimum of the likelihood landscape. Seeding with an improved track hypothesis, facilitates convergence to the global minimum. The track hypothesis provided by the deep learning approach is well suited as a seed due to the low number of outliers. As an example, the HESE alert[52] sent out on October 28$^{\text{th}}$, 2017, provided a very poor initial track reconstruction. Follow-up millipede[59] scans reconstructed a track which is over 130° away from the online SplineMPE track reconstruction. The alert was later retracted due to high directional uncertainty. When seeded with the deep learning track hypothesis (which is 3.5° away from the millipede result), the online reconstruction reconstructs a track which is 5.1° away from the millipede result. Assuming that the millipede result is close to the true direction, this is quite an improvement. However, to obtain quantitative results, a thorough study needs to be performed.

11 Uncertainty Estimation

In order to obtain an uncertainty estimation on the reconstructed quantities, a small fully connected network is added to the main network as illustrated in fig. 7.1. The network architecture used is the all-rounder model described in ch. 7.1. The smaller network obtains the flattened layer of the main network as an input with a gradient stop applied. Only with the gradient stop can the training of the smaller network be performed independently of the main network. In the following, this smaller fully connected network is referred to as the uncertainty layers. Two different approaches to train the uncertainty layers are described below.

Mean Squared Error

The uncertainty layers can be trained over the mean squared error as described in ch. 6 with a little modification in eq. (6.1). Instead of the residuals between the true and predicted values $|Y_{\text{true}} - Y_{\text{pred}}|$, the residuals

$$\Delta Y = |\sigma_{\text{pred}} - |Y_{\text{true}} - Y_{\text{pred}}|| \tag{11.1}$$

are used, where σ_{pred} are the uncertainties predicted by the neural network. In doing so, the network is trained to correctly estimate the residual $|Y_{\text{true}} - Y_{\text{pred}}|$ on average. This is not identical to a 1-sigma confidence interval and needs to be taken into account when interpreting the uncertainties estimated by the neural network.

Gaussian Likelihood

An alternative approach is to assume a Gaussian Likelihood. Under the assumption that the reconstructed value y_{pred} is normally distributed around the true value y_{true}, the probability density can be defined as

$$f(y_{\text{pred}}) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{1}{2}\frac{(y_{\text{pred}} - y_{\text{true}})^2}{\sigma^2}\right) \tag{11.2}$$

where σ is the uncertainty on the reconstruction of y_{true}.

For a batch of n events the Gaussian likelihood can then be written as

$$\begin{aligned}\mathcal{L} &= \prod_{i=1}^{n} f(y_{\text{pred}_i}) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \cdot \exp\left(-\frac{1}{2}\frac{(y_{\text{pred}_i} - y_{\text{true}_i})^2}{\sigma_i^2}\right).\end{aligned} \tag{11.3}$$

Maximizing the likelihood in eq. (11.3) is equivalent to minimizing the negative log likelihood which is used as the loss function

$$\text{loss} = \sum_{i=1}^{n}\left(\ln\left(\sigma^2_{\text{pred}_i}\right) + \frac{(y_{\text{pred}_i} - y_{\text{true}_i})^2}{\sigma^2_{\text{pred}_i}}\right) \tag{11.4}$$

during traing.

Equation (11.4) defines the loss function for a single label. To incorporate all labels as well as their weights according to the importance vector $\vec{C}$ as described in ch. 6 the loss function is modified to

$$\text{loss} = \sum_{l=1}^{L}\left(\vec{C}\right)_l \cdot \frac{1}{n}\sum_{i=1}^{n}\left(\ln\left((\sigma^l_{\text{pred}_i})^2\right) + \frac{(y^l_{\text{pred}_i} - y^l_{\text{true}_i})^2}{(\sigma^l_{\text{pred}_i})^2}\right) \tag{11.5}$$

where $l = 1, \ldots, L$ are the labels.

11.1 Results

To validate the uncertainty estimation by the neural network, the pull distributions for both approaches for the muon energy at detector entry are shown on the left of fig. 11.1. On the right, the resolution measured in different quantities is shown for events with a given estimated uncertainty. The top panel shows the results for the mean squared error approach and the bottom panel for the Gaussian Likelihood.

Assuming Gaussian distributed residuals, a perfect uncertainty estimator is expected to produce a Gaussian shaped pull distribution with a variance of 1[18]. As previously mentioned, the uncertainty estimation trained with a mean squared error loss function is expected to correctly estimate the residual between true and reconstructed value on average. This can also be seen on the top right of fig. 11.1. The uncertainty estimated by the neural network is highly correlated to the resolution. A one to one relation (solid red line) is observed between the estimated uncertainty and the mean absolute residuals for the mean squared error approach.

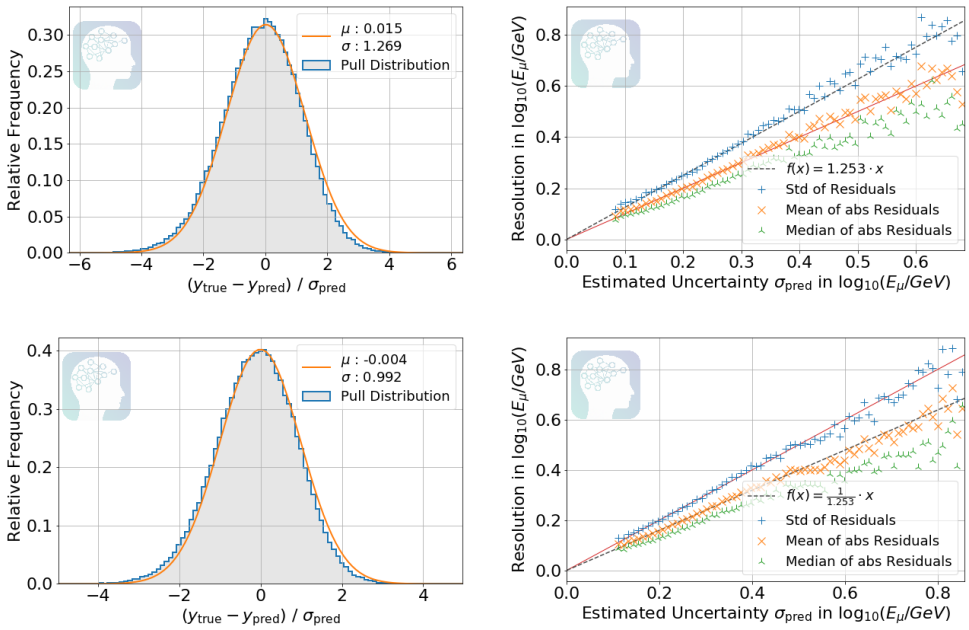


Figure 11.1: The pull distributions for the muon energy at detector entry are shown on the left for the mean squared error (top) and the Gaussian Likelihood approach (bottom). On the right, the resolution of the reconstruction for events with a given estimated uncertainty is shown for the two methods.

Assuming that the residuals $x = y_{\text{true}} - y_{\text{pred}}$ are normally distributed with a mean of zero:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{x^2}{2\sigma^2}\right), \tag{11.6}$$

the mean absolute residual is given by

$$\langle x_{\text{abs}} \rangle = 2 \int_0^\infty x \cdot f(x)\, \mathrm{d}x = \frac{2\sigma}{\sqrt{2\pi}}. \tag{11.7}$$

The 1-sigma confidence interval can therefore be obtained by the following relation:

$$\sigma = \sqrt{\frac{\pi}{2}} \cdot \langle x_{\text{abs}} \rangle \approx 1.253 \cdot \langle x_{\text{abs}} \rangle. \tag{11.8}$$

The uncertainty prediction through the mean squared error approach is therefore expected to underestimate the uncertainty by a factor of about 1.253. For comparison, a dashed line $f(x) = 1.253 \cdot x$ for the corrected uncertainty prediction value is shown. Similarly, a line $f(x) = \frac{1}{1.253} \cdot x$ is added in the resolution plot for the Gaussian Likelihood approach, showing that the results of these methods only differ by a constant factor (for Gaussian distributed residuals).

Training the uncertainty layers with the Gaussian Likelihood approach directly results in correctly estimated uncertainties as shown in fig. 11.1. In contrast to the mean squared error approach, the estimated uncertainties do not need to be corrected.

11.2 Selection of well Reconstructed Events

Another qualitative way of validating the uncertainty estimation is by choosing a subset of events with a low estimated uncertainty. If the uncertainty estimation produces reasonable results, this subset should consist of events, which are well reconstructed. In fig. 11.2, a subset of 10 % of all events with the lowest estimated uncertainty for the neutrino energy and the muon zenith angle is shown. The original sample is shown in the background. Using the estimated uncertainty to select well reconstructed events can significantly boost the resolution. Especially in regard to the neutrino energy, the achieved resolution is quite remarkable.

With the uncertainty estimation performed by the neural network, a subset of well reconstructed events can be chosen. This is also used in ch. 9.1.2 to recover events which previously would have been discarded due to unsuccessful reconstructions of the standard methods.

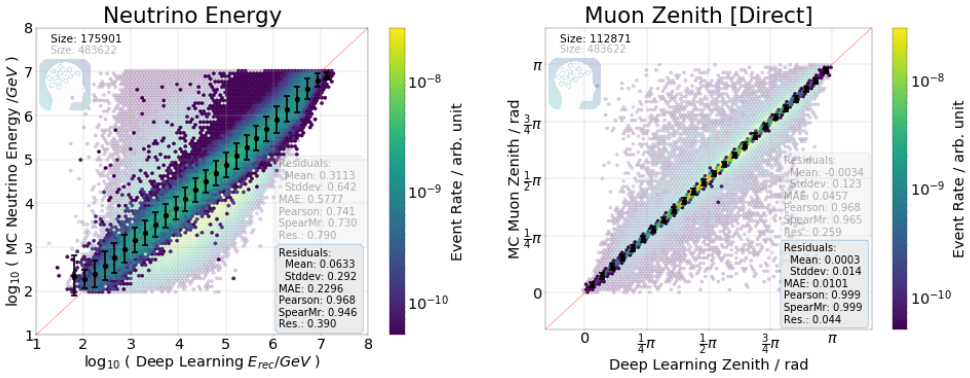


Figure 11.2: With the help of the uncertainty estimation performed by the neural network, a subset of well reconstructed events can be chosen. Shown are the 10 % of events with the lowest estimated uncertainty for the neutrino energy and the muon zenith angle. The tukey fine-tuned all-rounder model is used and the original sample is indicated in the background.

11.3 Track Uncertainty Estimation

The uncertainty estimation as described in this chapter can also be used to obtain an estimate on the uncertainty of the standard directional reconstruction. The results obtained here use the track uncertainty model as described in ch. 7.2. Uncertainty estimators for the standard directional reconstructions exist, but they are heavily biased and need to be corrected. One fast method is Cramér Rao as described in ch. 2.2.3. Better methods exists, however, their runtime is prohibitively long, so that they can only be run on a selection of a few events. The pull distributions for the (uncorrected) Cramér Rao method as well as for the deep learning approach are compared in fig. 11.3.

Under the assumption that the residuals between true and reconstructed value for the azimuth and zenith angle are normally distributed, uncorrelated and have a zero mean with equal variance, a Rayleigh distribution:

$$f_{\mathrm{r}}(x;\sigma_{\mathrm{r}}) = \frac{x}{\sigma_{\mathrm{r}}^2} \cdot \exp \frac{-x^2}{2\sigma_{\mathrm{r}}^2} \tag{11.9}$$

is expected for the overall track uncertainty. With these assumptions, the pull distribution for a perfect uncertainty estimator should therefore follow a Rayleigh distribution with $\sigma_{\mathrm{r}} = 1.0$.

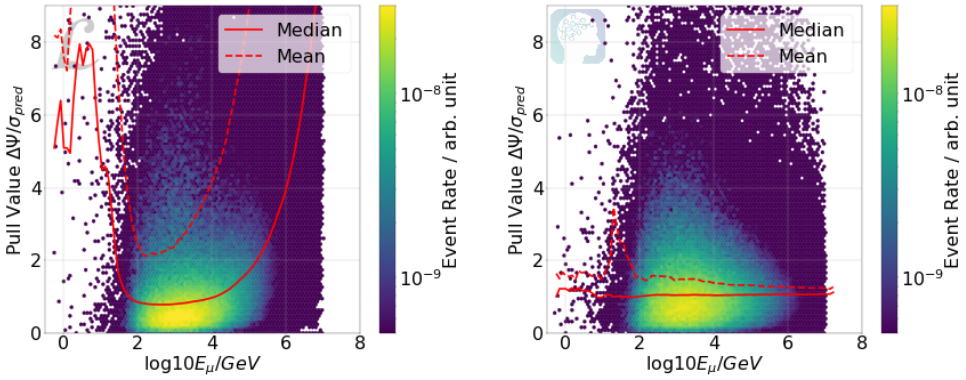


Figure 11.3: Shown are the energy dependent pull distributions for the Cramér Rao method (left) and for the deep learning appraoch (right). In contrast to the deep learning approach, the uncertainty estimate by the Cramér Rao method is heavily biased.

