

Arbeit zur Erlangung des akademischen Grades
Bachelor of Science

Ordinal Classification with Neural Networks in DSEA

Nicolai Weitkemper
geboren in Soest

2022

Lehrstuhl für Experimentelle Physik V
Fakultät Physik
Technische Universität Dortmund

Erstgutachter: Prof. Dr. Dr. Wolfgang Rhode
Zweitgutachter: Prof. Dr. Johannes Albrecht
Abgabedatum: 30. September 2022

Abstract

The deconvolution of energy spectra is a common problem in neutrino astronomy. As an inverse problem, its solution requires special methods. The Dortmund Spectrum Estimation Algorithm (DSEA⁺) [9] solves the deconvolution problem by reinterpreting it as a multinomial classification problem and eliminates bias from the training data by iteratively re-weighting the samples. In the problem at hand, discretized neutrino energies are *ordinal* quantities, implying that misclassification can be of different severity. However, most classifiers do not respect this property. Previous works have focused either on respecting ordinality [23] or on using a neural network as a classifier [21]. This thesis aims to combine the advantages of both approaches: the flexibility of neural networks and the potential improvements in physical plausibility due to respecting ordinality. First, the Conditional Ordinal Regression for Neural Networks (CORN) framework [43] is adapted to work with DSEA⁺. The proposed method is then optimized and evaluated on simulated data from IceCube.

Kurzfassung

Die Entfaltung von Energiespektren ist ein gängiges Problem in der Neutrinoastronomie. Da es sich um ein inverses Problem handelt, erfordert seine Lösung spezielle Methoden. Der Dortmund Spectrum Estimation Algorithm (DSEA⁺) [9] löst das Entfaltungsproblem, indem er dieses als ein mehrklassiges Klassifikationsproblem uminterpretiert, und eliminiert den Bias aus den Trainingsdaten, indem er die Samples iterativ neu gewichtet. Im vorliegenden Problem sind diskretisierte Neutrinoenergien *ordinal*, was impliziert, dass Fehlklassifikationen unterschiedlich stark sein können. Allerdings respektieren die meisten Klassifizierer diese Eigenschaft nicht. Vorherige Arbeiten haben sich entweder auf die Berücksichtigung der Ordinalität [23] oder auf die Verwendung eines neuronalen Netzwerks als Klassifizierer [21] fokussiert. Diese Arbeit hat zum Ziel, die Vorteile beider Ansätze zu kombinieren: die Flexibilität von neuronalen Netzwerken und die potenzielle Verbesserung der physikalischen Plausibilität aufgrund der Berücksichtigung von Ordinalität. Zunächst wird Conditional Ordinal Regression for Neural Networks (CORN) [43] angepasst, um mit DSEA⁺ kompatibel zu sein. Die vorgeschlagene Methode wird dann optimiert und auf simulierten Daten von IceCube evaluiert.

Contents

1	Introduction	1
2	Neutrino Astronomy	2
2.1	Neutrinos	2
2.2	IceCube	3
3	Solving the Deconvolution Problem with DSEA⁺	5
3.1	The Deconvolution Problem	5
3.2	Dortmund Spectrum Estimation Algorithm	6
4	Ordinal Classification	9
4.1	On Nominal and Ordinal Data	9
4.2	CORN	10
5	Unfolding with CORN	12
5.1	Setup	12
5.2	Performance Metrics	15
5.3	Hyperparameters	16
5.4	Uncertainty and Results	22
5.5	Bias	24
6	Summary and Outlook	25
A	Appendix	26
A.1	Detector Signatures of Different Neutrino Flavors	26
A.2	DSEA ⁺ : Complete Algorithm	27
A.3	From Threshold to Per-Class Probabilities	28
A.4	Monte Carlo Data Set	29
A.5	Additional Hyperparameter Plots	30
A.6	Bootstrap Distributions	31
A.7	Links etc.	32
	Bibliography	33

1 Introduction

Astroparticle physics is a relatively new field of physics that explores the universe with messenger particles. They carry information about the production processes and the environment they were created in, which can be obtained upon their detection. Neutrinos are especially valuable messenger particles because their propagation path is not affected by electromagnetic fields, and they can propagate long distances without interacting with matter.

IceCube is a detector for such neutrinos, which is located at the South Pole. The Antarctic ice is used as detector material and is sensitive to high-energy neutrinos. One goal of analyses with IceCube is to obtain a neutrino energy spectrum.

The Dortmund Spectrum Estimation Algorithm (DSEA⁺) [9] is a method to reconstruct the (discretized) neutrino energy spectrum from measured data of IceCube. While classifiers such as random forests have been successfully applied to IceCube data using DSEA⁺ [22], the possibilities of neural networks and deep learning are still largely unexplored. Haefs [21] combined DSEA⁺ with a neural network for the first time. Jäkel [23], on the other hand, focused on the ordinality of the discretized energy spectrum by employing *LogisticAT* [32], a classifier that considers the ordering of energy bins in its loss function. This thesis aims to combine the advantages of both approaches: the flexibility of neural networks and the potential improvements in physical plausibility due to respecting ordinality. The Conditional Ordinal Regression for Neural Networks (CORN) framework [43] allows neural networks to respect ordinality by using a special loss function and activation function in the output layer. It is adapted to work with DSEA⁺ by converting from conditional confidences to per-class confidences and adding support for sample weights. This combination of CORN and DSEA⁺ is then optimized and evaluated on simulated data from IceCube.

This paper is organized as follows: Chapter 2 briefly introduces IceCube and important concepts of neutrino astronomy. Chapter 3 describes the deconvolution problem and the DSEA⁺ approach to solving it. Chapter 4 explains the benefits of ordinal classification and introduces the CORN framework. Chapter 5 describes the setup for hyperparameter searches and evaluates the performance of the optimized model. Chapter 6 summarizes the thesis and discusses future work.

2 Neutrino Astronomy

This chapter provides an introduction to the field of neutrino astronomy and describes the IceCube Neutrino Observatory, the experimental background of this thesis.

2.1 Neutrinos

Neutrinos are elementary particles with no electric charge and small mass. Their low interactivity makes them difficult to detect, but is also the reason why they are valuable messenger particles for astroparticle physics: Since they are not affected by the electromagnetic force, cosmic magnetic fields have no effect on their propagation paths [25]; neither does the interstellar medium absorb them in significant amounts. Together with the energy of the neutrino, the direction of propagation is used to determine the location and properties of the source [25].

Neutrinos have various astronomical sources, many of which are not experimentally confirmed yet. This includes the cosmic neutrino background (CNB) [18] which originates from the Big Bang as well as various sources of cosmic ray acceleration (and therefore neutrino emission) [4], such as supernova remnant (SNR) shocks, active galactic nuclei (AGN), jets, starburst galaxies, and gamma-ray bursts. Neutrinos are also produced in the Sun and in the Earth's atmosphere as well as in nuclear reactors. They cover a vast range of energies, from μeV up to PeV , depending on the source. Figure 2.1 shows the flux spectrum of neutrinos from different sources.

While current models of astrophysical neutrino sources predict a flavor ratio of $\phi(\nu_e) : \phi(\nu_\mu) : \phi(\nu_\tau) = 1 : 2 : 0$ (assuming charged pions decays are the dominant mechanism for neutrino production), the observed ratio on Earth is $1 : 1 : 1$ [5]. This discrepancy is explained by neutrino oscillations [5], which are a consequence of the fact that neutrinos have mass. Albeit being very light (the current lowest upper limit on the Majorana mass being 0.06 eV to 0.161 eV [20]), it allows for oscillations between the different flavors, given the large distances that cosmic neutrinos travel.

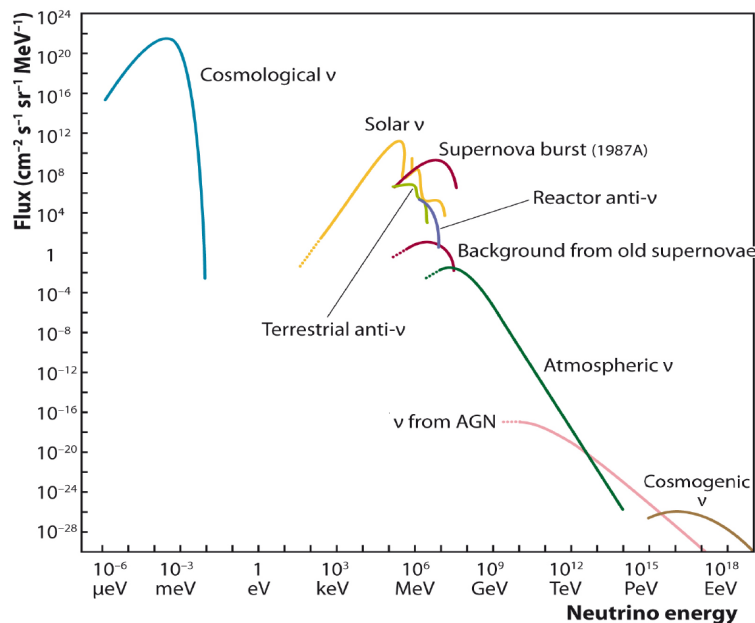


Figure 2.1: Measured and expected fluxes of natural and reactor neutrinos as a function of their energy [44].

2.2 IceCube

The IceCube Neutrino Observatory is located close to the geographic South Pole in proximity to the Amundsen-Scott South Pole Station. It utilizes the optically clear Antarctic ice as detector material, with a total detector volume of 1 km^3 [2]. The detector is composed of 86 strings, each consisting of 60 digital optical modules (DOMs), which are positioned 17 m apart along the string at depths ranging from 1450 m to 2450 m below the surface [2]. Each of the 5160 DOMs is equipped with a sensitive photomultiplier tube (PMT), which detects the Cherenkov light emitted by charged particles that interact with the ice. Figure 2.2 shows a schematic of the IceCube detector.

