Unsupervised learning for identification of event topologies in ORCA

Bachelorarbeit aus der Physik

Vorgelegt von

Jelena Mila Ćelić

30. Juli 2018

Erlangen Centre for Astroparticle Physics
Friedrich-Alexander-Universität Erlangen-Nürnberg

1. Betreuer: Dr. Thomas Eberl
2. Betreuer: Prof. Dr. Gisela Anton

Abstract

The KM3NeT neutrino detector ORCA,used to study the neutrino oscillation of atmospheric neutrinos and to determine the neutrino mass hierarchy, detects charged particles, which were produced from neutrino interaction, emitted Cherenkov light. The high-dimensional data from the neutrino events are analyzed and should identify the particle with the help of unsupervised learning approach. This work gives a short description of the detector and the physical background behind this experiment. Also the basic concepts of Convolutional Neural Networks and unsupervised learning will be presented. Also the effect of the ϵ parameter of the optimizer ADAM is analyzed. Due to the performance with the ϵ parameter, a dropout has been added to the neural network after every convolutional layer and the dropout value has been changed in the supervised encoder training, aiming to increase the overall performance. After that, different models with different bottleneck dimensions are being tested and their performances for the identification of event topologies are observed. In the end the best performing neural network has been tested on its robustness by using manipulated data to recreate a real setting.

Contents

1 Introduction

Nowadays, Deep Learning, a specific Machine learning algorithm, has become an emerging technique to visualize and classify highly dimensional data in science. Especially in particle physics, where it is tried to understand the fundamental nature of our universe, a huge amount of data is collected in order to analyze rare events like neutrino events. These simulations and recognitions are time-consuming and laborious. Some experiments like the Daya Bay Reactor Neutrino Experiment, studying anti-neutrinos which are produced in a nuclear reactor, or the Large Hadron Collider CERN, trying to discover new and unknown particles, have adopted deep learning algorithms for their data analysis. These projects have shown great results in event recognitions compared to normal simulations(for more information see [1] and [2]). In this work, a deep learning algorithms will be used and analyzed for a identification of event topologies in the neutrino detector KM3Net/ORCA.

2 The architecture of the neutrino telescope KM3NeT/ORCA

The main goals of the KM3-Net-Collaboration are on the one hand the discovery and observation of high-energetic neutrinos and on the other hand the mass hierarchy of these particles, which will be explained later in the thesis. An infrastructure of three neutrino telescopes are planned to realize in the Mediterranean Sea. One of them is the neutrino detector ORCA, which architecture will be described in the following.

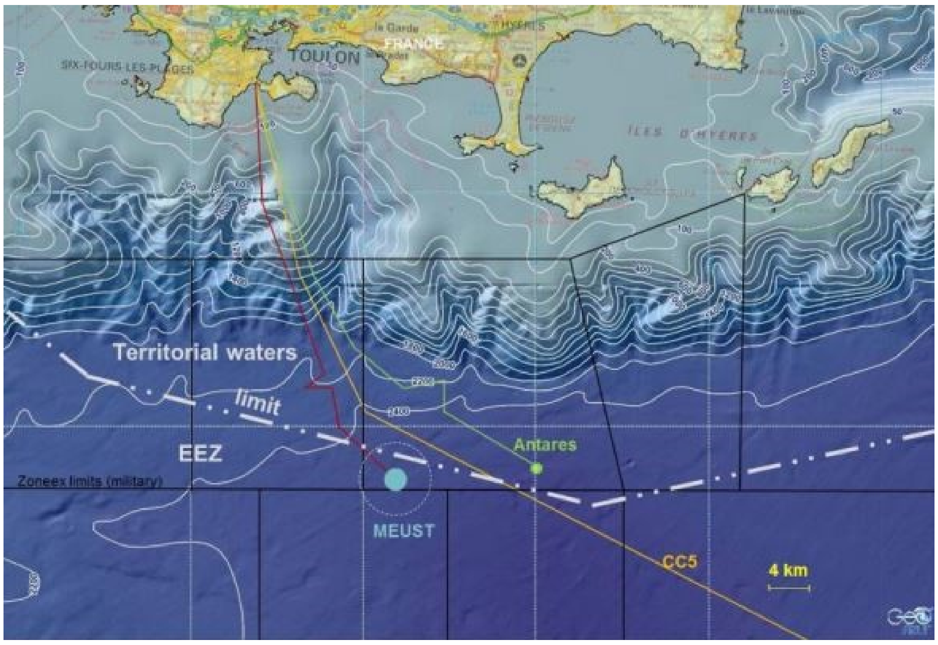


Figure 1: The detector site of the neutrino telescope KM3Net/ORCA[3]