In practice, however, these assumptions do not hold. The standard directional reconstruction has a lot of mis-reconstructed events, which do not follow a normal distribution. In addition, the uncertainty on the zenith and azimuth angle is correlated. Nevertheless, a comparison between the theoretically expected Rayleigh distribution and the pull distribution for the Cramér Rao method and the deep learning approach is shown in fig. 11.4. The results of the deep learning approach (on the right) tend to follow the expected Rayleigh distribution more closely.

As performed in ch. 11.2, the uncertainty estimation can be qualitatively checked by choosing well reconstructed events based on the estimated uncertainty. In fig. 11.5 the angular resolution is shown for differently sized subsamples of the charged current events passing the Muon Online L2 Filter. The subsamples are chosen according to the estimated uncertainty on the standard directional reconstruction: SplineMPE. On the left, the selection is based on the uncertainty estimation of the (uncorrected) Cramér Rao method and on the right it is based on the deep learning approach.

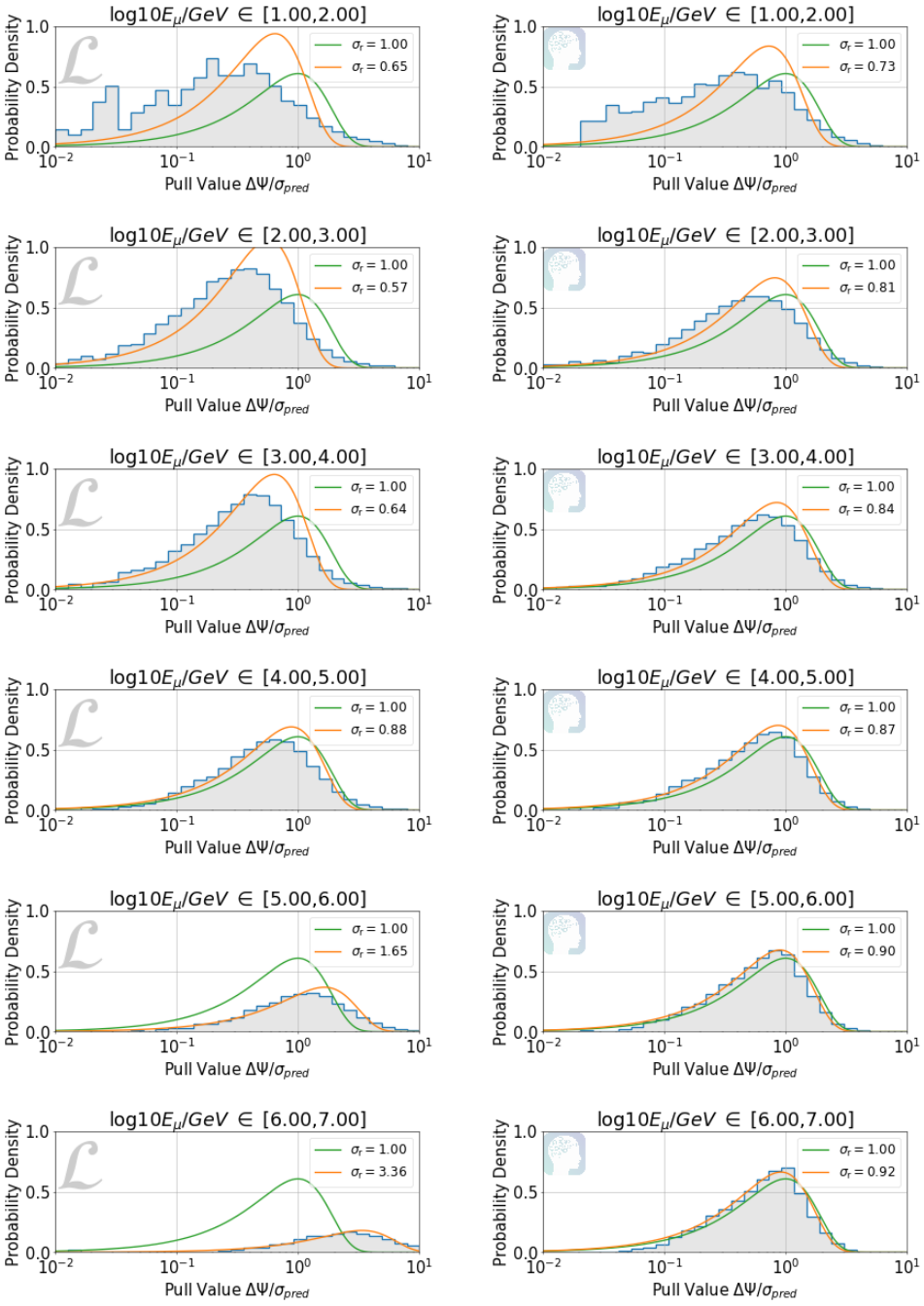


Figure 11.4: The pull distributions for the Cramér Rao Method (left) and the deep learning apraoch (right) are shown for different energy bins. The theoretically expected Rayleigh distribution is shown in green and the best fit in orange.

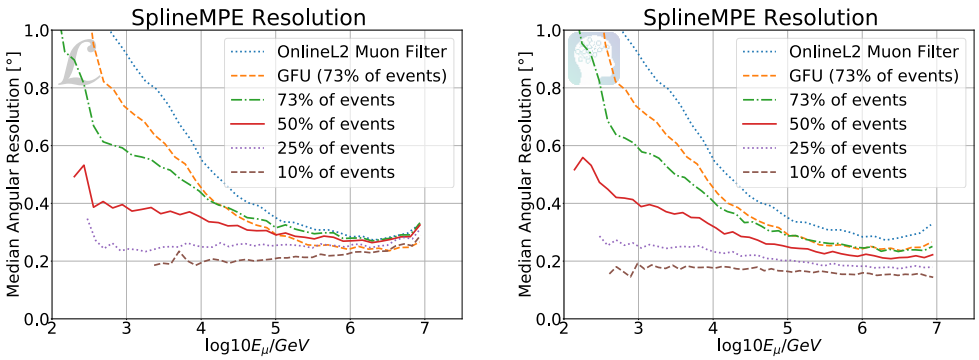


Figure 11.5: The uncertainty estimation for the standard directional reconstruction can be used to select well reconstructed tracks. On the left, the selection is made using Cramér Rao's (uncorrected) estimate, while the deep learning approach is used on the right. The angular resolution for events passing the GFU filter is shown for comparison.

11.4 Conclusion and Outlook

The deep neural network is able to estimate an uncertainty on its reconstruction. In addition, it can provide an estimate on the uncertainty of the standard reconstructions. This is qualitatively shown by the pull distributions and the selection of well reconstructed tracks. For a thorough investigation, Monte Carlo simulations have to be performed in which single events are re-simulated and re-constructed many times. Nevertheless, the results are very promising.

Currently, the standard energy reconstructions do not provide an event-by-event uncertainty estimate. However, a valid uncertainty estimate on the reconstruction quantities could greatly improve physics analyses.

In addition, for real-time alerts and follow-up programs, a valid uncertainty estimate is of utmost importance. The HESE alert[52] sent out on October 28$^{\text{th}}$, 2017, is an example in which the uncertainty on the online track reconstruction is heavily underestimated. Folllow-up millipede[59] scans reconstructed a track over 140° away from the initially reported position while the uncertainty on the online reconstruction was reported as 8.9°. Although the deep learning-based reconstruction is not yet tested on data, the estimated uncertainty of 142° seems very reasonable.

12 Runtime and Performance

The developed deep learning-based reconstruction method as presented in this thesis is designed to improve the Muon Online L2 Filter (see ch. 2.1). To be run online at the South Pole, strict restrictions on the computational requirements have to be met. In the following, the runtime of the deep learning approach is evaluated in comparison to the reconstructions performed for the Muon Online L2 Filter.

Module	**Mean Time**	σ	99 % **percentile**
SPE 2-it. Fit	0.085 s	0.129 s	0.670 s
MPE Fit	0.054 s	0.102 s	0.485 s
Cramér Rao Fits	0.049 ms	0.133 ms	0.560 ms
Bayesian Fit	1.022 ms	5.825 ms	0.024 s
Split Fits	0.066 s	0.236 s	1.291 s
MuEx	6.576 ms	0.023 s	0.091 s
Truncated Energy	2.612 ms	6.204 ms	0.022 s
Paraboloid	0.014 s	0.107 s	0.311 s
SplineMPE	0.036 s	0.152 s	0.793 s
All modules	0.273 s	0.605 s	3.297 s

Table 12.1: The runtime of the reconstructions performed for the Muon Online L2 Filter are shown. Only the SPE and MPE fits are run on all events of the Muon Filter stream. The other reconstructions are only run on events passing the Muon Online L2 Filter or on upgoing events, which means that their actual runtime is higher. [58, p.6]

The average runtime for the different reconstructions of the Muon Online L2 Filter is listed in tab. 12.1. The deep learning approach with the all-rounder model as described in ch. 7.1 has a runtime of (1.5 ± 0.2) ms per event on a Tesla P40. On a Quadro M1000M, the runtime for the prediction increases to (12 ± 2) ms per event. This does not include the time needed to load and preprocess the data.

Within the IceTray framework [20] it is not possible to perform the reconstruction simultaneously on a batch of events. Therefore, the batch size for the inference step has to be set to one. As a result, the time needed for prediction increases by approximately a factor of 4. An i3-module within the IceTray framework is

implemented to test the runtime for the Muon Online L2 Filter in comparison to the deep learning approach. The i3-module is tested on a data sample as well as on a simulation dataset. Both datasets only contain events which pass the Muon Online L2 Filter. Due to brighter events in the simulation dataset (simulated with a spectral index of $\gamma = -1$), the runtime is expected to be higher compared to the data sample. The test is performed on a notebook setup with a quadro M1000M graphics card and an Intel © Core™ i7-6700HQ CPU @2.60 GHz × 8. The results are displayed in fig.12.1.

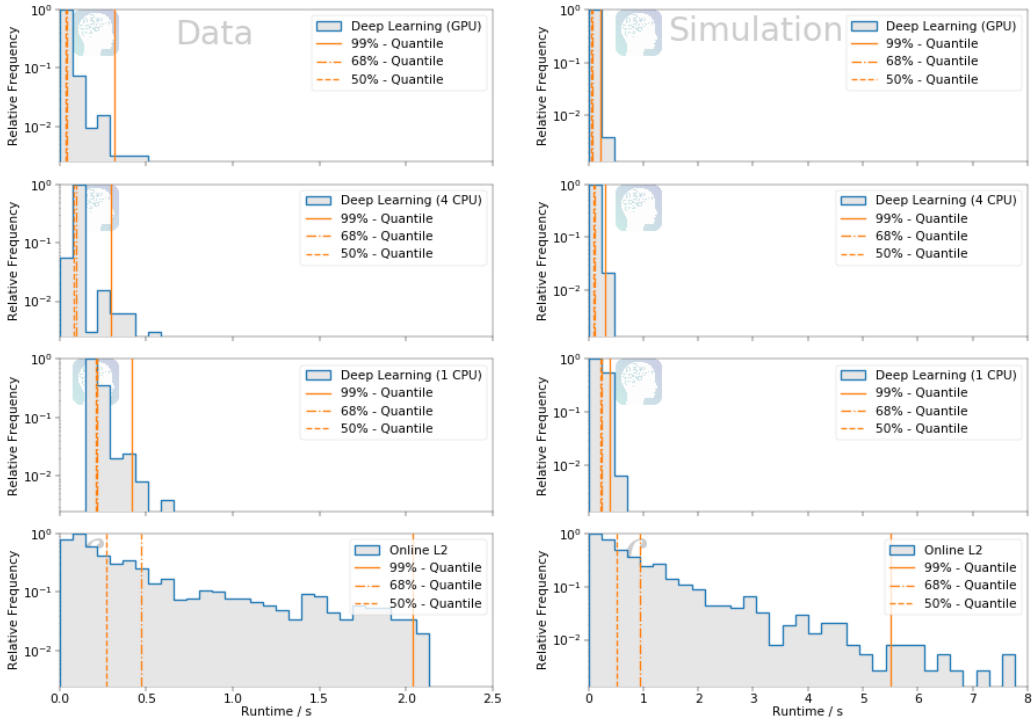


Figure 12.1: The distribution of per event runtime is shown for events passing the Muon Online L2 Filter. The simulation dataset is simulated with a spectral index of $\gamma = -1$ and therefore contains more high energy events than the data sample.

For comparison, the deep learning approach is also run on a CPU instead of a GPU. However, this is not preferred as advantages of the GPU's vectorization can not be exploited. The standard reconstructions are not capable of running on a GPU or on multiple CPUs in parallel.

The runtime of the deep learning approach in fig.12.1 is limited by the data prepro-

cessing which is currently implemented in python. This also adds a slight dependence of the runtime on the brightness of the event. As a result, the variance in the distribution of the runtime increases. The runtime of the prediction itself is independent of the event. In the current implementation, two loops over the measured pulses are performed. Transitioning the preprocessing to c++ should considerably speed up the runtime.

12.1 Fast Model

While developing the all-rounder model as described in ch. 7.1, the runtime did not play a key role. In case further restrictions on the runtime are imposed, the presented deep learning approach can be optimized for a short runtime. As an example, the number of filters are greatly reduced in the fast model (7.3) resulting in a decrease in the runtime for the prediction on a Quadro M1000M from (12 ± 2) ms down to (1.7 ± 0.9) ms per event. The reconstruction accuracy of this fast model is summarized in fig. 12.2.