Neutrinos interact with matter via the weak interaction. In order to compensate for the low cross-section of the weak interaction, the effective detector volume is maximized by utilizing existing naturally occurring detector materials, such as the Earth's atmosphere, the sea, or the ice in the Antarctica. When a neutrino interacts with a nucleus in the ice, it produces a charged lepton and a neutrino. The charged lepton then produces a Cherenkov light cone as it propagates through the ice with greater speed than the speed of light in the ice. The light is detected by the PMTs and the direction of the neutrino can be reconstructed from the position

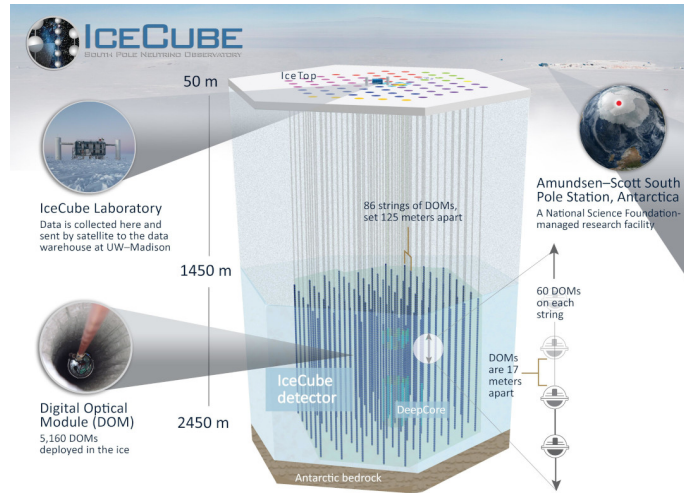


Figure 2.2: Infographic of the IceCube Neutrino Observatory [13]. Each dot represents a DOM.

of the DOMs that detected the light. The intensity of the light and the number of PMTs that detected it are used to determine the energy of the neutrino. IceCube is sensitive to the approximate neutrino energy range from GeV to PeV [2] and therefore primarily to atmospheric and AGN neutrinos. Higher energies require even larger detector volumes.

Depending on the flavor of the leptons, different types of interactions occur, allowing to determine the flavor of the neutrino in turn. Muon neutrinos are detected as *tracks* [2] which originate from a charged-current interaction of a high-energy muon neutrino with a nucleus. They have a good angular resolution because muons typically fall under the Cherenkov limit as soon as they scatter. Electron neutrinos, on the other hand, are detected as *cascades* [2]. Contrary to muons, electrons typically scatter several times before falling below the Cherenkov threshold, therefore prohibiting the reconstruction of the neutrino direction. On the other hand, they are more likely to be fully contained in the detector, which makes them useful for energy studies [2]. A third type of signature is the *double bang* [28] or *lollipop* [5], which very high energy tau neutrinos could produce. It has not been observed so far [3]. Examples of interactions and corresponding detector patterns are shown in Figure A.1.

Not only neutrinos can leave traces in the detector, but also atmospheric muons. For this reason, the detection is mostly limited to *up-going* events [2], because muons – in contrast to neutrinos – are mostly blocked by the Earth’s mass.

3 Solving the Deconvolution Problem with DSEA⁺

In this chapter, a formal definition of inverse problems and the deconvolution problem in particular is given. The DSEA⁺ algorithm is then introduced as a solution thereof.

3.1 The Deconvolution Problem

Inverse problems are omnipresent in modern physics. They occur whenever a physical quantity is measured indirectly. For example, the intensity of light can be measured by a photodetector, which converts the light into an electrical signal. However, this conversion is not perfect: A real detector has limited acceptance and resolution and the signal is subject to noise. The deconvolution problem – as a special case of inverse problems – is to reconstruct the distribution of the physical quantity of interest from the indirect measurements.

Mathematically, a set of single physical quantities x limited to an arbitrary range $a \leq x \leq b$ (such as the energy of a neutrino) can be interpreted as samples from an event spectrum $f(x)$. Given the measured distribution $g(y)$ and a *response function* $A(x, y)$, which describes the detector, the deconvolution problem is to find a distribution $f(x)$ that satisfies the Fredholm integral equation of the first kind [19]:

$$\int_a^b A(x, y) f(x) dx = g(y). \quad (3.1)$$

3.1.1 Discretization

In the context of physical measurements, the integral equation is discretized to account for the finite number of samples. The continuous distribution functions $f(x)$ and $g(y)$ are replaced by vectors \vec{f} and \vec{g} , and the kernel (response function) $A(x, y)$ by a *transfer matrix* \mathbf{A} . The discretized deconvolution problem is then given by

$$\mathbf{A}\vec{f} = \vec{g}. \quad (3.2)$$

In practice, the transfer matrix can be approximated by Monte Carlo simulations of the detector, where both the true and the measured quantities are known. Given an actual set of measurements \vec{g} , the deconvolution problem can then be solved by inverting the transfer matrix:

$$\vec{f} = \mathbf{A}^{-1}\vec{g}. \quad (3.3)$$

However, as is common with inverse problems, the matrix is usually *ill-conditioned*, leading to numerical instabilities and oscillations in the solution. One approach to overcome this problem is *regularization*. It allows for better results at the cost of introducing additional a priori assumptions and parameters (*bias-variance trade-off*) [27]. A common regularization technique is to penalize the second derivative of the solution [45], which is known as *Tikhonov regularization* [46]. This incentivizes the solution to be smooth, which is often a reasonable assumption for distributions of physical quantities.

3.2 Dortmund Spectrum Estimation Algorithm

The Dortmund Spectrum Estimation Algorithm (DSEA⁺) [9] is an iterative method for solving the previously stated deconvolution problem. It improves upon its predecessor Dortmund Spectrum Estimation Algorithm (DSEA) [41] by correcting the re-weighting of training examples and giving more control over the speed of convergence. A formal definition of the DSEA⁺ algorithm is given in section A.2.

3.2.1 Deconvolution as a Classification Task

DSEA⁺ makes use of classifiers to solve the deconvolution problem. This requires the deconvolution problem to be discretized (see subsection 3.1.1) and reformulated as a multinomial classification task.

Any classifier that outputs probabilities for each class can be used with DSEA⁺. This is an advantage, as the choice of a classifier can be tailored to the specific problem at hand. Contrary to other algorithms like Time-dependent Regularized Unfolding for Economics and Engineerings (TRUEE) / Regularized Unfolding (RUN) [30] or Iterative Bayesian Unfolding (IBU) [14, 15], no restrictions on the input data are imposed.

Furthermore, DSEA⁺ transparently provides the contributions of individual observations to the deconvolved spectrum. This not only gives deeper insight into the performance of the algorithm, but also allows for time-dependent deconvolution [8].

3.2.2 Procedure

Initialization

Since no prior knowledge about the true spectrum is available, the initial spectrum is chosen to be uniform. Given I bins, each bin is initialized to

$$\hat{\mathbf{f}}_i^{(0)} = \frac{1}{I} \quad \forall i. \quad (3.4)$$

The initial weights are then determined as in Equation 3.6.

Iteration

First, the classifier is trained on the training data to predict the class of one sample at a time, where each sample is weighted according to its true class. Second, the classifier is used to predict the class of each sample in the observed (/test) data. Third, the predicted classes are used to get an updated estimate of the spectrum and, consequently, updated weights for the next iteration. The new spectrum is determined by the sum of the confidences of all events. For each energy bin with index i , this can be written as

$$\hat{\mathbf{f}}_i = \frac{1}{N} \sum_{n=1}^N \hat{c}_{i,n}, \quad (3.5)$$

where N is the number of events in the observation data set, k is the current iteration number, and $\hat{c}_{i,n}$ is the confidence that event n belongs to class i . The factor $1/N$ is introduced to normalize the spectrum to a true probability density distribution. The weights of the training samples are then updated according to the new spectrum $\hat{\mathbf{f}}_i^{(k)}$. In DSEA⁺, the reconstructed spectrum is divided by the training spectrum in order to mitigate the impact of the training spectrum on the deconvolution result. A more detailed reasoning is given in [8]. The updated weights are given by

$$w_i^{(k+1)} = \frac{\hat{\mathbf{f}}_i^{(k)}}{\mathbf{f}_i^{\text{train}}}, \quad (3.6)$$

where $w_i^{(k+1)}$ is the weight applied to training samples with true bin i in iteration $k+1$, and $\mathbf{f}_i^{\text{train}}$ is the value of the i -th bin in the training spectrum.

The iterative procedure is repeated with the updated weights until convergence (see subsection 3.2.3) or, in case of fixed step sizes, a maximum number of iterations is reached. The final deconvolution result is the spectrum obtained in the last iteration.

3.2.3 Step Size Functions

DSEA⁺ introduces the concept of a step size α , which allows the user to control the speed of convergence, which in turn has a significant impact on the quality of the result.

A *step* $p_i^{(k)}$ is the difference between the current and previous deconvolution result:

$$p_i^{(k)} = \hat{\mathbf{f}}_i^{(k)} - \hat{\mathbf{f}}_i^{(k-1)}. \quad (3.7)$$

Instead of updating the spectrum with the current deconvolution result $\hat{\mathbf{f}}_i^{(k)}$ directly, the step is multiplied with the step size and added to the previous deconvolution result:

$$\hat{\mathbf{f}}_i^{(k)+} = \hat{\mathbf{f}}_i^{(k-1)} + \alpha \cdot p_i^{(k)}. \quad (3.8)$$

This improved estimate $\hat{\mathbf{f}}_i^{(k)+}$ is then considered instead of $\hat{\mathbf{f}}_i^{(k)}$.

While the original DSEA algorithm uses a fixed step size of $\alpha = 1$, DSEA⁺ allows arbitrary constants $\alpha > 0$ or functions of the iteration number k . Commonly used step size functions include multiplicative decay $\alpha^{(k)} = k^{\eta-1}$ and exponential decay $\alpha^{(k)} = \eta^{(k-1)}$, each with a *decay rate* $0 < \eta < 1$. These decaying step sizes ensure that the algorithm converges, decreasing the importance of the maximum number of iterations K , while enabling the use of a stopping criterion: When the χ^2 distance becomes smaller than ϵ , convergence is assumed and the training is stopped. In this work, the probabilistic symmetric χ^2 distance [11, 8]

$$\chi_{\text{Sym}}^2(\hat{\mathbf{f}}, \mathbf{f}) = 2 \cdot \sum_{i=1}^I \frac{(\hat{\mathbf{f}}_i - \mathbf{f}_i)^2}{\hat{\mathbf{f}}_i + \mathbf{f}_i} \quad (3.9)$$

is used.