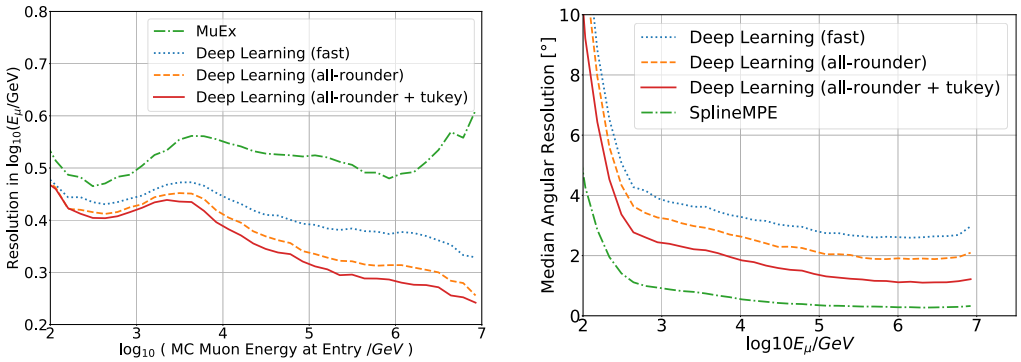


Figure 12.2: The results of the fast model are shown. On the left, the energy resolution for the muon energy at detector entry is shown and the median angular resolution is shown on the right.

Another approach to further improve the runtime is to reduce the number of layers in the network architecture or to investigate how important the input parameters are and if some of them are negligible. Omitting negligible input parameters will greatly reduce the time needed for preprocessing.

12.2 Conclusion and Outlook

The runtime of the deep learning-based approach is fast and essentially independent of the brightness of the event. Solely the preprocessing introduces a slight dependence on the number of pulses measured. The deep learning approach is therefore well suited to be run on-site at the South Pole. Moreover, if necessary, the runtime can be further improved by code optimization and pruning of the network architecture.

Furthermore, with the advent of Neural Processing Units (AI-chips) in smartphones [1][2], the optimization of neural network architectures and inference in regard to runtime as well as energy consumption is a field of active research. Significant advances are to be expected in the near future.

[1] http://www.dailymail.co.uk/sciencetech/article-4850136/Huawei-installing-AI-chips-smartphone.html
[2] https://www.androidauthority.com/smartphone-ai-processor-803019/

13 Conclusions

The developed deep learning-based reconstruction as presented in this thesis is currently the best reconstruction method available in IceCube for the reconstruction of the muon and neutrino energy of charged-current muon-neutrinos. In addition, the runtime of the reconstruction is very fast and stable, which makes the developed method a perfect candidate for the on-site employment at the South Pole. Not only can the energy resolution be greatly improved, but these reconstructions can be made available at the very beginning of the online processing chain, opening up new possibilities for real-time analyses and follow-up programs.

The most sophisticated reconstruction methods in IceCube such as millipede [59] do not reconstruct the muon or neutrino energy, but only the energy loss profile and the resulting deposited energy in the detector. As shown in app. A.7, the deep learning approach obtains similar results for the deposited energy. However, the millipede reconstruction can take hours to reconstruct a single event, while the deep learning approach has a runtime on the order of ms.

With the deep learning approach, outliers in the directional reconstruction can be greatly reduced. Currently, the standard directional reconstruction provides the best median angular resolution, however, possible improvements remain to be investigated for the deep learning-based reconstruction.

Furthermore, the developed method provides estimates on the uncertainty on all reconstructed quantities. The neural network can also be used to estimate the uncertainty on the standard reconstructions. Valid uncertainty estimates are of utmost importance for real-time alerts and follow-up programs.

All in all, the deep learning-based reconstruction method is a very powerful tool enhancing the current standard with a great potential for further improvements.

14 Outlook

The developed deep learning-based reconstruction is able to significantly increase the reconstruction accuracy while reducing the runtime in comparison to standard methods. Yet, many paths remain to be explored.

Further Improvements

While the deep learning approach is able to greatly boost the energy resolution, there is still room for improvement in the directional reconstruction. As mentioned in 10.3 the results of the standard directional reconstruction and the deep learning approach can be combined to further improve the angular resolution.

The directional reconstruction heavily relies on the timing information which is not yet optimally used in the network architecture. A use of the complete DOM waveform and a full four-dimensional data representation (ch. 4.3.2) could further improve the performance.

The neural network as presented in this thesis is trained on a single dataset. However, further datasets exist which can be used for additional training. Consequently, the network architecture can be increased without the risk of overfitting.

Moreover, the detector consists of multiple parts which are currently handled separately and only combined in subsequent steps. In addition, the DOMs of the main IceCube array are not distributed on a perfect grid. A solution is desired which is able to uniformly handle DOMs in arbitrary positions not restricted to a fixed grid. One option might be a type of continuous convolution over the input array. This is also of relevance for the possible detector upgrade which is planned to instrument strings in a sunflower configuration.

The success of convolutional networks is based on the exploitation of symmetries and a priori knowledge. In the physics use-case, many symmetries, invariances, and physical laws exist which could be used to further reduce the number of free parameters. A combination of likelihood-based methods and deep learning techniques seems very promising.

Moving Towards Data

While many options for possible improvements remain to be investigated, the next steps should include an investigation on the application of the deep learning-based method on data. Possible data-MC-mismatches as well as the effects of detector systematics need to be studied. A first test on historical EHE alerts[9] as described in appendix A.7 indicates that the deep learning-based reconstruction produces reasonable results comparable to the performed milllipede[59] scans.

Various options are available to overcome possible obstacles which may arise. Datasets simulated with different ice models and detector systematics exist. These can be used to train the network in order to achieve a robust reconstruction. Furthermore, a neural network autoencoder[14, 53] can be employed to train on data. Once the autoencoder is trained, it can be applied on the Monte Carlo datasets, resulting in robust features which can then be used as input into a neural network architecture such as presented in this thesis.

Other Applications

The presented neural network is intended to improve the event reconstructions for the Muon Online L2 Filter. However, it can be generalized to perform a classification of event topologies. Most of the online filters in use are based on straight cuts. A deep learning approach can possibly greatly improve the classification in order to obtain purer and more efficient data samples. Dedicated deep learning-based event reconstructions can then be performed for each event topology.

Another advantage of the deep learning approach is that labels can be freely defined. Currently, five different energy labels and many more directionally related labels are reconstructed as listed in tab. A.1. However, it is straight forward to define new labels if desired. This can be useful for analyses which require specific reconstruction quantities.

Ongoing work is put forth to better understand detector systematics. A neural network can be trained on real data to predict an event given the first few measured waveforms. Consequently, the network will learn to model the ice properties and light propagation. In contrast to standard parameterization approaches, the neural network is capable of fitting a non-linear function to data in a highly dimensional feature space. The trained network could then be used to simulate events and to study the ice properties.

All in all, deep learning techniques have great potential in physics. This successful feasibility study of a deep learning-based event reconstruction is just the beginning.

A Appendix

Label	Models	Description
PrimaryMuonEnergyEntry	All-Rounder, Fast Model, Track Uncertainty	Energy of muon at the entry point into the detector (convex hull + 60m)
PrimaryMuonEnergyCenter	All-Rounder	Energy of muon at the closest approach point to the center of the detector
PrimaryMuonEnergy	All-Rounder	Energy of muon
PrimaryEnergy	All-Rounder	Energy of neutrino
PrimaryInDetectorEnergyLoss	All-Rounder, Track Uncertainty	Energy deposited in detector by neutrino or any of its daughter particles
PrimaryMuonAzimuth	All-Rounder, Track Uncertainty	Azimuth angle of muon
PrimaryMuonZenith	All-Rounder, Track Uncertainty	Zenith angle of muon
PrimaryMuonDirectionX	All-Rounder, Fast Model, Track Uncertainty	X-coordinate of the muon direction vector
PrimaryMuonDirectionY	All-Rounder, Fast Model, Track Uncertainty	Y-coordinate of the muon direction vector
PrimaryMuonDirectionZ	All-Rounder, Fast Model, Track Uncertainty	Z-coordinate of the muon direction vector
PrimaryMuonEntryX	All-Rounder, Track Uncertainty	X-coordinate of the muon point of entry into the detector
PrimaryMuonEntryY	All-Rounder, Track Uncertainty	Y-coordinate of the muon point of entry into the detector
PrimaryMuonEntryZ	All-Rounder, Track Uncertainty	Z-coordinate of the muon point of entry into the detector
PrimaryMuonExitX	All-Rounder, Track Uncertainty	X-coordinate of the muon at the exit point out of the detector
PrimaryMuonExitY	All-Rounder, Track Uncertainty	Y-coordinate of the muon at the exit point out of the detector
PrimaryMuonExitZ	All-Rounder, Track Uncertainty	Z-coordinate of the muon at the exit point out of the detector
PrimaryMuonCenterX	All-Rounder, Track Uncertainty	X-coordinate of the muon closest approach point to the center of the detector
PrimaryMuonCenterY	All-Rounder, Track Uncertainty	Y-coordinate of the muon closest approach point to the center of the detector
PrimaryMuonCenterZ	All-Rounder, Track Uncertainty	Z-coordinate of the muon closest approach point to the center of the detector
SplineMPETrackUncertainty	Track Uncertainty	Angular error estimate on the SplineMPE track reconstruction
SplineMPEAzimuthUncertainty	Track Uncertainty	Estimated uncertainty on the SplineMPE Azimuth reconstruction
SplineMPEZenithUncertainty	Track Uncertainty	Estimated uncertainty on the SplineMPE Zenith reconstruction

Table A.1: An overview of the reconstructed labels is shown. The models which reconstruct the given labels are listed in the second column.

A.1 All-Rounder Model

In the following, supplementary material is shown for the all-rounder model as described in ch. 7.1.

A.1.1 All-Rounder Model Architecture and Training

A detailed description of the neural network architecture and training procedure of the all-rounder model (ch. 7.1) is given in the following tables.

Num. Filters	Filter Size	Padding	Activation	Res. Add.	Max Pooling
100	hex. [2,0,3]	same	relu	Yes	No
100	hex. [2,0,3]	same	relu	Yes	[2,2,2]
100	hex. [1,0,15]	same	relu	Yes	No
100	hex. [2,0,3]	same	relu	Yes	[1,1,2]
100	hex. [1,0,15]	same	relu	Yes	No
100	hex. [2,0,3]	same	relu	Yes	[2,2,2]
100	hex. [2,0,2]	same	relu	Yes	[1,1,2]
100	hex. [2,0,3]	same	relu	Yes	No
100	hex. [2,0,3]	same	relu	Yes	No
100	hex. [1,0,1]	same	relu	Yes	No
100	hex. [1,0,1]	same	relu	Yes	No

Table A.2: An overview of the 11 convolutional layers over the main IceCube array input is shown. Dropout is applied to all layers.

Num. Filters	Filter Size	Padding	Activation	Res. Add.	Max Pooling
100	[8,5]	same	relu	Yes	[1,2]
100	[8,7]	same	relu	Yes	No
100	[8,7]	same	relu	Yes	No
100	[8,3]	valid	relu	No	No
100	[1,7]	same	relu	Yes	[1,2]
100	[1,7]	same	relu	Yes	No
100	[1,1]	same	relu	Yes	No
100	[1,1]	same	relu	Yes	No

Table A.3: The convolutional layers over the DeepCore input are described. Dropout is applied to all layers.

	Num. Neurons	Dropout	Activation	Residual Addition
Prediction Layers:	300	Yes	relu	No
	19	No	none	Yes
Uncertainty Layers:	50	No	relu	No
	50	No	elu	No
	19	No	none	No

Table A.4: An overview of the fully connected layers for the prediction and uncertainty estimation are shown. The combined and flattened result of the convolutional layers is used as input.

Num. Steps	Learning Rate	Dropout Rate		File Epochs	Files at Once	Batch Size	Label Importance
30000	10^{-3}	0.2	0.5	3	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1
10000	10^{-3}	0.0	0.3	3	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1
10000	10^{-4}	0.0	0.2	3	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1
30500	10^{-4}	0.0	0.0	1	200	256	All labels: 1
2000	10^{-5}	0.0	0.0	1	200	256	All labels: 1

Table A.5: The all-rounder model is trained in 5 steps as described above. The dropout rate for the convolutional layers is given in the first dropout column. The second column is the dropout rate for the fully connected layers.

A.1.2 All-Rounder Model Results

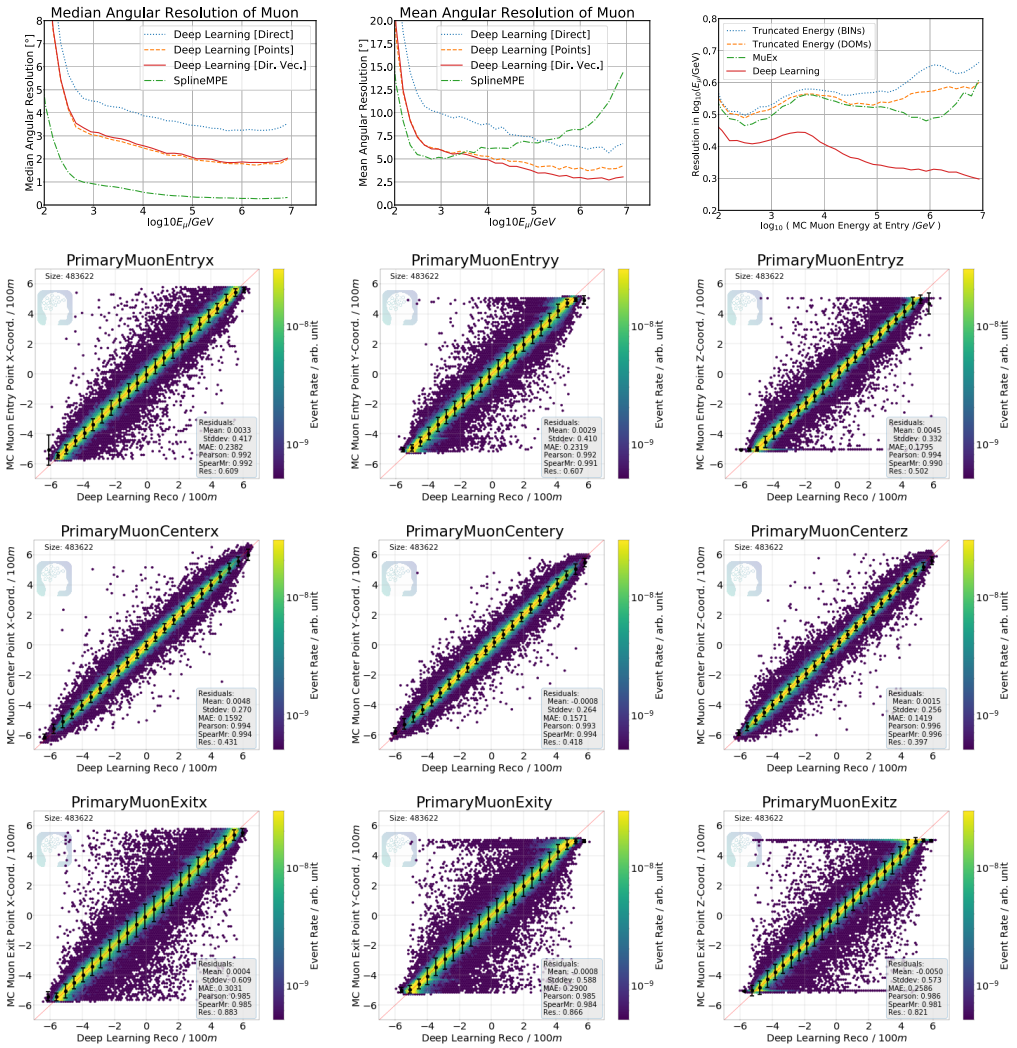


Figure A.1: The results for the all-rounder model are shown.

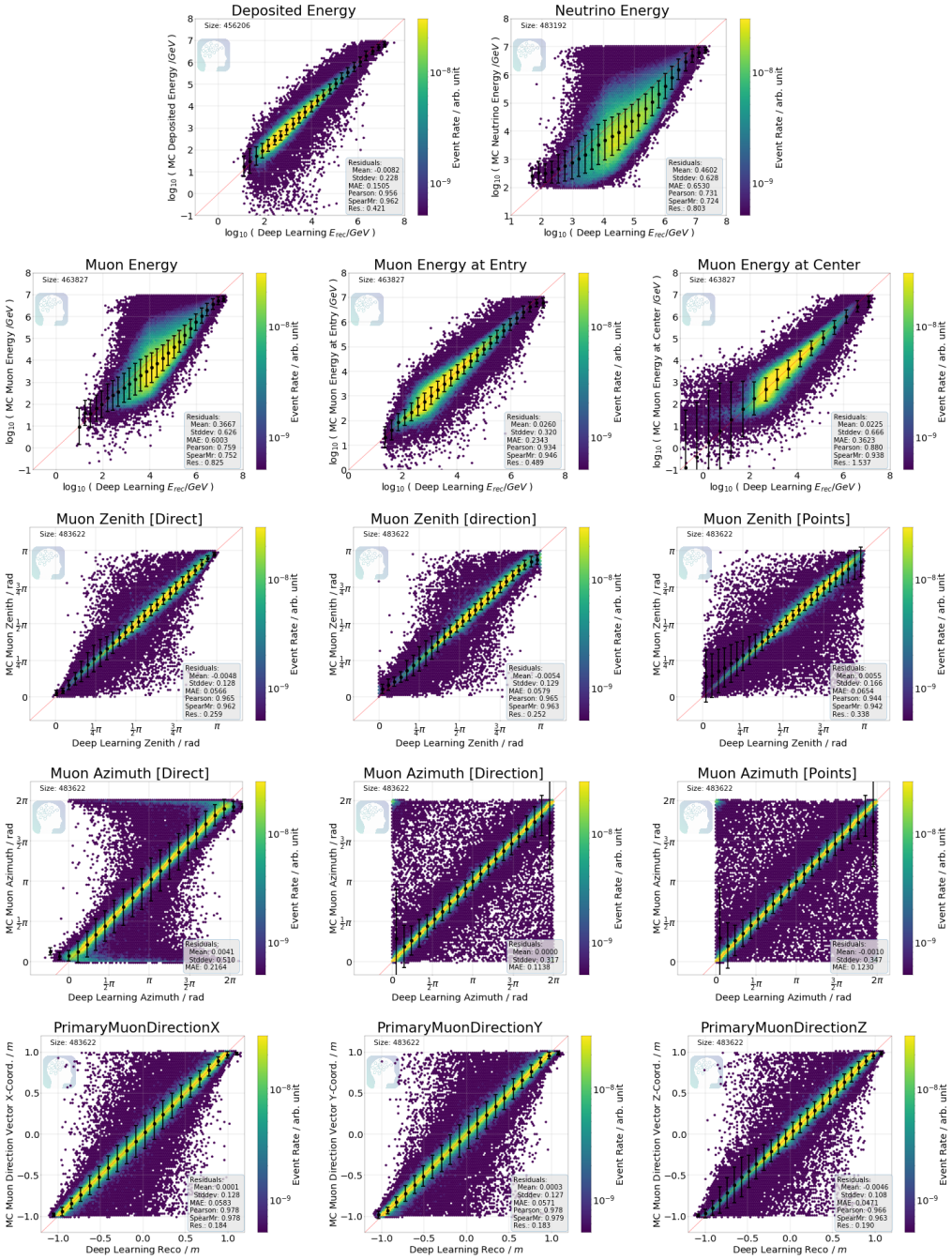


Figure A.2: Correlation plots for the all-rounder model are shown.

A.2 All-Rounder Model – Tuned to GFU Filter

Num. Steps	Learning Rate	Dropout Rate		File Epochs	Files at Once	Batch Size	Label Importance
38000	10^{-3}	0.0	0.0	1	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1
22500	10^{-4}	0.0	0.0	1	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1
23000	10^{-5}	0.0	0.0	1	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1

Table A.6: The all-rounder model is fine-tuned to events passing the GFU filter. The first three steps as described in tab. A.5 are kept, while the training steps above are performed only on events passing the GFU filter.

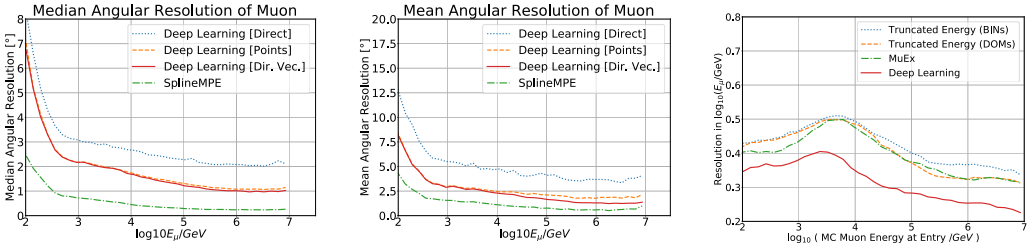


Figure A.3: The results for the GFU fine-tuned all-rounder model are shown for events passing the GFU filter.

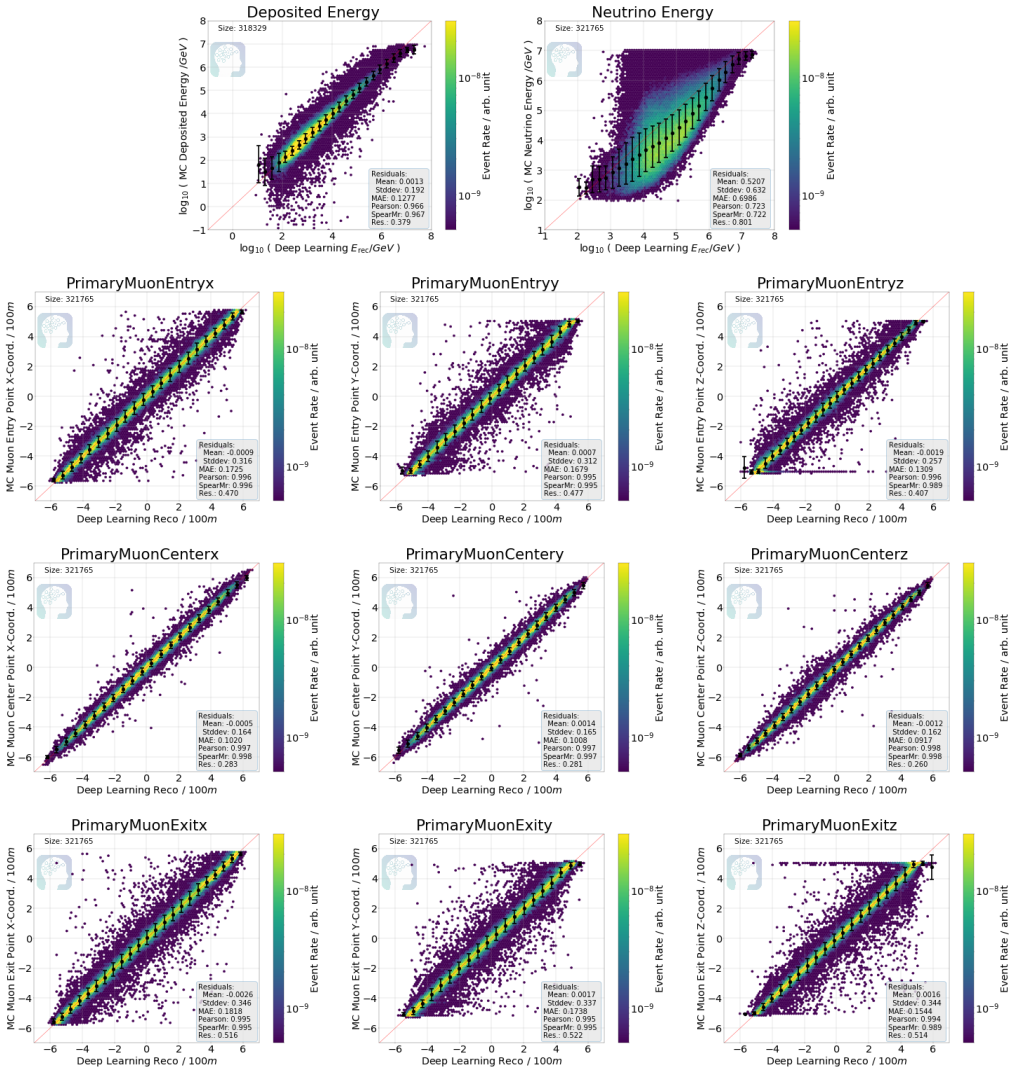


Figure A.4: The results for the GFU fine-tuned all-rounder model are shown for events passing the GFU filter.