The utilization of *adaptive step sizes* [8] can further improve the convergence of the algorithm by choosing an optimal step size for each iteration. This is achieved by searching along the direction of the step $p_i^{(k)}$ for the step size $\alpha \geq 0$ which minimizes the RUN [30] loss function. In the process, the method discretizes the training data using a decision tree, thus adding a hyperparameter J that controls the number of its leaves.

4 Ordinal Classification

DSEA⁺ can solve the deconvolution problem given any capable classifier. However, classifiers usually do not take into account *ordinality*, which contradicts the physical meaning of the unfolded spectrum. This chapter clarifies the concept of ordinality and introduces the CORN algorithm, which applies it to neural networks.

4.1 On Nominal and Ordinal Data

Nominal data can be thought of as a set of distinct categories (*classes*) with no inherent ordering. The most common classifiers, such as neural networks with softmax output (used in [21]), and random forests (used in [22]), treat data as nominal. Ordinal data, on the other hand, has an inherent ordering. The distance between two categories is not necessarily the same. Instead of classes, the different categories are commonly referred to as *ranks*. There are classifiers that can handle ordinal data, such as *LogisticAT* [32, 23] and CORN (see section 4.2).

In the context of the problem at hand, the neutrino energies are ordinal, since they are discretized into bins, while the ordering by energy remains intact. A classifier can then either treat the data as nominal, disregarding the ordering, or as ordinal, taking the ordering into account. Disregarding the ordering is especially problematic for the reconstruction of single events or spectra that depend on an additional parameter (such as time [22]), because in both cases, not only the unfolded spectrum (as a normalized sum of confidence distributions), but also the confidence distribution of single events is considered. With nominal classification, there is no incentive for the classifier to return a unimodal confidence distribution. An event could therefore yield high confidences for both very low and very high energies, which is not physically plausible.

4.2 CORN

Conditional Ordinal Regression for Neural Networks (CORN) [43] is a framework for ordinal classification in neural networks. It is based on the ideas of binary subtasks and conditional probabilities and improves upon its direct predecessor Consistent Rank Logits (CORAL) [10] as well as the approach by Niu et al. [34].

4.2.1 Method

Let $D = \{\mathbf{x}^{[i]}, y^{[i]}\}_{i=1}^N$ denote a data set of N training examples, where $\mathbf{x}^{[i]}$ is the i -th example and $y^{[i]}$ is its class label. Since the class labels are ordinal, they are referred to as *rank* labels. Each rank label is an element of the set of all ranks $\{r_1, r_2, \dots, r_K\}$, where K is the number of ranks and $r_1 < r_2 < \dots < r_K$.

For every rank label $y^{[i]}$, $K - 1$ subtasks are created. Each subtask $y_k^{[i]} \in \{0, 1\}$ is a binary classification task, where $y_k^{[i]} = 1$ if $y^{[i]} > r_k$ (in words: $y^{[i]}$ exceeds rank r_k) and $y_k^{[i]} = 0$ otherwise. This method of creating binary subtasks is referred to as *extended binary classification* [29].

Given a test example $\mathbf{x}^{[i]}$ and probability predictions $f_k(\mathbf{x}^{[i]}) \in [0, 1]$ for each subtask k , the *rank index* $q \in \{1, 2, \dots, K\}$ is computed as

$$q = \sum_{k=1}^{K-1} \mathbb{1} \{f_k(\mathbf{x}^{[i]}) > 0.5\} . \quad (4.1)$$

The predicted rank label is then obtained via $h(\mathbf{x}^{[i]}) = r_q$, where $h : \mathcal{X} \rightarrow \mathcal{Y}$ is the mapping from input space \mathcal{X} to output space \mathcal{Y} which minimizes the CORN loss function.

Rank monotony describes a desirable property of the rank labels, whereby the probability of exceeding a rank r_k is always greater than or equal to the probability of exceeding a higher rank r_{k+1} . While not strictly necessary for the computation of rank indices q (or ranks r_q), it is intuitively clear that imposing such a restriction could improve the quality of predictions. CORN ensures rank monotony $f_1(\mathbf{x}^{[i]}) \leq f_2(\mathbf{x}^{[i]}) \leq \dots \leq f_{K-1}(\mathbf{x}^{[i]})$ by applying the chain rule of probability

$$\hat{P}(y^{[i]} > r_k) = \prod_{j=1}^k f_j(\mathbf{x}^{[i]}) \quad (4.2)$$

to the conditional probabilities

$$f_k(\mathbf{x}^{[i]}) = \hat{P}(y^{[i]} > r_k \mid y^{[i]} > r_{k-1}) . \quad (4.3)$$

CORN also provides the loss function. The conditional nature of the predictions $f_k(\mathbf{x}^{[i]})$ (see Equation 4.3) is respected by splitting the training data into conditional training subsets S_k each time the loss function is computed:

$$\begin{aligned} S_1 &: \text{all } \{(\mathbf{x}^{[i]}, y^{[i]})\} \text{ for } i \in \{1, \dots, N\}, \\ S_2 &: \{(\mathbf{x}^{[i]}, y^{[i]}) \mid y^{[i]} > r_1\}, \\ &\dots \\ S_{K-1} &: \{(\mathbf{x}^{[i]}, y^{[i]}) \mid y^{[i]} > r_{K-2}\}. \end{aligned}$$

In words, for $k > 1$, S_k contains all training examples for which the rank label exceeds rank r_{k-1} . The first subset S_1 contains all training examples.

The definition of the loss function is shown in Equation 4.4. Here, $|S_k|$ denotes the number of elements in the subset S_k .

$$\begin{aligned} L(\mathbf{X}, \mathbf{y}) = & -\frac{1}{\sum_{j=1}^{K-1} |S_j|} \sum_{j=1}^{K-1} \sum_{i=1}^{|S_j|} \left[\log(f_j(\mathbf{x}^{[i]})) \cdot \mathbb{1}\{y^{[i]} > r_j\} \right. \\ & \left. + \log(1 - f_j(\mathbf{x}^{[i]})) \cdot \mathbb{1}\{y^{[i]} \leq r_j\} \right] \quad (4.4) \end{aligned}$$

4.2.2 Obtaining Probabilities from CORN

As explained in subsection 4.2.1, CORN only returns a single predicted rank r_q for a given input \mathbf{x} . In contrast, DSEA⁺ is strongly coupled to the idea of per-class probabilities (confidences). Therefore, a conversion is necessary. Under the assumption that all energies belong to one of the ranks (bins), such a conversion is realizable.

As an example, given four rank indices $q \in \{0, 1, 2, 3\}$, the conditional probabilities are

$$\begin{aligned} \hat{P}(q = 0) &= 1 - \hat{P}(q > 0) \\ \hat{P}(q = 1) &= \hat{P}(q > 1) - \hat{P}(q > 1) \\ \hat{P}(q = 2) &= \hat{P}(q > 1) - \hat{P}(q > 2) \\ \hat{P}(q = 3) &= \hat{P}(q > 2) \end{aligned}$$

A more detailed explanation is given in section A.3.

5 Unfolding with CORN

This chapter describes the performance and optimization tests of the CORN algorithm in DSEA⁺.

5.1 Setup

In order to evaluate the performance of CORN in DSEA⁺, a data set has to be prepared and hyperparameters need to be chosen. The setup is largely based on previous works [23, 21] to ensure comparability of the results as much as possible.

5.1.1 Monte Carlo Data Set and Preprocessing

The unmodified data set *11374* [12] consists of about 13 million Monte Carlo simulated up-going muon neutrino events, with a weighted E^{-2} energy spectrum ranging from 10^2 GeV to 10^8 GeV. The energies are stored under the key `MCPPrimary.energy`. See Figure A.2 for a histogram of the data.

To ensure comparability to [23] and [21], only the first 500 000 events of the data set are considered. This also allows for more thorough hyperparameter optimization, which would otherwise be limited by the available computational resources as well as the timeframe of this thesis. Unless otherwise stated, 90 % of the data is used for training, while the remaining 10 % is used for evaluation. No separate validation set is used.

Because unfolding is highly dependent on the selection of features [23] and training is negatively affected by the presence of irrelevant ones [16], not all available features are considered. Jäkel [23] has employed the *mRMR* (Minimum Redundancy Maximum Relevance) algorithm [38] to select the 12 most relevant features. The algorithm takes into account both the *relevance* of a feature to the target variable, measured by their correlation, and the *redundancy* of a feature to other features. This way, the *minimal-optimal* set of features is selected, in contrast to the *all-relevant* set of features, which would also contain redundant features. A list of said features is provided in Table 5.1. They are reused in this thesis.

```

SplineMPEDirectHitsICE.n_dir_doms
VariousVariables.Cone_Angle
SplineMPECramerRaoParams.variance_theta
Borderness.Q_ratio_in_border
SplineMPETruncatedEnergy_SPICEMie_BINS_MuEres.value
SplineMPETruncatedEnergy_SPICEMie_DOMS_Neutrino.energy
SplineMPEDirectHitsICB.n_late_doms
Dustyness.n_doms_in_dust
LineFitGeoSplit1Params.n_hits
SplineMPEDirectHitsICC.dir_track_hit_distribution_smoothness
SPEFit2GeoSplit1BayesianFitParams.logl
SplineMPECharacteristicsIC.avg_dom_dist_q_tot_dom

```

Table 5.1: 12 best features according to the *mRMR* algorithm [23].

It has been shown that none of the selected features are normally distributed [23]. In accordance with [23], the features are therefore transformed using the *Yeo-Johnson* transformation [48], a power transformation which reduces skewness. Additionally, zero-mean, unit-variance normalization is applied to all features.

As described in section 3.2, DSEA⁺ requires discrete energy classes. The target variable `MCPprimary.energy` is therefore discretized into 10 bins (in accordance with [21]). Contrary to [23] and [21], under- and overflow bins are added in order to allow for the application to real data. The lower limit of the overflow bin is chosen so that it contains a similar number of events as the previous bin, ensuring sufficient statistics. A lower energy limit of 10^5 GeV (in accordance with [21]) was found to satisfy this requirement. The underflow bin is assigned the energy range from 10^2 GeV to $10^{2.1}$ GeV because the data set does not contain any events with exceptionally low energies below 10^2 GeV. This way, the event count in the underflow bin is similar to the neighboring bin. The remaining 8 bins are spaced logarithmically between the under- and overflow bins so that the entire energy range of the Monte Carlo data set is covered. A histogram utilizing the aforementioned bins is shown in Figure 5.1.