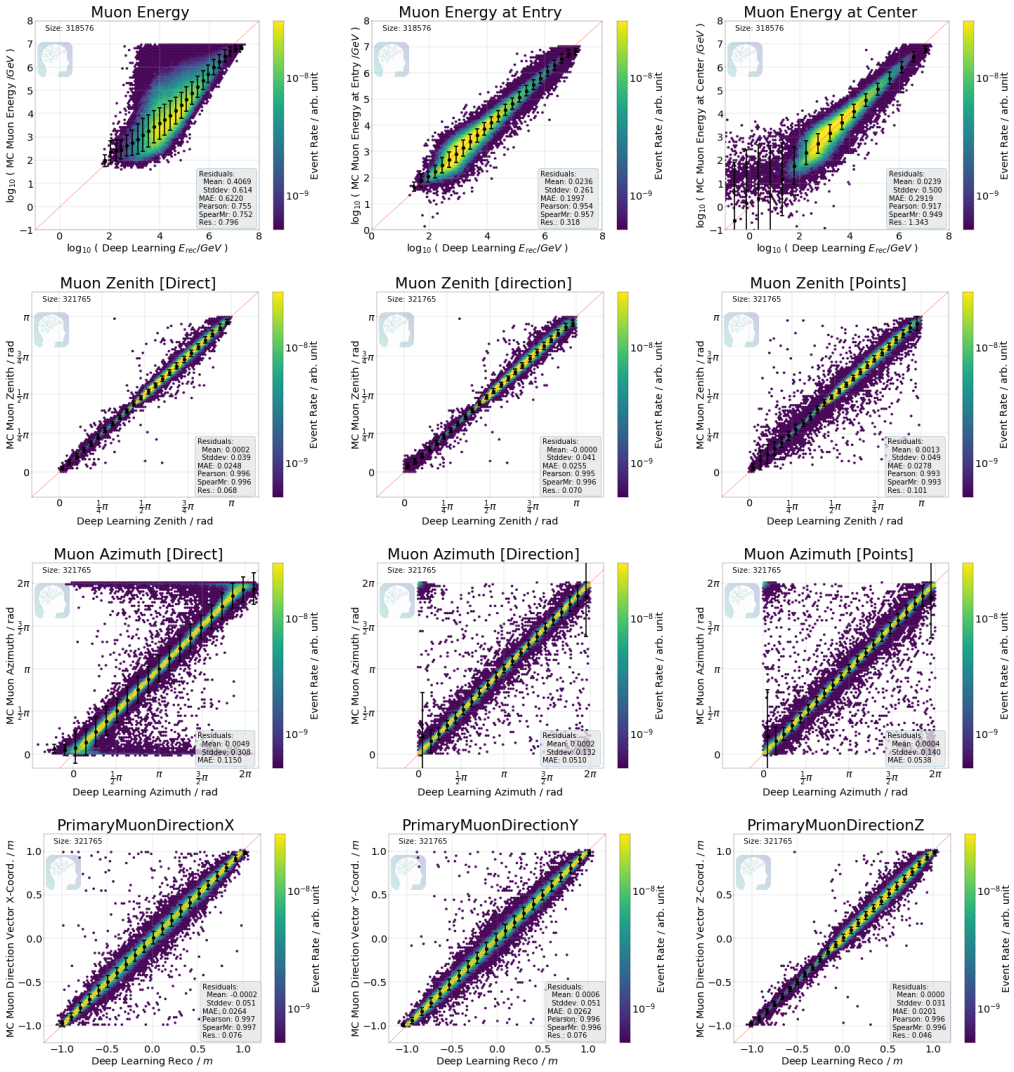


Figure A.5: The results for the GFU fine-tuned all-rounder model are shown for events passing the GFU filter.

A.3 All-Rounder Model – Tuned to Final Level

Num. Steps	Learning Rate	Dropout Rate		File Epochs	Files at Once	Batch Size	Label Importance
25000	10^{-3}	0.0	0.0	1	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1
23500	10^{-4}	0.0	0.0	1	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1
18000	10^{-5}	0.0	0.0	1	200	256	PrimaryEnergy: 0.2 PrimaryMuonEnergy: 0.2 PrimaryMuonEnergyEntry: 2 All other labels: 1

Table A.7: The all-rounder model is fine-tuned to the point source final level. The first three steps as described in tab. A.5 are kept, while the training steps above are performed only on events of the final data level.

A.3.1 Results

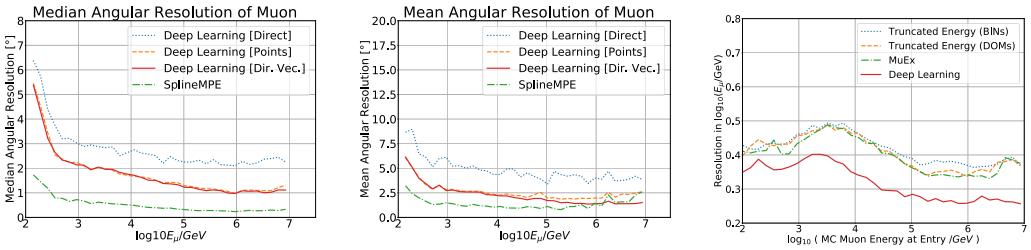


Figure A.6: The results for the all-rounder model fine-tuned to the final point source level are shown.

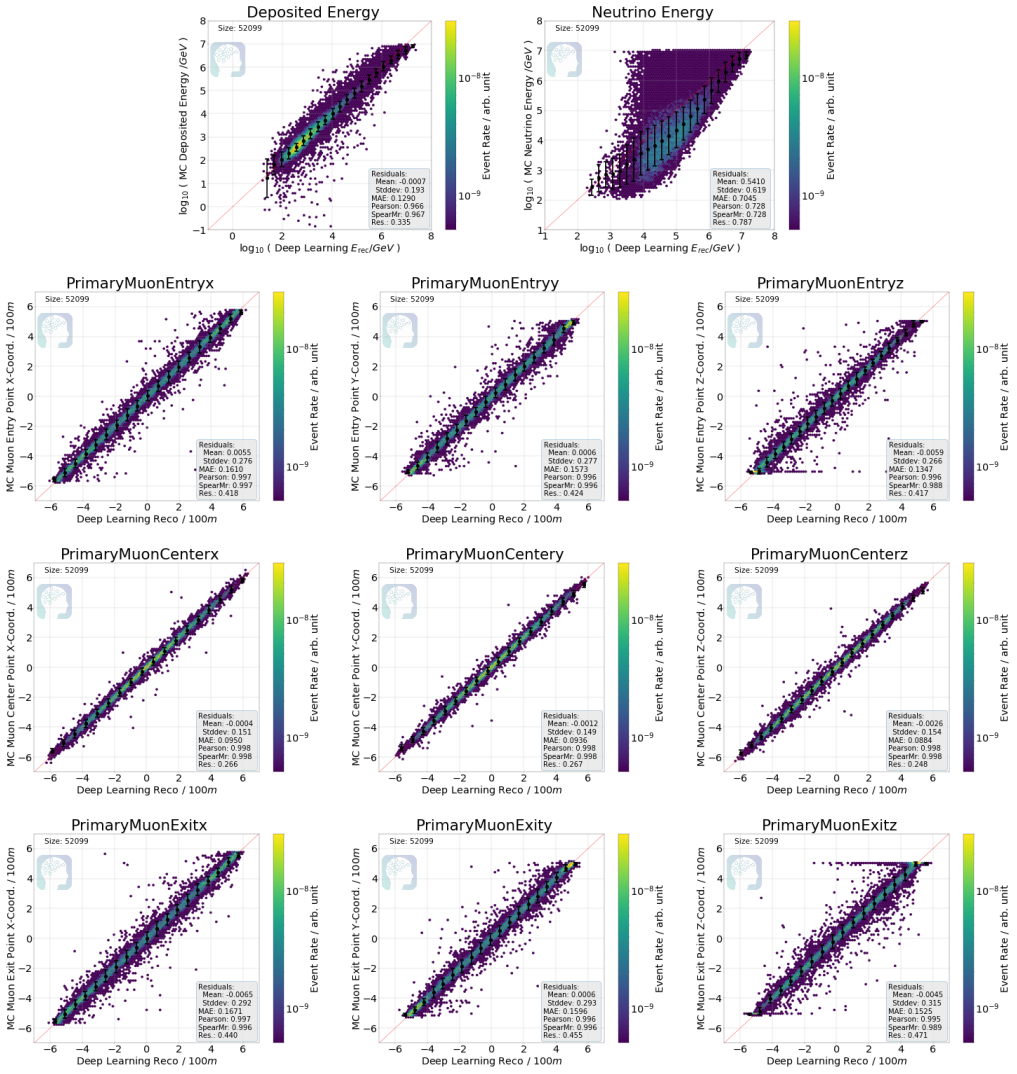


Figure A.7: The results for the all-rounder model fine-tuned to the final point source level are shown.

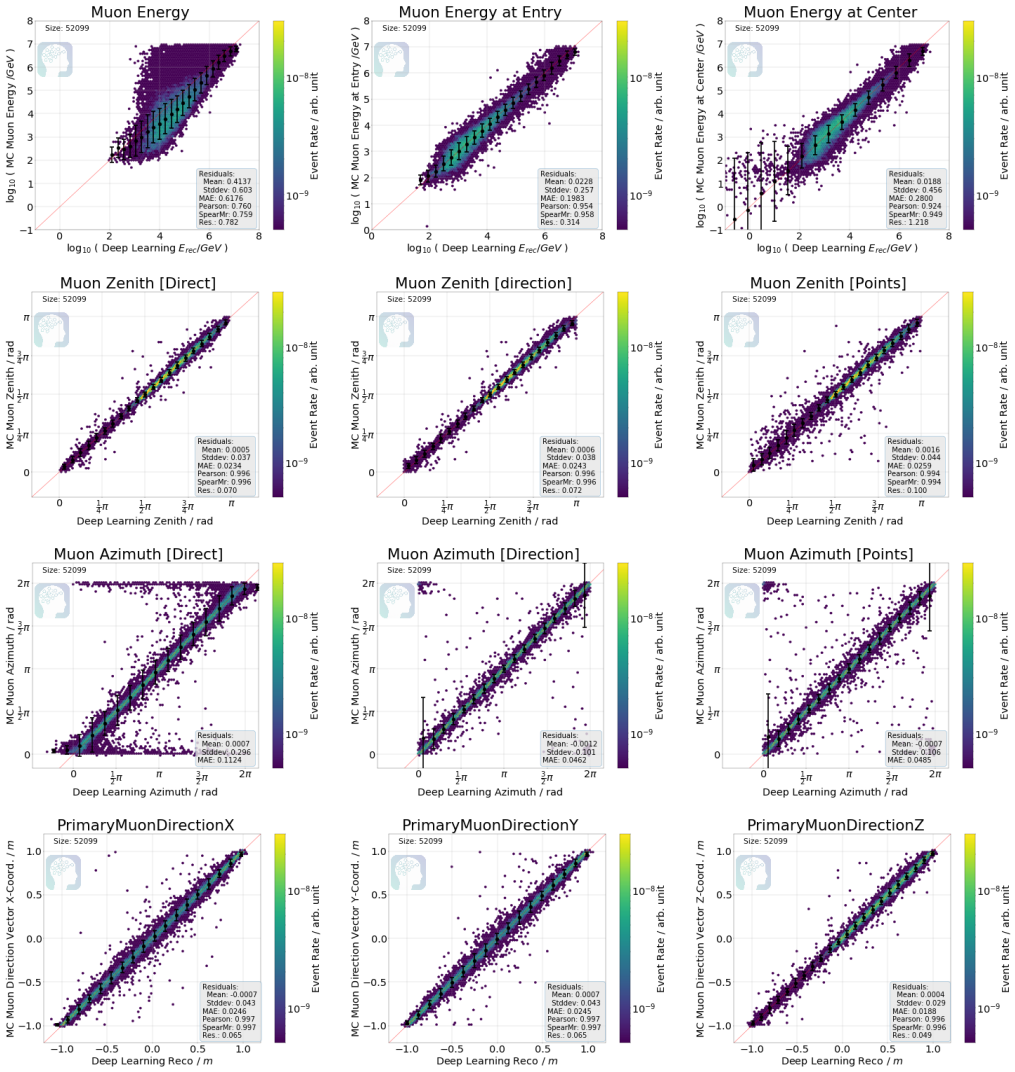


Figure A.8: The results for the all-rounder model fine-tuned to the final point source level are shown.

A.4 All-Rounder Model – Tuned with Tukey Loss

Num. Steps	Learning Rate	Dropout Rate		File Epochs	Files at Once	Batch Size	Label Importance
43500	10^{-3}	0.0	0.0	1	200	256	All labels: 1
58500	10^{-4}	0.0	0.0	1	200	256	All labels: 1
1500	10^{-5}	0.0	0.0	1	200	256	All labels: 1

Table A.8: The all-rounder model is fine-tuned with the tukey loss. The first three steps as described in tab. A.5 are kept, while the training steps above are performed with the tukey loss instead of the mean squared error.

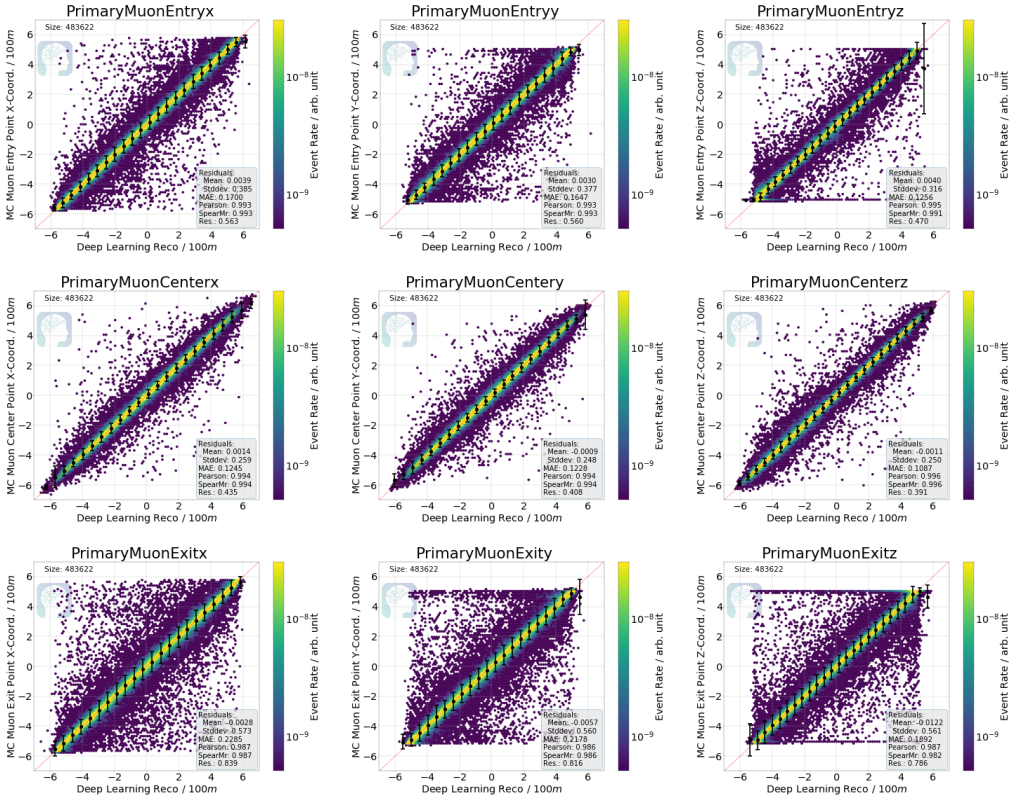


Figure A.9: The results for the tukey fine-tuned all-rounder model are shown.

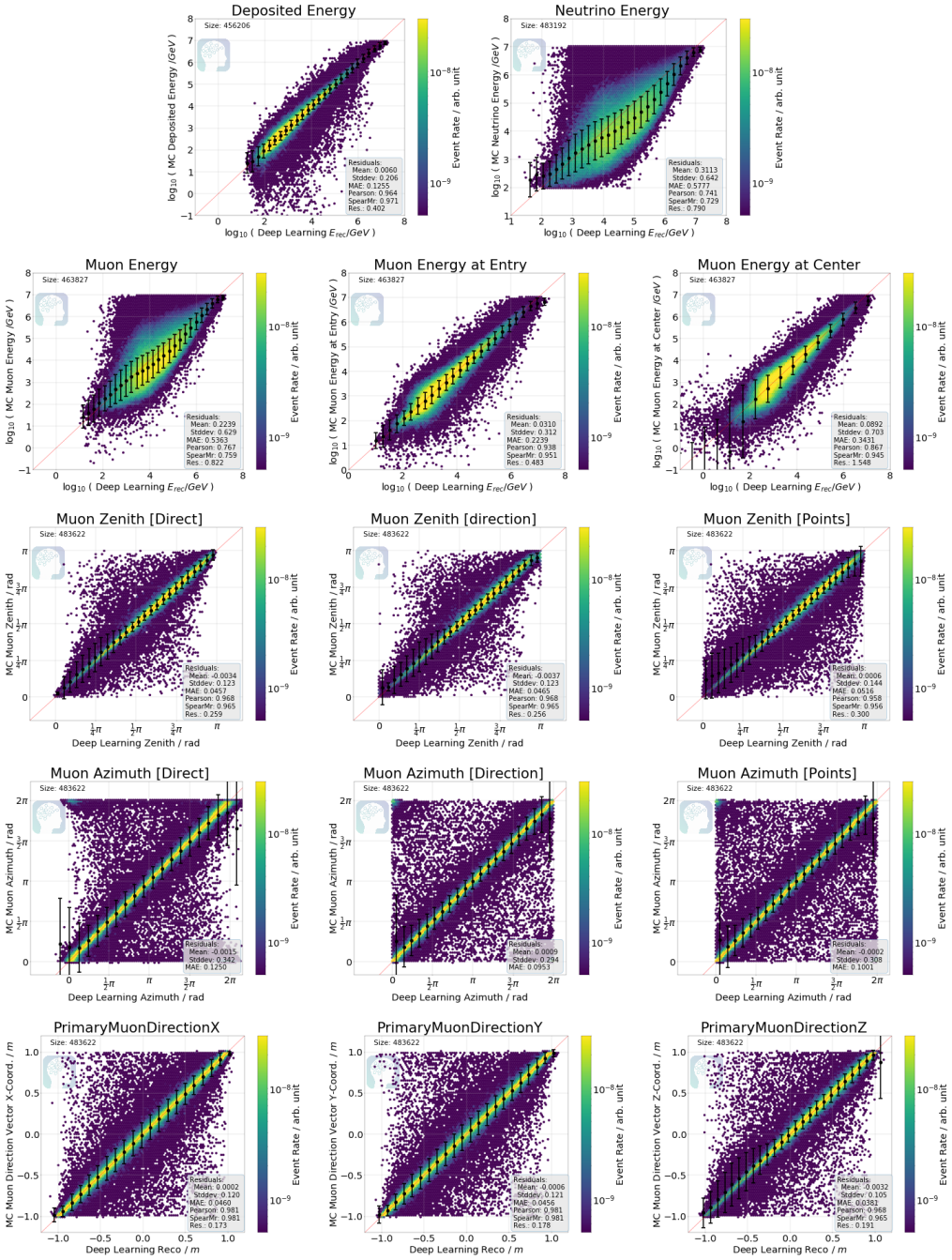


Figure A.10: The results for the tukey fine-tuned all-rounder model are shown.

A.5 Track Uncertainty Model

Num. Steps	Learning Rate	Dropout Rate		File Epochs	Files at Once	Batch Size	Label Importance
32000	10^{-3}	0.2	0.5	3	70	256	SplineMPETrackUncertainty: 3 All other labels: 1
17000	10^{-3}	0.0	0.3	3	100	256	SplineMPETrackUncertainty: 3 All other labels: 1
21000	10^{-4}	0.0	0.2	3	100	256	SplineMPETrackUncertainty: 3 All other labels: 1
11500	10^{-4}	0.0	0.0	1	100	256	SplineMPETrackUncertainty: 3 All other labels: 1

Table A.9: The track uncertainty model is trained in 4 steps as described above. The dropout rate for the convolutional layers is given in the first dropout column, while the dropout rate for the fully connected layers is given in the second column.

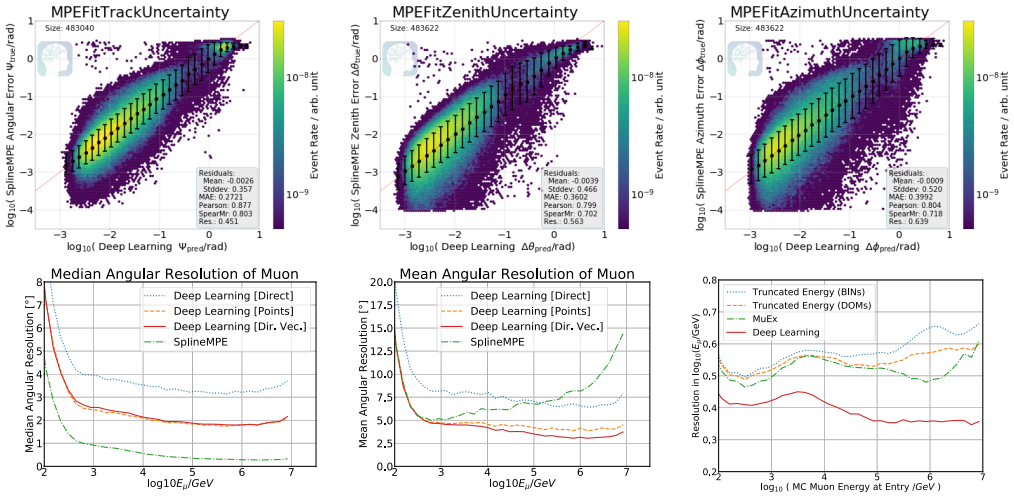


Figure A.11: Results of the track uncertainty model are shown.

A.6 Fast Model

In the following, supplementary material is shown for the fast model as described in ch. 7.3.

Num. Filters	Filter Size	Padding	Activation	Res. Add.	Max Pooling
20	hex. [2,0,3]	same	relu	Yes	No
20	hex. [2,0,3]	same	relu	Yes	[2,2,2]
20	hex. [1,0,15]	same	relu	Yes	No
20	hex. [2,0,3]	same	relu	Yes	[1,1,2]
20	hex. [1,0,15]	same	relu	Yes	No
20	hex. [2,0,3]	same	relu	Yes	[2,2,2]
20	hex. [2,0,2]	same	relu	Yes	[1,1,2]
20	hex. [2,0,3]	same	relu	Yes	No
20	hex. [2,0,3]	same	relu	Yes	No
20	hex. [1,0,1]	same	relu	Yes	No

Table A.10: An overview of the 10 convolutional layers over the main IceCube array input is shown. Dropout is applied to all layers.

Num. Filters	Filter Size	Padding	Activation	Res. Add.	Max Pooling
20	[8,5]	same	relu	Yes	[1,2]
20	[8,7]	same	relu	Yes	No
20	[8,7]	same	relu	Yes	No
20	[8,3]	valid	relu	No	No
20	[1,7]	same	relu	Yes	[1,2]
20	[1,7]	same	relu	Yes	No
20	[1,1]	same	relu	Yes	No

Table A.11: The convolutional layers over the DeepCore input are described. Dropout is applied to all layers.

	Num. Neurons	Dropout	Activation	Residual Addition
Prediction Layers:	100	Yes	relu	No
	4	No	none	Yes
Uncertainty Layers:	20	No	relu	No
	20	No	elu	No
	4	No	exp	No

Table A.12: An overview of the fully connected layers for the prediction and uncertainty estimation are shown. The combined and flattened result of the convolutional layers is used as input.

Num. Steps	Learning Rate	Dropout Rate		File Epochs	Files at Once	Batch Size	Label Importance
101500	10^{-2}	0.0	0.0	20	200	512	All labels: 1
52500	10^{-3}	0.0	0.0	6	200	512	All labels: 1
31500	10^{-4}	0.0	0.0	1	200	512	All labels: 1

Table A.13: The fast model is trained in 3 steps as described above. The dropout rate for the convolutional layers is given in the first dropout column, while the dropout rate for the fully connected layers is given in the second column.

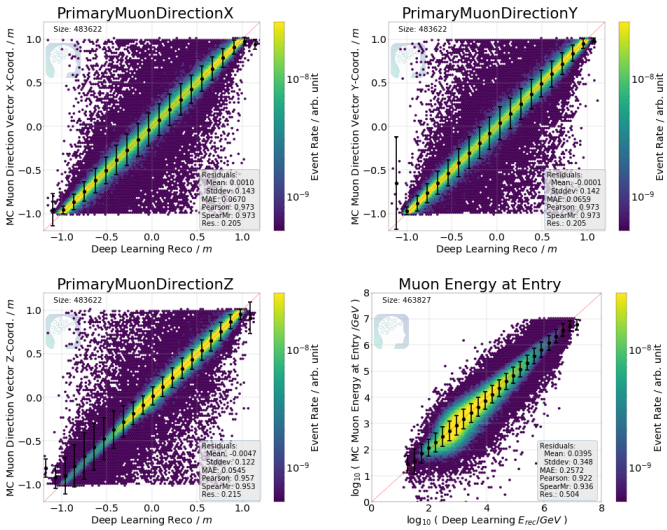


Figure A.12: Results for the fast model are shown.

A.7 EHE Alerts

The application of the deep learning-based reconstruction on data has not been investigated yet. However, a first test on historical EHE alerts[9] is shown here. For these events, computationally expensive millipede [59] scans are performed. The millipede scans reconstruct the track as well as the deposited energy in the detector and are currently the best possible reconstruction available in IceCube. A likelihood scan for a single event can require several hours of runtime. The results of the developed deep learning-based reconstruction are compatible with the results obtained by the millipede scan. In tab. A.14, the reconstruction of the

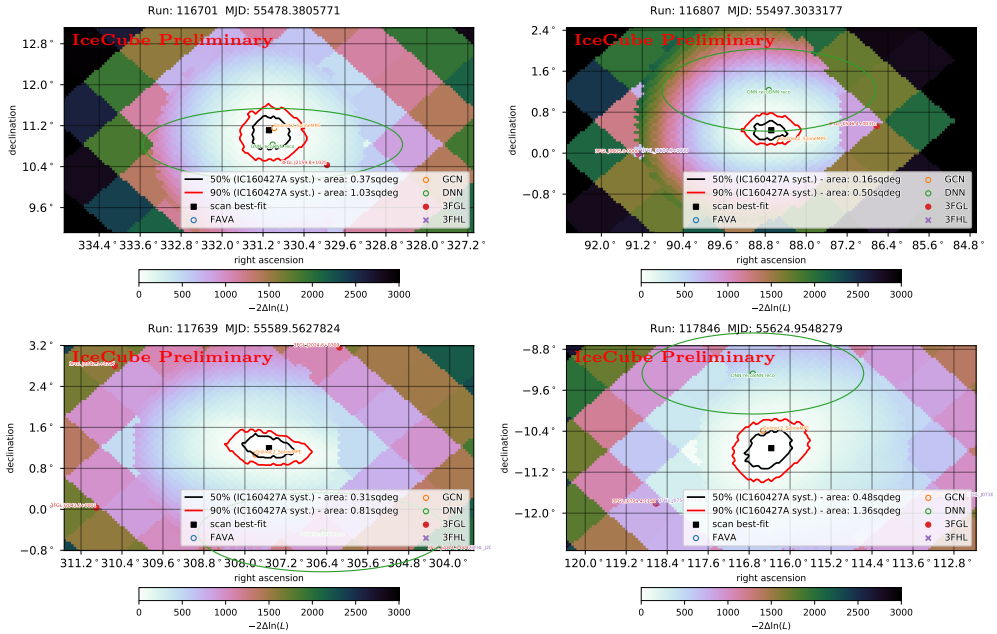


Figure A.13: Millipede scans for historical EHE events are shown.

deposited energy in the detector is compared. Millipede scans the likelihood to obtain the best fit track as well as the 50 % and 90 % confidence intervals. The results are shown in fig. A.13, A.14, A.15, A.16. For the deep learning-approach, the 1 σ-error contours are plotted. Although, this is just a qualitative check, the results of the deep learning-method seem reasonable and are in overall agreement with the millipede reconstruction.