5.1.2 Neural Network and DSEA⁺

A PyTorch [35] implementation of the CORN method as well as several examples demonstrating its application on different data sets are provided by [43]. This work makes use of said implementation of CORN and hence the PyTorch framework. Additionally, PyTorch Lightning [39], TorchMetrics [33], and scikit-learn [37] are used.

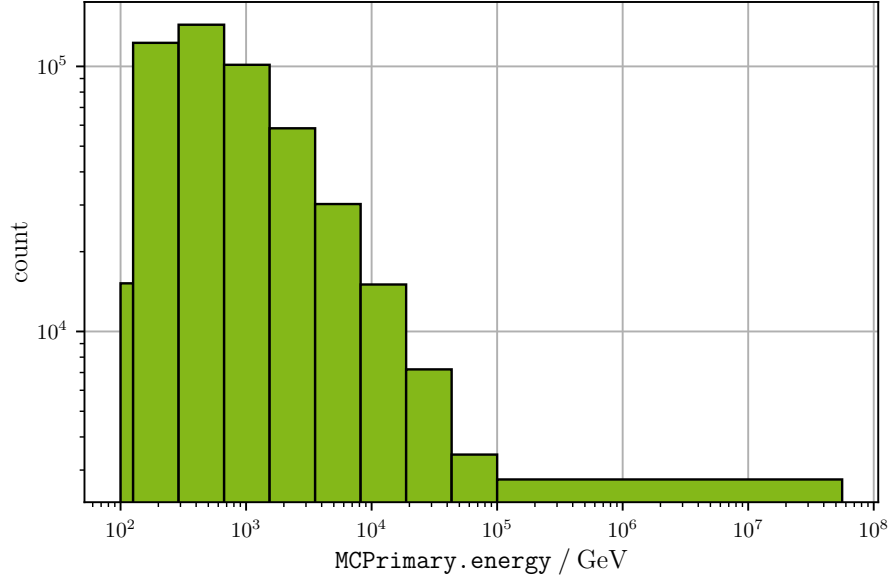


Figure 5.1: Energy spectrum of the first 500 000 events in the Monte Carlo data set using the discretized energy ranges as bins.

The neural network consists of 4 fully connected hidden layers. The input layer has 12 neurons, corresponding to the number of features, while the output layer has 9 neurons, corresponding to the number of binary classification subtasks, i.e. the number of bins minus one. In total, the neural network has about 6100 trainable parameters. The number of neurons and the activation function per layer are shown in Table 5.2. In order to retrieve probabilities from the output neurons, a modified version of the CORN-provided function `corn_label_from_logits(logits)` is used, which implements the conversion from threshold- to per-class probabilities outlined in subsection 4.2.2.

Adaptive Moment Estimation (Adam) [26] is used as the optimizer. It minimizes the loss function provided by CORN (see Equation 4.4).

The neural network keeps its weights between DSEA⁺ iterations. In theory, this could improve its performance, analogous to *fine-tuning* a pre-trained model. However, [21] found no significant effect on the performance.

For this work, the Python implementation of DSEA⁺ [7] is used, which expects a *scikit-learn* classifier. In order to interface with this library, a wrapper class is implemented, which exposes a constructor as well as the needed methods `fit(X, y, sample_weight)` and `predict_proba(X)`.

neurons	activation function
12	–
120	leaky ReLU
240	leaky ReLU
120	leaky ReLU
12	leaky ReLU
9	leaky ReLU

Table 5.2: Shape and activation functions of the neural network. The number of neurons in the input and output layers is determined by the number of features and bins, respectively. Each activation function precedes the neurons in the same row.

5.2 Performance Metrics

In order to evaluate the performance of the models and to compare them to prior works, several metrics are used. For the calculation of all metrics that are mentioned here, scikit-learn [37] is used.

5.2.1 Accuracy

The *accuracy* [1] is the fraction of correctly classified events to the total number of events. It is a common metric for classification tasks, but it is not ideal for ordinal classification since it does not take into account the ordering of the classes. For example, the metric is the same for a misclassification by one rank and a misclassification by two ranks. Nonetheless, it gives an indication of the overall performance of the model.

5.2.2 Mean Absolute Error

The *mean absolute error* (MAE) [47] is a metric that is commonly used for regression tasks. It is defined as

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y^{[i]} - \hat{y}^{[i]}| \quad (5.1)$$

where N is the number of events, $y^{[i]}$ is the true value of the i -th event, and $\hat{y}^{[i]}$ is the predicted value of the i -th event.

Since the absolute value of the error is considered, overestimation and underestimation are treated equally and do not cancel each other out. In contrast to the *root mean squared error* (RMSE), the MAE is not especially sensitive to outliers and has a more natural interpretation [47].

5.2.3 Wasserstein Distance

The two previous metrics were based on single predictions for each event. They disregard both the confidences of the predictions, considering only the prediction with the highest confidence, and the spectrum, which results from summing up the confidences over all events.

In contrast, the *Wasserstein distance* [40] compares the unfolded spectrum to the true spectrum. It is also known as *earth mover's distance* (EMD), hinting at the analogy of moving earth to transform one distribution into another, where the cost is given as the product of the distance and the amount of earth moved.

Mathematically, the Wasserstein distance (of first kind) can be defined as

$$\text{WD}(\mathbf{p}, \mathbf{q}) = \inf_{\pi \in \Pi(\mathbf{p}, \mathbf{q})} \int_{\mathbb{R}^2} |x - y| d\pi(x, y) \quad (5.2)$$

where \mathbf{p} and \mathbf{q} are the probability distributions subject to comparison, $\Pi(\mathbf{p}, \mathbf{q})$ is the set of all probability distributions on \mathbb{R}^2 , and π is a probability distribution on \mathbb{R}^2 .

5.3 Hyperparameters

Hyperparameters are parameters that are not learned from the data, but adjusted by the user. In order to find the optimal hyperparameters, the unfolding procedure is repeated for different values of the hyperparameters.

A starting point is determined by Bayesian optimization of the hyperparameters (see subsection 5.3.2). Then, individual grid searches are performed for the hyperparameters of interest. In each case, 10-fold cross-validation is utilized to reduce the variance of single data points (see subsection 5.3.1).

The adaptive step size method of DSEA⁺ (see section 3.2.3) is used because it performs well with all reasonable convergence thresholds. It eliminates the need to optimize for step size related hyperparameters, such as exponential vs. multiplicative

decay or the initial step size, apart from the J -factor, which only has a minor effect on the results, and provides accurate results after a few iterations [8].

Since Adam maintains separate learning rates for each parameter, optimization for the learning rate is not essential. This is verified by the initial Bayesian optimization (subsection 5.3.2), where the results have no significant correlation with the learning rate.

5.3.1 Cross-Validation

Cross-validation [6] is a method to evaluate the performance of a model on unseen data without having to reserve a part of the data for testing. The data is split into k folds of equal size. The model is then trained on $k - 1$ folds, and evaluated on the remaining fold. This is repeated k times, each time using a different fold for evaluation. The average of the results is then used as the performance metric.

In this work, cross-validation is used primarily to get more meaningful performance metrics for each hyperparameter setting, since individual runs have a high variance. The number of folds is set to $k = 10$, striking a balance between the size of the data set, the statistical uncertainty, and the computational cost.

5.3.2 Initial Bayesian Search

Since a grid search is computationally expensive, especially for a large number of hyperparameters, a Bayesian optimization search [36] is used to find a good starting point for the grid search. Contrary to a random search, the results of previous runs are taken into account by building a probabilistic model of the objective function (in this case, the Wasserstein distance), which allows the search to focus on promising areas of the hyperparameter space.

Figure 5.2 shows the results of the Bayesian hyperparameter search. Based on these results, the initial hyperparameters are set to the values shown in Table 5.3.

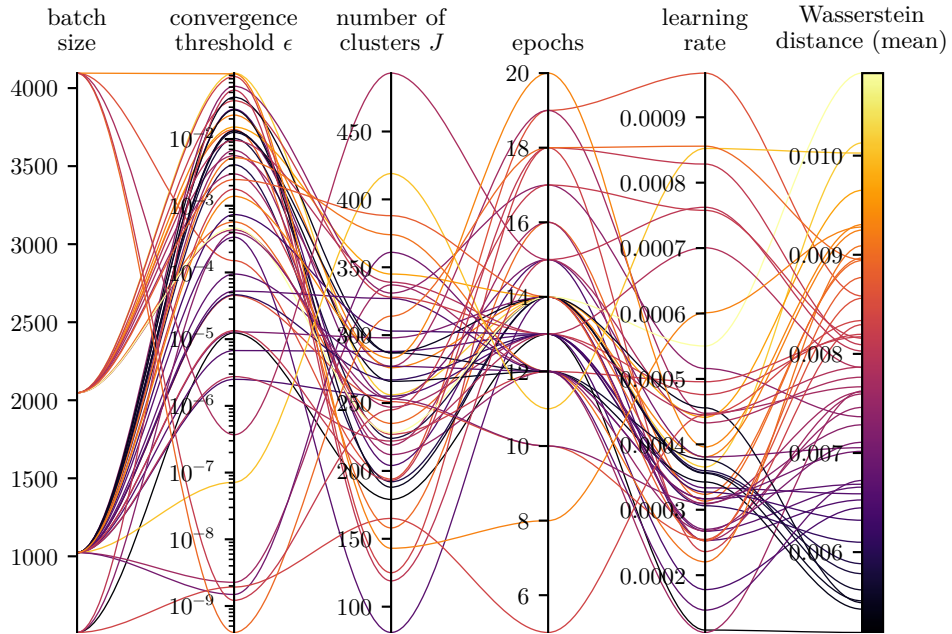


Figure 5.2: Parallel coordinates plot of the initial Bayesian hyperparameter search. For clarity, only the mean of each 10-fold cross-validation run is shown.

hyperparameter	value
batch size	1024
convergence threshold ϵ	10^{-2}
clusters J	250
epochs	12
learning rate	0.0004

Table 5.3: Optimal hyperparameters as determined by a Bayesian optimization search.