EHE Event	**Millipede Scan** /TeV	**Deep Learning** /TeV **(all-rounder + tukey)**
2010 alert 0	82.61	105^{+32}_{-25}
2010 alert 1	108.85	91^{+28}_{-21}
2010 alert 2	177.59	138^{+52}_{-38}
2010 alert 3	143.30	70^{+22}_{-16}
2011 alert 0	26.53	27^{+8}_{-6}
2011 alert 1	170.77	126^{+63}_{-42}
2012 alert 0	40.29	51^{+23}_{-16}
2012 alert 1	33.15	34^{+12}_{-9}
2012 alert 2	64.78	77^{+24}_{-18}
2013 alert 0	880.83	1243^{+406}_{-306}
2013 alert 1	98.72	43^{+112}_{-31}
2013 alert 2	13.61	16^{+6}_{-4}
2013 alert 3	184.37	212^{+99}_{-67}
2013 alert 4	507.70	429^{+173}_{-123}
2013 alert 5	24.45	10^{+6}_{-4}
2014 alert 0	2370.78	3431^{+1895}_{-1221}
2014 alert 1	28.21	34^{+11}_{-8}
2015 alert 0	37.87	42^{+13}_{-10}
2015 alert 1	42.46	65^{+27}_{-19}
2015 alert 2	21.48	24^{+8}_{-6}
2015 alert 3	53.86	53^{+19}_{-14}
2015 alert 4	38.79	37^{+11}_{-9}
2015 alert 5	32.63	34^{+11}_{-9}
2015 alert 6	22.11	18^{+5}_{-4}
2016 alert 0	134.95	282^{+115}_{-82}
2016 alert 1	36.15	25^{+9}_{-7}
2016 alert 2	55.81	25^{+31}_{-14}
2016 alert 3	16.95	16^{+7}_{-5}

Table A.14: The reconstructed deposited energy of historical EHE alerts is shown for the millipede scan and the deep learning approach.

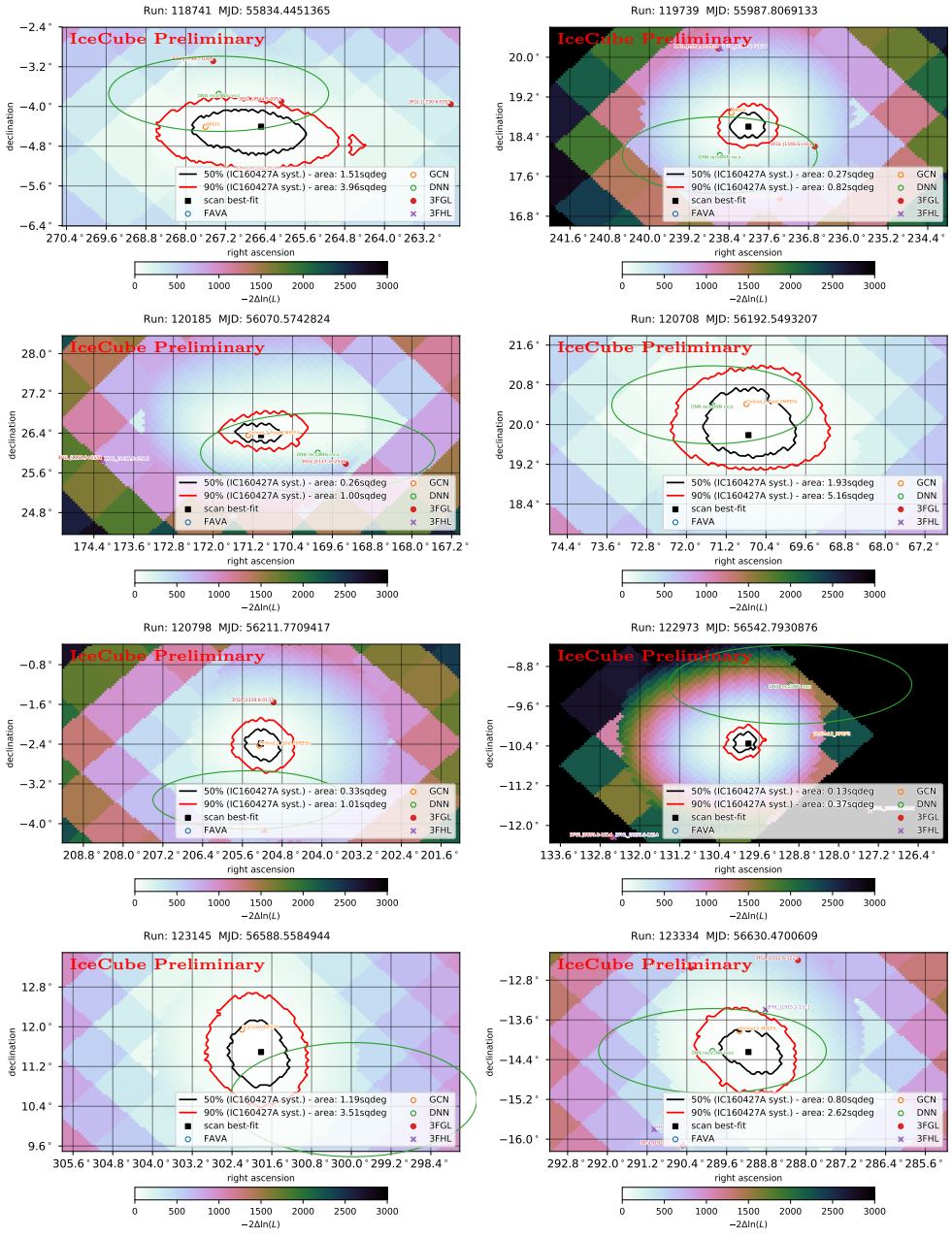


Figure A.14: Millipede scans for historical EHE events are shown.

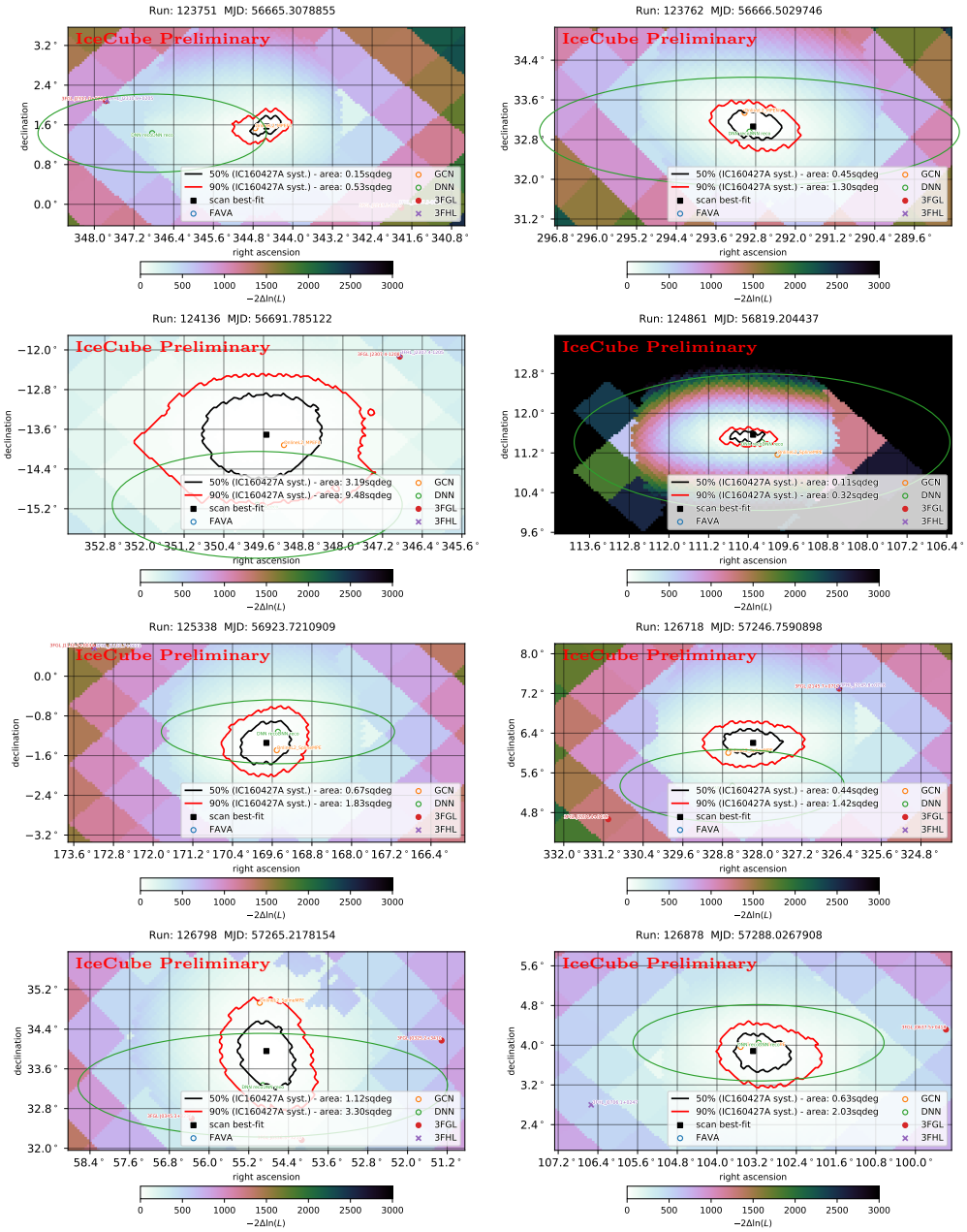


Figure A.15: Millipede scans for historical EHE events are shown.

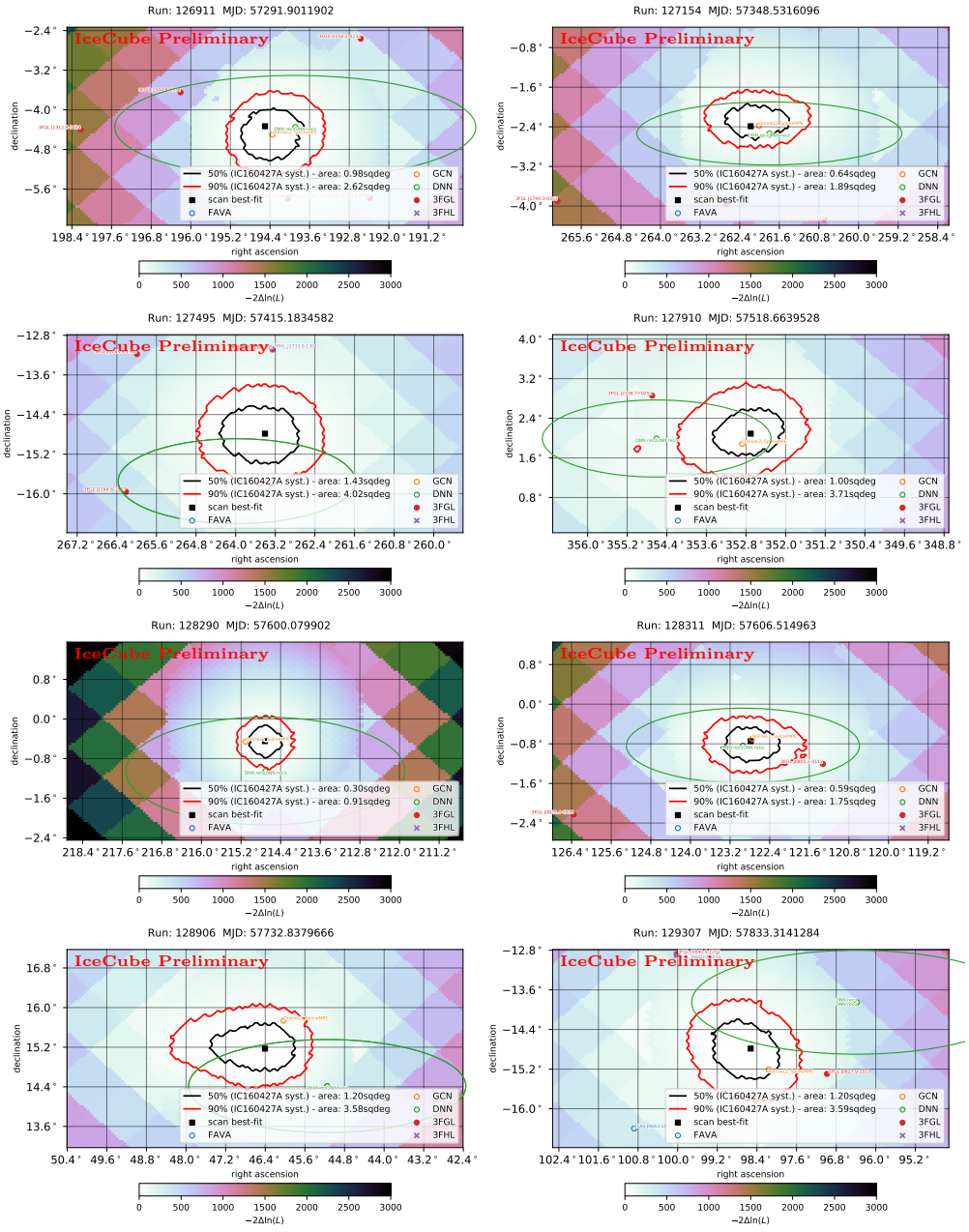


Figure A.16: Millipede scans for historical EHE events are shown.