5.3.3 Batch Size

The *batch size* determines the number of events used for each training step. While larger batch sizes increase the training speed on optimized hardware, the performance of the model can be negatively affected [24].

The box plot in Figure 5.3 shows the results of a grid search for the batch size. A batch size of 4096 is chosen as the optimal value, not only because it has the smallest Wasserstein distance and low variance, but also because it allows for a faster training time compared to smaller values.

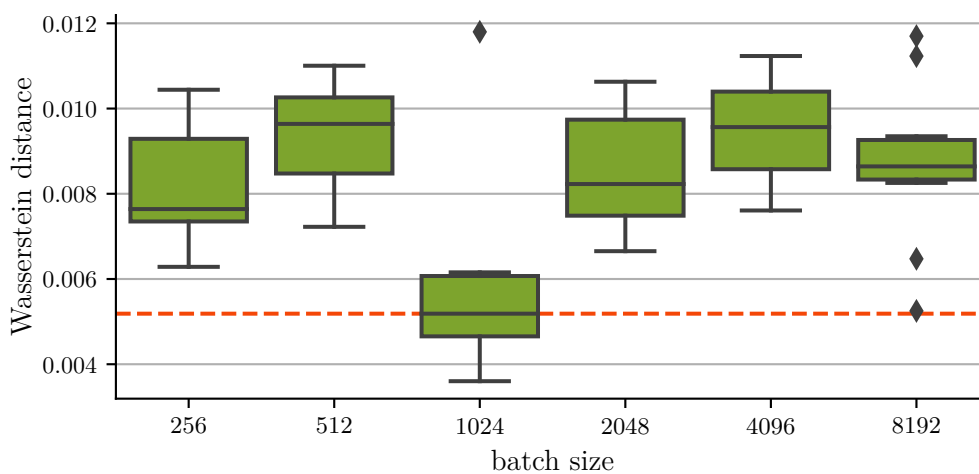


Figure 5.3: Box plot of the performance of the model for different batch sizes. The performance is measured using the *Wasserstein distance*. The box shows the 25th to 75th percentile, the center line denotes the median, and the whiskers show the minimum and maximum, apart from outliers, which are shown as dots. A dashed orange line indicates the best median value.

5.3.4 Convergence Threshold

The box plot in Figure 5.4 shows the results of a grid search for the convergence threshold (as described in subsection 3.2.3). For larger values, the performance tends to improve, until approximately $\epsilon > 0.05$. This is because a high convergence threshold can be thought of as a form of *implicit* regularization, analogous to early stopping, contrary to *explicit* regularization, as described in section 3.1.1. Selecting even higher values of ϵ (not shown in the plot) stops DSEA⁺ after the first iteration, which results in particularly poor performance. $\epsilon = 0.025$ is chosen as the optimal value.

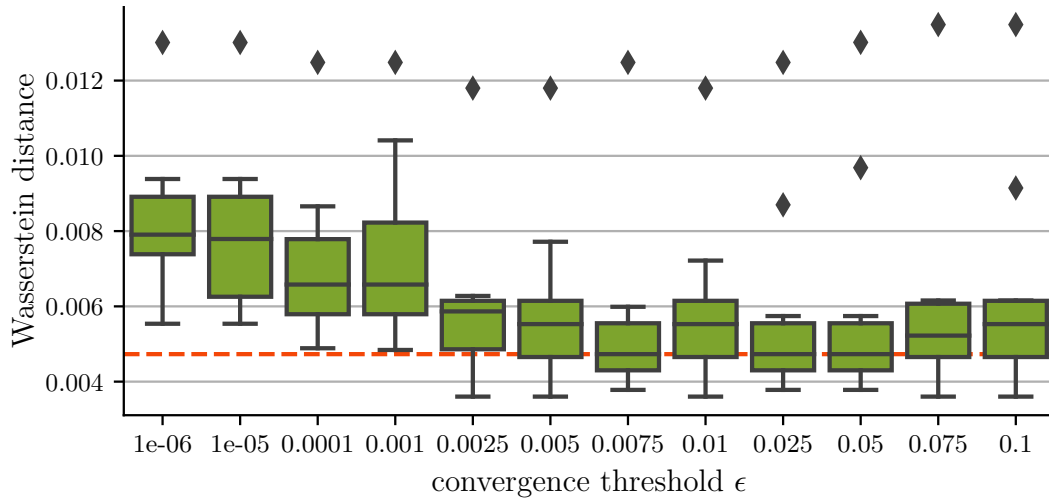


Figure 5.4: Box plot of the Wasserstein distance for different convergence thresholds. Note that the abscissa is not uniformly spaced.

5.3.5 Number of Clusters

The adaptive step size function, which has been explained in section 3.2.3, internally relies on clustering the data into J clusters. The number of clusters J is therefore a hyperparameter of the algorithm. Large values of J have previously been shown to lead to slightly better results [8]. Figure 5.5 confirms this, although the median does not decrease monotonically with J . The optimal value is 200, which has a slightly better performance than 500.

5.3.6 Number of Epochs

While a higher number of epochs typically increases the model’s performance on the training data, it also increases the risk of overfitting. Figure 5.6 visualizes a hyperparameter search for the number of epochs per DSEA⁺ iteration. The results show that the model’s performance on the training data is best for 12 epochs. Figure A.3 shows that the accuracy is not negatively affected by a higher number of epochs, indicating that the model is not overfitting.

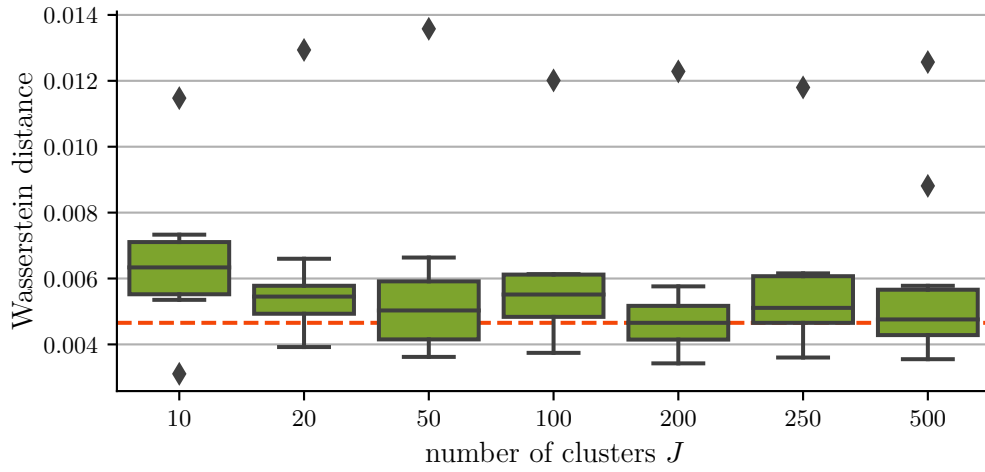


Figure 5.5: Box plot of the Wasserstein distance for different numbers of clusters.

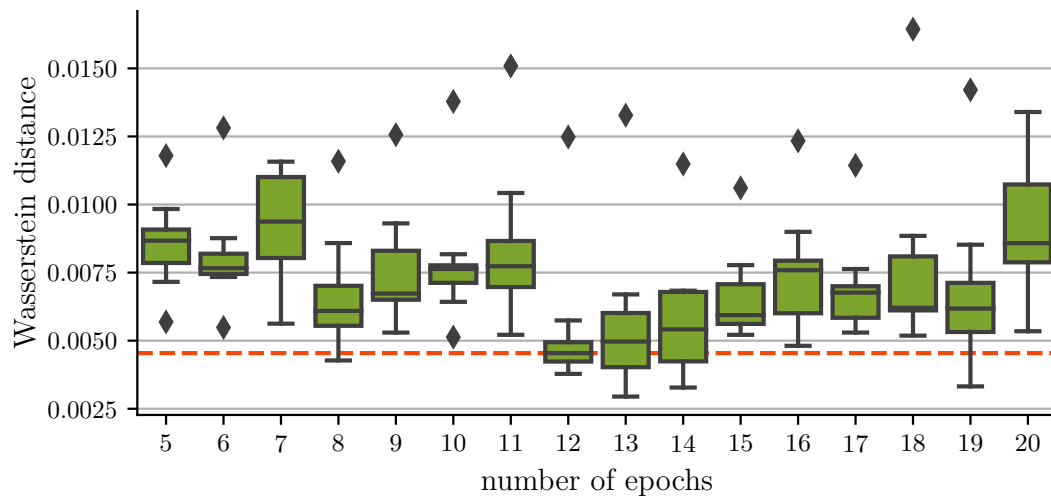


Figure 5.6: Box plot of the Wasserstein distance for different epoch counts.

5.4 Uncertainty and Results

Using the best-performing hyperparameters from the hyperparameter search, the final model is evaluated in more detail. Specifically, the uncertainty of the predictions is analyzed, and the physical plausibility of the predictions is investigated.

5.4.1 Bootstrapping

Bootstrapping [17] is a method to estimate the uncertainty of a model. It considers the given data as a random sample from a larger population. Therefore, by repeatedly sampling from the data and training a model on each sample, the model’s uncertainty can be determined as the variance of the results of the different models. For the present work, 50 bootstrap samples are used. Since using a bootstrap sample of the same size as the original (“bag”) data set might produce an inconsistent bootstrap estimator [42], 1 000 000 events are used as the original data set, while the bootstrap samples used for training contain 500 000 events as before.

5.4.2 Energy Spectrum

Figure 5.7 shows the unfolded energy spectrum of the optimized model, as well as the 68% confidence intervals. In Figure A.4, the per-bin histograms of the estimations for individual bootstrap runs are shown. While the probability density in the bins from 10^2 GeV to 10^4 GeV is estimated with high precision, both the relative deviation and the quantile range are large for the higher energy bins. This may be caused by the comparably small number of events in these bins, which leads to a large uncertainty in the estimation of the spectrum. Additionally, oscillations are a common issue in deconvolution problems. Similar behavior has been observed in [21]. The MAE of the predictions is 0.809 ranks, which explains the rather low accuracy of 42.7%. The Wasserstein distance is unexpectedly high at 0.0108.