Bibliography

[1] **IceCube, MAGIC, VERITAS** Collaboration, M. G. Aartsen et al. "Very High-Energy Gamma-Ray Follow-Up Program Using Neutrino Triggers from IceCube". In: *JINST* 11.11 (2016), P11009. DOI: 10.1088/1748-0221/11/11/P11009.

[2] **IceCube** Collaboration, M. G. Aartsen et al. "Evidence for Astrophysical Muon Neutrinos from the Northern Sky with IceCube". In: *Phys. Rev. Lett.* 115 (8 Aug. 2015), p. 081102. DOI: 10.1103/PhysRevLett.115.081102.

[3] **IceCube** Collaboration, M. G. Aartsen et al. "Evidence for High-Energy Extraterrestrial Neutrinos at the IceCube Detector". In: *Science* 342.6161 (2013). ISSN: 0036-8075. DOI: 10.1126/science.1242856.

[4] **IceCube** Collaboration, M. G. Aartsen et al. "Observation of High-Energy Astrophysical Neutrinos in Three Years of IceCube Data". In: *Phys. Rev. Lett.* 113 (10 Sept. 2014), p. 101101. DOI: 10.1103/PhysRevLett.113.101101.

[5] **IceCube** Collaboration, M.G. Aartsen et al. "Energy reconstruction methods in the IceCube neutrino telescope". In: *JINST* 9.03 (2014), P03009. DOI: 10.1088/1748-0221/9/03/P03009.

[6] **IceCube** Collaboration, M.G. Aartsen et al. "Neutrino oscillation studies with IceCube-DeepCore". In: *Nuclear Physics B* 908.Supplement C (2016), pp. 161–177. ISSN: 0550-3213. DOI: 10.1016/j.nuclphysb.2016.03.028.

[7] **IceCube** Collaboration, M.G. Aartsen et al. "Observation and Characterization of a Cosmic Muon Neutrino Flux from the Northern Hemisphere Using Six Years of IceCube Data". In: *The Astrophysical Journal* 833.1 (2016), p. 3. DOI: 10.3847/0004-637X/833/1/3.

[8] **IceCube** Collaboration, M.G. Aartsen et al. "The IceCube Neutrino Observatory: instrumentation and online systems". In: *JINST* 12.03 (2017), P03012. DOI: 10.1088/1748-0221/12/03/P03012.

[9] **IceCube** Collaboration, M.G. Aartsen et al. "The IceCube realtime alert system". In: *Astropart. Phys.* 92 (2017), pp. 30–41. ISSN: 0927-6505. DOI: 10.1016/j.astropartphys.2017.05.002.

[10] Martín Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *CoRR* abs/1603.04467 (2016). URL: http://arxiv.org/abs/1603.04467.

[11] **IceCube** Collaboration, R. Abbasi et al. "An improved method for measuring muon energy using the truncated mean of dE/dx". In: *Nucl. Instrum. Meth.* A703 (2013), pp. 190–198. DOI: 10.1016/j.nima.2012.11.081.

[12] **ANTARES** Collaboration, J. A. Aguilar et al. "A fast algorithm for muon track reconstruction and its application to the ANTARES neutrino telescope". In: *Astropart. Phys.* 34 (2011), pp. 652–662. DOI: 10.1016/j.astropartphys.2011.01.003.

[13] J. Ahrens et al. "Muon track reconstruction and data selection techniques in AMANDA". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 524.1 (2004), pp. 169–194. ISSN: 0168-9002. DOI: 10.1016/j.nima.2004.01.065.

[14] Pierre Baldi. "Autoencoders, Unsupervised Learning, and Deep Architectures". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Ed. by Isabelle Guyon et al. Vol. 27. Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR, Feb. 2012, pp. 37–49. URL: http://proceedings.mlr.press/v27/baldi12a.html.

[15] Vasileios Belagiannis et al. "Robust Optimization for Deep Regression". In: *CoRR* abs/1505.06606 (2015). DOI: 10.1109/ICCV.2015.324.

[16] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 08856125. DOI: 10.1023/A:1010933404324.

[17] G. E. Dahl et al. "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1 (Jan. 2012), pp. 30–42. ISSN: 1558-7916. DOI: 10.1109/TASL.2011.2134090.

[18] Luc Demortier and Louis Lyons. *Everything you always wanted to know about pulls*. 2002. URL: http://physics.rockefeller.edu/luc/technical_reports/cdf5776_pulls.pdf.

[19] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[20] T. DeYoung. "IceTray: A software framework for IceCube". In: *Computing in high energy physics and nuclear physics. Proceedings, Conference, CHEP'04, Interlaken, Switzerland, September 27-October 1, 2004*. 2005, pp. 463–466. URL: http://doc.cern.ch/yellowrep/2005/2005-002/p463.pdf.

[21] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back Propagating Errors". In: *Nature* 323 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0.

[22] Yoav Freund, Robert E Schapire, et al. "Experiments with a new boosting algorithm". In: *Icml*. Vol. 96. 1996, pp. 148–156.

[23] F. A. Gers, J. Schmidhuber, and F. Cummins. "Learning to forget: continual prediction with LSTM". In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[24] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Vol. 15. Proceedings of Machine Learning Research. PMLR, Nov. 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a.html.

[25] **IceCube** Collaboration, Christian Hack and Christopher Wiebusch. "A Measurement of the Diffuse Astrophysical Muon Neutrino Flux Using Eight Years of IceCube Data". In: vol. 1005. PoS, 2017. URL: https://pos.sissa.it/cgi-bin/reader/contribution.cgi?id=PoS(ICRC2017)1005.

[26] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). URL: http://arxiv.org/abs/1512.03385.

[27] Geoffrey Hinton, Alex Krizhevsky, et al. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105. DOI: 10.1145/3065386.

[28] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[29] Peter J. Huber. "Robust Estimation of a Location Parameter". In: *Ann. Math. Statist.* 35.1 (Mar. 1964), pp. 73–101. DOI: 10.1214/aoms/1177703732.

[30] Mirco Hünnefeld. "Data Mining am IceCube-Neutrinodetektor. Ein Data-Mining-Ansatz zur Energierekonstruktion mittels k-NN-Regression". Bachelor Thesis. TU Dortmund, 2014. URL: http://app.tu-dortmund.de/cms/Medienpool/Bachelorarbeiten/Huennefeld_Mirco.pdf.

[31] **IceCube** Collaboration, Mirco Hünnefeld. "Deep Learning in Physics exemplified by the Reconstruction of Muon-Neutrino Events in IceCube". In: *Proceedings of the 35th International Cosmic Ray Conference*. Vol. 1057. PoS, 2017. URL: https://pos.sissa.it/cgi-bin/reader/contribution.cgi?id=PoS(ICRC2017)1057.

[32] IceCube/NSF. *IceCube Array*. Aug. 2017. URL: https://icecube.wisc.edu/gallery.

[33] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167.

[34] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[35] Günter Klambauer et al. "Self-Normalizing Neural Networks". In: *CoRR* abs/1706.02515 (2017). URL: http://arxiv.org/abs/1706.02515.

[36] J.-H. Koehne et al. "PROPOSAL: A tool for propagation of charged leptons". In: *Computer Physics Communications* 184.9 (2013), pp. 2070–2090. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2013.04.001.

[37] **IceCube** Collaboration, Claudio Kopper. "Observation of Astrophysical Neutrinos in Six Years of IceCube Data". In: vol. 981. PoS, 2017. URL: https://pos.sissa.it/cgi-bin/reader/contribution.cgi?id=PoS(ICRC2017)981.

[38] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *The CIFAR Dataset*. 2017. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[39] Y. Lecun. "Une procédure d'apprentissage pour réseau à seuil asymétrique". In: *Proceedings of Cognitiva 85, Paris* (1985), pp. 599–604.

[40] Yann A. LeCun et al. "Efficient backprop". In: *Neural Networks: Tricks of the Trade*. 2012, pp. 9–48. ISBN: 9783642352881. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 0028-0836. DOI: 10.1038/nature14539.

[42] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database*. 2017. URL: http://yann.lecun.com/exdb/mnist/.

[43] Yann LeCun et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems 2*. Morgan-Kaufmann, 1990, pp. 396–404. URL: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

[44] Honglak Lee et al. "Unsupervised learning of hierarchical representations with convolutional deep belief networks". In: *Communications of the ACM* 54.10 (Oct. 2011), pp. 95–103. ISSN: 0001-0782. DOI: 10.1145/2001269.2001295.

[45] Jan Lünemann. "Suche nach Dunkler Materie in Galaxien und Galaxienhaufen mit dem Neutrinoteleskop IceCube". PhD thesis. Mainz U., 2014. URL: http://d-nb.info/104967698X/34.

[46] A. r. Mohamed, G. E. Dahl, and G. Hinton. "Acoustic Modeling Using Deep Belief Networks". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1 (Jan. 2012), pp. 14–22. ISSN: 1558-7916. DOI: 10.1109/TASL.2011.2109382.

[47] Till Neunhöffer. "Estimating the angular resolution of tracks in neutrino telescopes based on a likelihood analysis". In: *Astroparticle Physics* 25.3 (2006), pp. 220–225. ISSN: 0927-6505. DOI: 10.1016/j.astropartphys.2006.01.002.

[48] Dirk Pandel. "Bestimmung von Wasser- und Detektorparametern und Rekonstruktion von Myonen bis 100 TeV mit dem Baikal-Neutrinoteleskop NT-72". PhD thesis. Humboldt-Universität zu Berlin, 1996. URL: https://docushare.icecube.wisc.edu/dsweb/Get/Document-74814/pandel-diplom.ps.gz.

[49] D.B. Parker, Massachusetts Institute of Technology, and Sloan School of Management. *Learning Logic: Casting the Cortex of the Human Brain in Silicon*. Technical report: Center for Computational Research in Economics and Management Science. Massachusetts Institute of Technology, Center for Computational Research in Economics and Management Science, 1985. URL: http://www.worldcat.org/title/learning-logic-casting-the-cortex-of-the-human-brain-in-silicon/oclc/17489314.

[50] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. "Large-scale deep unsupervised learning using graphics processors". In: *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*. Ed. by Andrea P. Danyluk, Léon Bottou, and Michael L. Littman. 2009, p. 110. URL: 10.1145/1553374.1553486.

[51] C. Radhakrishna Rao. "Information and the Accuracy Attainable in the Estimation of Statistical Parameters". In: *Breakthroughs in Statistics: Foundations and Basic Theory*. Ed. by Samuel Kotz and Norman L. Johnson. New York, NY: Springer New York, 1992, pp. 235–247. ISBN: 978-1-4612-0919-5. DOI: 10.1007/978-1-4612-0919-5_16.

[52] *Retraction of high energy neutrino candidate IceCube-171028*. 2017. URL: https://gcn.gsfc.nasa.gov/gcn3/22065.gcn3.

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1". In: ed. by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal

Representations by Error Propagation, pp. 318–362. ISBN: 0-262-68053-X. URL: http://dl.acm.org/citation.cfm?id=104279.104293.

[54] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[55] Kai Schatto. "Stacked searches for high-energy neutrinos from blazars with IceCube". PhD thesis. Mainz U., 2014-06-02. URL: http://inspirehep.net/record/1409209/files/doc.pdf.

[56] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[57] Amey Varangaonkar. *Top 10 Deep Learning Frameworks*. 2017. URL: https://datahub.packtpub.com/deep-learning/top-10-deep-learning-frameworks/.

[58] Markus Voge. *2014 TFT Proposal Request for a Level2 Online Muon Filter*. 2013. URL: https://docushare.icecube.wisc.edu/dsweb/Get/Document-67821/onlinel2proposal2014.pdf.

[59] Alexander Lyle Wallace. "Direction reconstruction of IceCube neutrino events with millipede". Master thesis. University of Adelaide, School of Physical Sciences, 2016. URL: https://digital.library.adelaide.edu.au/dspace/handle/2440/100191.

[60] Paul Werbos. "Beyond regression : new tools for prediction and analysis in the behavioral sciences". PhD thesis. Harvard University, 1974. URL: https://www.researchgate.net/publication/35657389_Beyond_regression_new_tools_for_prediction_and_analysis_in_the_behavioral_sciences.

Eidesstattliche Versicherung

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem Titel “Online Reconstruction of Muon-Neutrino Events in IceCube using Deep Learning Techniques” selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

Belehrung

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50 000.00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden (§ 63 Abs. 5 Hochschulgesetz –HG–).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z. B. die Software “turnitin”) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen.

Ort, Datum	Unterschrift