5.4.3 Individual Events

Ordinal classification methods promise physically plausible probability distributions. In contrast to *LogisticAT* (used by Jäkel [23]), unimodality of the probability distribution is not enforced directly. Instead, only the threshold probabilities are constrained to be monotonically increasing (see subsection 4.2.1). As can be seen in Figure 5.8, slight deviations from unimodality do occur. Still, the indirect constraint on the threshold probabilities might have had a positive impact.

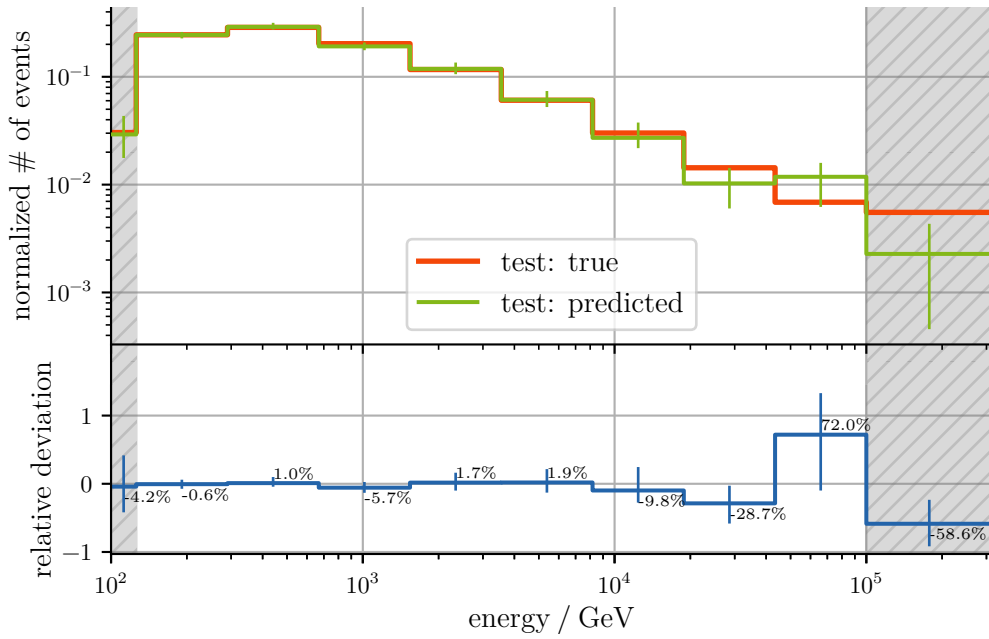


Figure 5.7: Energy spectrum and relative deviations of the bootstrap. The lines show the median of each bin. The error bars indicate the 68 % confidence intervals, ranging from the 16 % to the 84 % percentile. The greyed out areas mark the under- and overflow bins. The training spectrum is indistinguishable from the training spectrum and is therefore not shown.

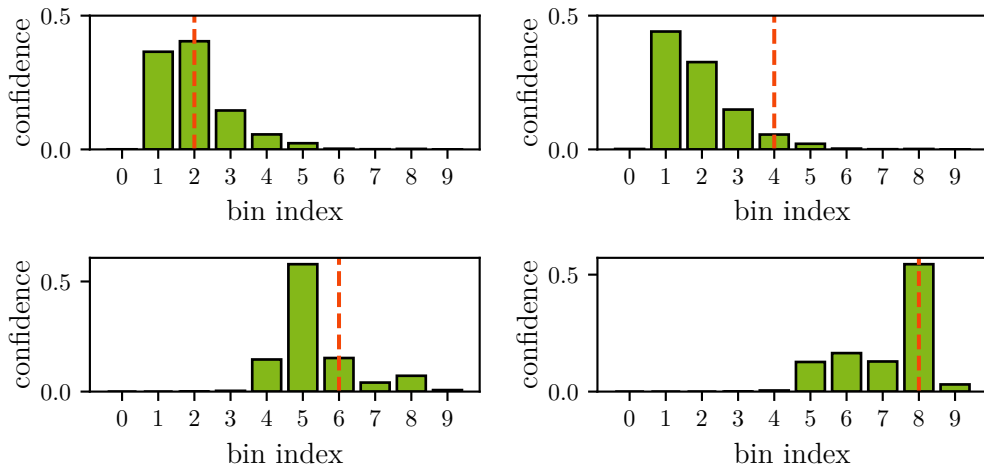


Figure 5.8: Confidence distributions of selected events. The dashed orange line shows the true bin of the event. Unimodality is violated with varying severity. Two events are misclassified, which is indicative of the low accuracy of the model.

5.5 Bias

As explained in section 3.2, DSEA⁺ is intended to eliminate the bias introduced by the energy spectrum of the Monte Carlo training data. In order to test whether the bias is indeed eliminated, the model is evaluated on a *stratified* data set, where each bin contains an equal number of events, whereas the training data is the same as before.

The results are shown in Figure 5.9. As can be seen, The model adapts to the unseen distribution of the test data. No significant amount of bias is observed. In comparison to the unfolding of unmodified test data in Figure 5.7, the relative deviations are even smaller on average. For comparison, in the work by Haefs [21], where the training data is stratified instead of the test data, relative deviations of more than 1500% are observed.

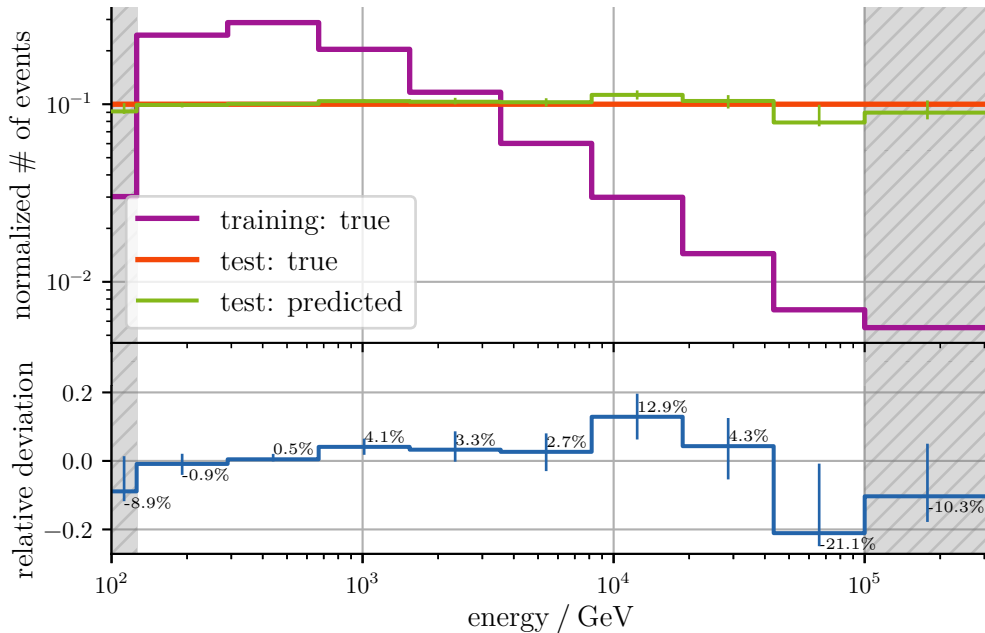


Figure 5.9: Energy spectrum and relative deviations of a bootstrap run on a stratified test data set.

6 Summary and Outlook

It has been shown that the combination of neural networks, ordinality and DSEA⁺ can be successfully applied to the problem of neutrino energy spectrum estimation with IceCube data. This was enabled by adding support for sample weights and confidences to CORN.

The new method is not unambiguously superior to the previous ones ([23] and [21]). A strict comparison is not possible in the first place, as this work introduces under-/overflow bins and makes use of adaptive step sizes. The randomly selected confidence distributions of a common neural network using softmax [21] are of comparable quality to those obtained in this work (see Figure 5.8), even though ordinality is disregarded in the former case. Compared to [21], higher accuracy is achieved (ours: 42.7% vs. theirs: < 39%). Both the RMSE (0.0164 vs. 0.000269) and the χ^2 distance (0.0392 vs. 0.003123) are worse, however, because of the large deviations in higher energy bins. In comparison to [23], similar Wasserstein distances are achieved (0.0108 vs. 0.00879), but using 10 instead of 12 bins. On the other hand, our probability distributions of single events are not strictly unimodal.

There is still a multitude of ways in which DSEA⁺ and the application thereof could be improved. Explicit regularization could dampen the currently observed oscillations in higher energy bins. Other hyperparameters, such as the shape of the neural network, are yet to be optimized. It might be possible to modify CORN so that the per-class confidence distributions are strictly unimodal. In general, other neural network architectures could be investigated. For example, graph neural networks already exceed boosted decision trees in terms of both resolution and speed [31]. Graph neural networks have the additional benefit of being less dependent on feature engineering as they can be applied to “raw” data (which DOM was hit, the collected charge, and the time of arrival). Finally, more data could be used for training. Because of the shape of the spectrum, the number of events in the highest energy bins remains relatively small compared to the complete data set. Although this thesis has demonstrated that DSEA⁺ eliminates the training bias, the effect of stratified training data on the overall performance has not been investigated.

A Appendix

A.1 Detector Signatures of Different Neutrino Flavors

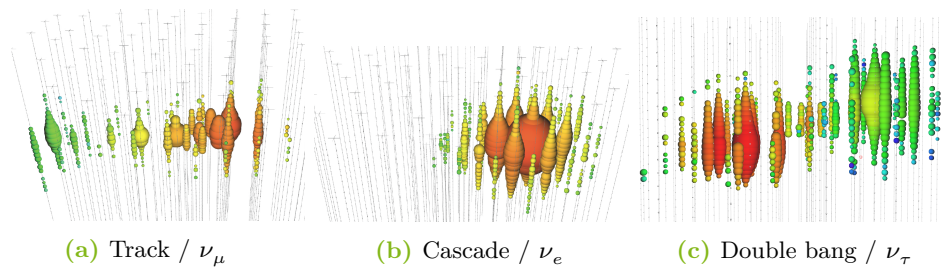


Figure A.1: Shapes of the Cherenkov light produced by neutrinos of different flavors. The coloring of the DOMs indicates the time of interaction, with red being the earliest and blue being the latest. [28]

A.2 DSEA⁺: Complete Algorithm

Algorithm 1 The DSEA⁺ algorithm with re-weighting of training examples and adjustable step size [8].

Input:

Observed data set $\mathcal{D}_{\text{obs}} = \{\mathbf{x}_n \in \mathcal{X} : 1 \leq n \leq N\}$

Training data set $\mathcal{D}_{\text{test}} = \{(\mathbf{x}_n, y_n) \in \mathcal{X} \times \{1, \dots, I\} : 1 \leq n \leq N'\}$

$\tau \geq 0$, regularization strength employed in the step size adaptation (default: 0)

$\epsilon > 0$, the minimal χ_{Sym}^2 distance between subsequent iterations (default: 10^{-6})

Prior density $\hat{\mathbf{f}}^{(0)}$ (default: $\hat{\mathbf{f}}_i^{(0)} = \frac{1}{I} \forall 1 \leq i \leq I$)

Output: Estimated target density $\hat{\mathbf{f}} \in \mathbb{R}^I$

$k \leftarrow 0$

repeat

$k \leftarrow k + 1$

$\forall 1 \leq n \leq N' : w_n^{(k)} \leftarrow \hat{\mathbf{f}}_{i(n)}^{(k-1)} / \mathbf{f}_{i(n)}^{\dagger}$

Infer \mathcal{M} from $\mathcal{D}_{\text{train}}$ weighted by $w_n^{(k)}$

$\forall 1 \leq i \leq I : p_i^{(k)} \leftarrow \frac{1}{N} \sum_{n=1}^N c_{\mathcal{M}}(i | \mathbf{x}_n) - \hat{\mathbf{f}}^{(k-1)}$

$\alpha_{\text{RUN}}^{(k)} \leftarrow \operatorname{argmin}_{\alpha \geq 0} \ell_r(\hat{\mathbf{f}}^{(k-1)} + \alpha p^{(k)})$

$\hat{\mathbf{f}}_i^{(k)+} \leftarrow \hat{\mathbf{f}}^{(k-1)} + \alpha_{\text{RUN}}^{(k)} \cdot p^{(k)}$

until $\chi_{\text{Sym}}^2(\hat{\mathbf{f}}^{(k)}, \hat{\mathbf{f}}^{(k-1)}) \leq \epsilon$

return $\hat{\mathbf{f}} \leftarrow \hat{\mathbf{f}}^{(k)}$

A.3 From Threshold to Per-Class Probabilities

Given four ranks with indices $q \in \{1, 2, 3, 4\}$, CORN's output layer has three neurons, which – after applying sigmoid and a cumulative product – yield three threshold probabilities: $\hat{P}(q > 1)$, $\hat{P}(q > 2)$ and $\hat{P}(q > 3)$. The goal is to calculate the probability of each class q , i.e. $\hat{P}(q = 1)$, $\hat{P}(q = 2)$, $\hat{P}(q = 3)$ and $\hat{P}(q = 4)$.

Using $q \in \{1, 2, 3, 4\}$, the following equations hold:

$$\hat{P}(q = 1) = \neg\hat{P}(q > 1) = 1 - \hat{P}(q > 1)$$

$$\begin{aligned} \hat{P}(q = 2) &= \neg\hat{P}(q \neq 2) \\ &= \neg(\hat{P}(q < 2) \vee \hat{P}(q > 2)) \\ &= 1 - ((1 - \hat{P}(q > 1)) + \hat{P}(q > 2)) \\ &= \hat{P}(q > 1) - \hat{P}(q > 2) \end{aligned}$$

$$\begin{aligned} \hat{P}(q = 3) &= \neg\hat{P}(q \neq 3) \\ &= \neg(\hat{P}(q < 3) \vee \hat{P}(q > 3)) \\ &= 1 - ((1 - \hat{P}(q > 2)) + \hat{P}(q > 3)) \\ &= \hat{P}(q > 2) - \hat{P}(q > 3) \end{aligned}$$

$$\hat{P}(q = 4) = \hat{P}(q > 3)$$

The same principle can be applied to any number of classes.

A.4 Monte Carlo Data Set

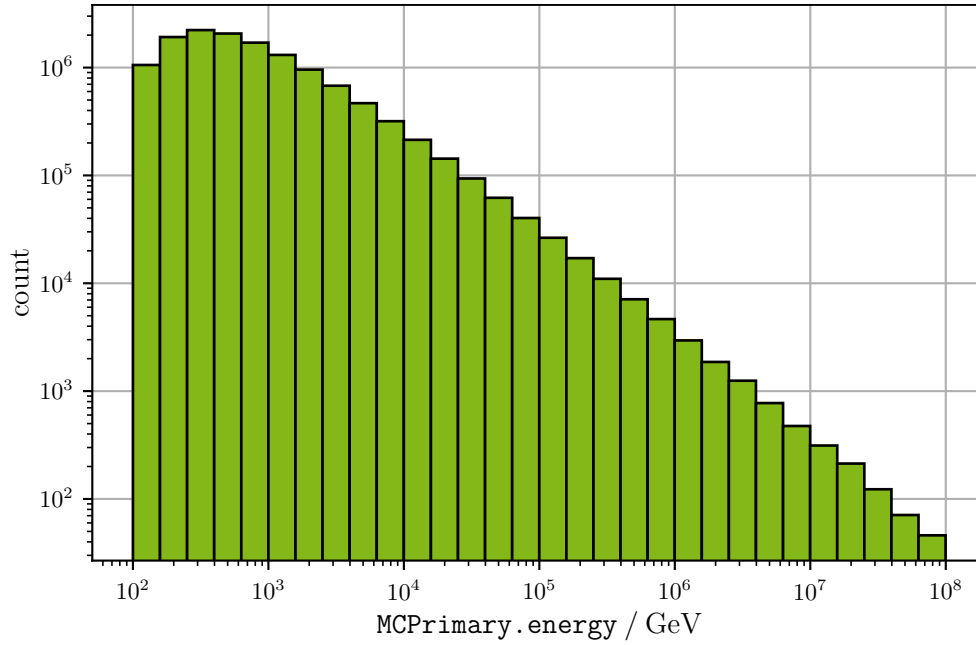


Figure A.2: Energy spectrum of the full, untouched Monte Carlo data set using 30 bins.

A.5 Additional Hyperparameter Plots

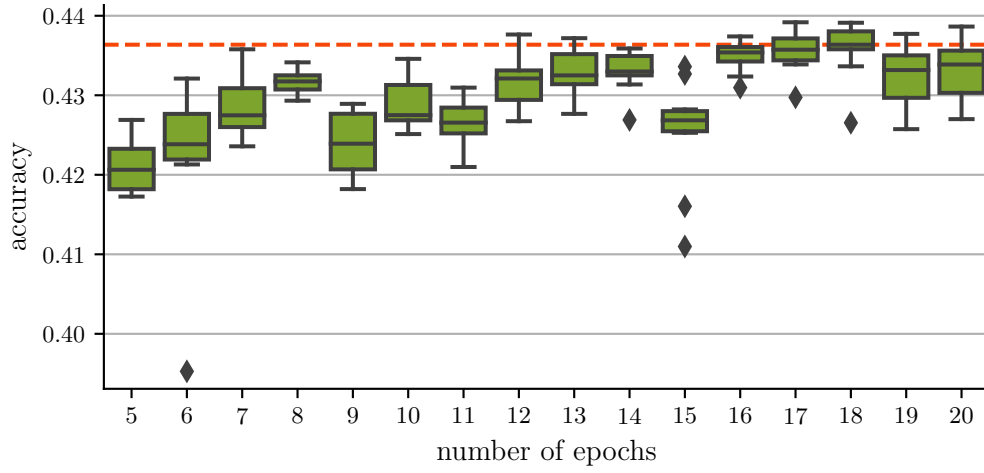


Figure A.3: Box plot of the accuracy for different epoch counts.

A.6 Bootstrap Distributions

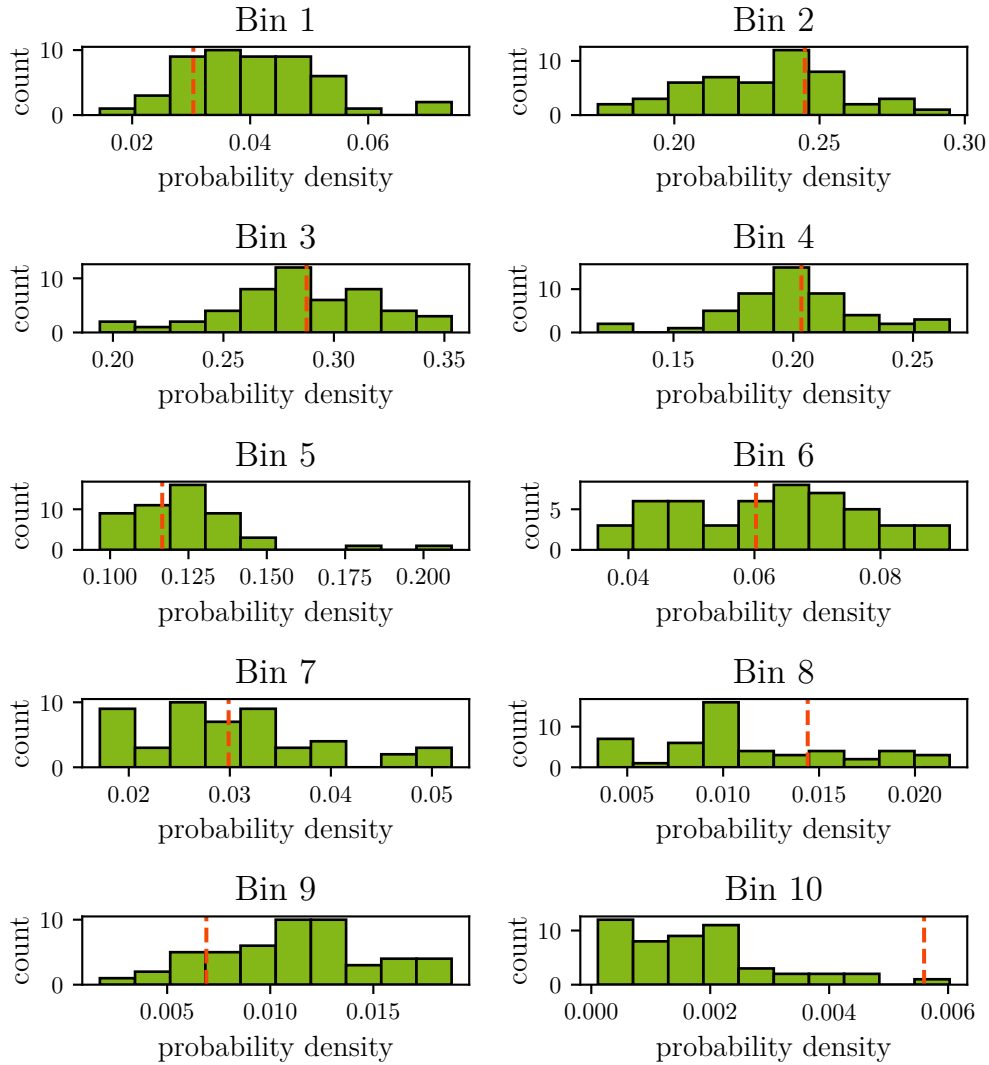


Figure A.4: Bootstrap distributions of each bin in Figure 5.7. The dashed orange line shows the true value.

A.7 Links etc.

Code on the chair's GitLab

<https://git.e5.physik.tu-dortmund.de/nweitkemper/Bachelor-code>

Data set in the chair's POOL file system

/net/big-tank/POOL/users/lkardum/new_mc_binning.csv (14.6 GB)

Bibliography

- [1] “5725-1: 1994, Accuracy (trueness and precision) of measurement methods and results-Part 1: General principles and definitions.” In: *International Organization for Standardization, Geneva* (1994).
- [2] Mark G Aartsen et al. “The IceCube Neutrino Observatory: instrumentation and online systems.” In: *Journal of Instrumentation* 12.03 (2017), P03012.
- [3] MG Aartsen et al. “Search for astrophysical tau neutrinos in three years of IceCube data.” In: *Physical Review D* 93.2 (2016), p. 022001.
- [4] MG Aartsen et al. “Searches for extended and point-like neutrino sources with four years of IceCube data.” In: *The Astrophysical Journal* 796.2 (2014), p. 109.
- [5] John F Beacom et al. “Measuring flavor ratios of high-energy astrophysical neutrinos.” In: *Physical Review D* 68.9 (2003), p. 093005.
- [6] Michael W Browne. “Cross-validation methods.” In: *Journal of mathematical psychology* 44.1 (2000), pp. 108–132.
- [7] Mirko Bunse. *CherenkovDeconvolution.py*. 2013. URL: <https://github.com/mirkobunse/CherenkovDeconvolution.py>.
- [8] Mirko Bunse. “DSEA Rock-Solid. Regularization and Comparison with other Deconvolution Algorithms.” Master’s thesis. TU Dortmund, 2018.
- [9] Mirko Bunse et al. “Unification of Deconvolution Algorithms for Cherenkov Astronomy.” In: *5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2018, pp. 21–30.
- [10] Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. “Rank consistent ordinal regression for neural networks with application to age estimation.” In: *Pattern Recognition Letters* 140 (2020), pp. 325–331.
- [11] Sung-Hyuk Cha. “Comprehensive survey on distance/similarity measures between probability density functions.” In: *City* 1.2 (2007), p. 1.
- [12] IceCube Collaboration. *Dataset 11374*. Not publicly available. 2015. URL: <https://grid.icecube.wisc.edu/simulation/dataset/11374>.
- [13] IceCube Collaboration. *IceCube*. URL: <https://icecube.wisc.edu/science/icecube/>.

- [14] Giulio D’Agostini. “A multidimensional unfolding method based on Bayes’ theorem.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 362.2-3 (1995), pp. 487–498.
- [15] Giulio D’Agostini. “Improved iterative Bayesian unfolding.” In: *arXiv preprint arXiv:1010.0632* (2010).
- [16] Manoranjan Dash and Huan Liu. “Feature selection for classification.” In: *Intelligent data analysis* 1.1-4 (1997), pp. 131–156.
- [17] Bradley Efron. “Bootstrap methods: another look at the jackknife.” In: *Breakthroughs in statistics*. Springer, 1992, pp. 569–593.
- [18] Brent Follin et al. “First detection of the acoustic oscillation phase shift expected from the cosmic neutrino background.” In: *Physical review letters* 115.9 (2015), p. 091301.
- [19] Ivar Fredholm. “Sur une classe d’équations fonctionnelles.” In: *Acta mathematica* 27 (1903), pp. 365–390.
- [20] Azusa Gando et al. “Search for Majorana neutrinos near the inverted mass hierarchy region with KamLAND-Zen.” In: *Physical review letters* 117.8 (2016), p. 082503.
- [21] Samuel Haefs. “Lösungen inverser Probleme. Entfaltung von Energiespektren mit neuronalen Netzen in DSEA.” Bachelor’s thesis. TU Dortmund, 2022.
- [22] Karolin Hymon and Tim Ruhe. “Seasonal Variations of the Unfolded Atmospheric Neutrino Spectrum with IceCube.” In: *arXiv preprint arXiv:2107.09349* (2021).
- [23] Jan Philipp Jäkel. “Ordinal Classification in DSEA.” Bachelor’s thesis. TU Dortmund, 2021.
- [24] Ibrahim Kandel and Mauro Castelli. “The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset.” In: *ICT express* 6.4 (2020), pp. 312–315.
- [25] Ulrich F Katz and Ch Spiering. “High-energy neutrino astrophysics: Status and perspectives.” In: *Progress in Particle and Nuclear Physics* 67.3 (2012), pp. 651–704.
- [26] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).
- [27] Ron Kohavi, David H Wolpert, et al. “Bias plus variance decomposition for zero-one loss functions.” In: *ICML*. Vol. 96. 1996, pp. 275–83.

-
- [28] Marek Kowalski, IceCube Collaboration, et al. “Neutrino astronomy with IceCube and beyond.” In: *Journal of Physics: Conference Series*. Vol. 888. 1. IOP Publishing. 2017, p. 012007.
- [29] Ling Li and Hsuan-Tien Lin. “Ordinal regression by extended binary classification.” In: *Advances in neural information processing systems* 19 (2006).
- [30] Natalie Milke et al. “Solving inverse problems with the unfolding program TRUEE: Examples in astroparticle physics.” In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 697 (2013), pp. 133–147.
- [31] Martin Ha Minh. “Reconstruction of Neutrino Events in IceCube using Graph Neural Networks.” In: *arXiv preprint arXiv:2107.12187* (2021).
- [32] *mord: ordinal regression in Python*. URL: <https://github.com/fabianp/mord>.
- [33] Nicki Skafte Detlefsen et al. *TorchMetrics - Measuring Reproducibility in PyTorch*. Feb. 2022. DOI: 10.21105/joss.04101. URL: <https://github.com/Lightning-AI/metrics>.
- [34] Zhenxing Niu et al. “Ordinal regression with multiple output cnn for age estimation.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4920–4928.
- [35] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In: *Advances in Neural Information Processing Systems* 32. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [36] Sayak Paul. *Bayesian Hyperparameter Optimization - A Primer*. 2020. URL: <https://wandb.ai/site/articles/bayesian-hyperparameter-optimization-a-primer>.
- [37] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [38] Hanchuan Peng, Fuhui Long, and Chris Ding. “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy.” In: *IEEE Transactions on pattern analysis and machine intelligence* 27.8 (2005), pp. 1226–1238.
- [39] *PyTorch Lightning*. URL: <https://pytorchlightning.ai/>.
- [40] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. “A metric for distributions with applications to image databases.” In: *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 59–66.

- [41] Tim Ruhe et al. “Mining for Spectra. The Dortmund Spectrum Estimation Algorithm.” In: *Proc. of the ADASS XXVI* (2016).
- [42] Jun Shao. “Bootstrap sample size in nonregular cases.” In: *Proceedings of the American Mathematical Society* 122.4 (1994), pp. 1251–1262.
- [43] Xintong Shi, Wenzhi Cao, and Sebastian Raschka. “Deep Neural Networks for Rank-Consistent Ordinal Regression Based On Conditional Probabilities.” In: *arXiv preprint arXiv:2111.08851* (2021).
- [44] Christian Spiering. “Towards high-energy neutrino astronomy.” In: *From Ultra Rays to Astroparticles*. Springer, 2012, pp. 231–263.
- [45] Jean-Luc Starck, Eric Pantin, and Fionn Murtagh. “Deconvolution in astronomy: A review.” In: *Publications of the Astronomical Society of the Pacific* 114.800 (2002), p. 1051.
- [46] Andrei Nikolaevich Tikhonov. “On the solution of ill-posed problems and the method of regularization.” In: *Doklady akademii nauk*. Vol. 151. 3. Russian Academy of Sciences. 1963, pp. 501–504.
- [47] Cort J Willmott and Kenji Matsuura. “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance.” In: *Climate research* 30.1 (2005), pp. 79–82.
- [48] In-Kwon Yeo and Richard A Johnson. “A new family of power transformations to improve normality or symmetry.” In: *Biometrika* 87.4 (2000), pp. 954–959.

Acknowledgements

Thanks to Prof. Dr. Dr. Wolfgang Rhode and Prof. Dr. Johannes Albrecht for taking the time to review my thesis and for being open to questions.

I would also like to thank my supervisors Karolin Hymon and Leonora Kardum as well as Tim Ruhe and Mirko Bunse for their support and guidance throughout this project.

Furthermore, I am grateful to Samuel Haefs, who worked on his Bachelor's thesis in parallel with me, for the inspiring discussions and for providing a baseline for the implementation of neural networks in DSEA⁺.

I appreciate the chair E5b providing the technical infrastructure as well as helpful feedback regarding my half-time talk.

Thanks to Dr. Maximilian Linhoff for providing a \LaTeX template for this thesis.

A special thanks to my family for their continued support and encouragement.

Eidesstattliche Versicherung

(Affidavit)

Weitkemper, Nicolai

220837

Name, Vorname
(surname, first name)

Matrikelnummer
(student ID number)

Bachelorarbeit
(Bachelor's thesis)

Masterarbeit
(Master's thesis)

Titel
(Title)


Ordinal classification with neural networks in DSEA

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel 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.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Soest, 30.09.2022

Ort, Datum
(place, date)


Unterschrift
(signature)

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:

Official notification:

Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).


The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Soest, 30.09.2022

Ort, Datum
(place, date)


Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**