The neutrino detector KM3NeT/ORCA is a three dimensional array of photosensors, which are detecting the Cherenkov light of secondary particles. The neutrino telescope will be located at a depth of 2450 m and about 40 km offshore from Toulon, France. Furthermore, it is located 10 km to the west of the neutrino telescope ANTARES, as it can be seen on Figure 1. In total, the ORCA detector has about 115 strings, each with 18 optical sensor units, called Digital Optical Modules (DOMs). The layout of the detector with each string position can be seen in figure 2 (right). Furthermore, each string is connected to a junction box at the sea ground, which functions as a power supply and transfers the data to the shore, and is pulled upwards by buoys. For the ORCA configuration, the average horizontal spacing between the strings is about 20 m. A DOM is a transparent 17 inch diameter glass sphere, which houses 21 Photomultipliers and their electronics for the readout. In addition, the spacing between two DOMs is set to 9 m [3].

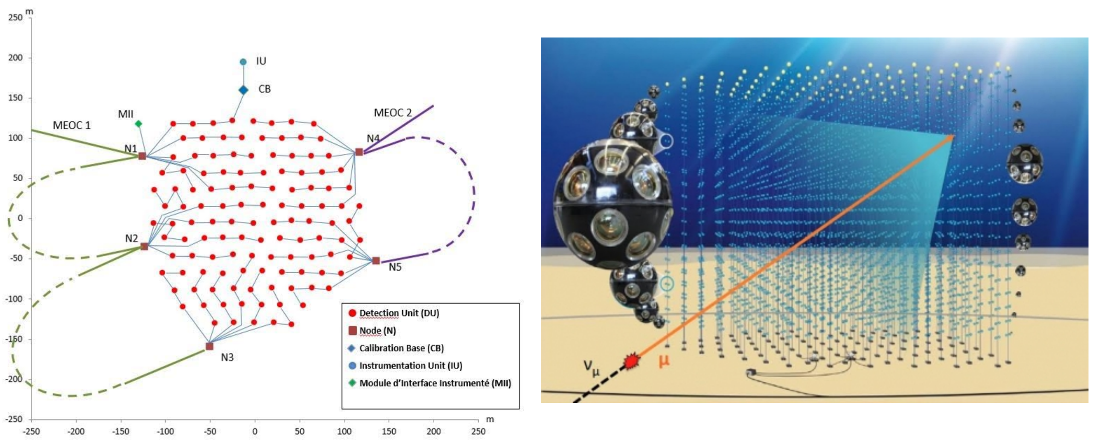


Figure 2: Layout of the string positions (left) and an scheme (right) of the neutrino telescope ORCA[3]

3 Theoretical background of neutrino physics

3.1 Production of atmospheric neutrinos

Overall, neutrinos are very light uncharged particles in the Standard Model of particle physics, interacting only by the weak interaction. Especially, atmospheric neutrinos emerge from interactions of cosmic rays with the atmosphere. Cosmic rays, which consist of about 98% of hadrons – like protons ($\approx$ 87%), α-particles ($\approx$ 11%), heavier nuclei ($\approx$ 2%) – and 2% electrons, have an energy spectrum that can be described as:

$$N(E) \propto E^{-2.7} \text{ for } E < 10^{15} \text{ eV} \tag{1}$$

As a result of equation (1), highly energetic atmospheric neutrinos are very rare to find in the atmosphere. In interactions of cosmic rays with nitrogen and oxygen nuclei, particles are produced, which interact with other particles. As a result, a cascade of particles is formed, called air shower, that consists of charged mesons, like kaons and pions. These particles decay to muon neutrinos, electron neutrinos and their antineutrinos:

$$\pi^+,\, K^+ \rightarrow \nu_\mu \mu^+ \rightarrow \nu_\mu e^+ \nu_e \bar{\nu_\mu}$$
$$\pi^-,\, K^- \rightarrow \bar{\nu_\mu} \mu^- \rightarrow \bar{\nu_\mu} e^- \bar{\nu}_e \nu_\mu$$

Besides that, the ratio R of the produced muon neutrinos and electron neutrinos is

$$R = \frac{\nu_\mu + \bar{\nu}_\mu}{\nu_e + \bar{\nu}_e} \approx 2. \tag{2}$$

The approximation of (2) applies only to the case that all muons decay. Due to that, the average number of muon neutrinos is two times higher than the number of electron neutrinos [5].

3.2 Neutrino oscillations in vacuum

A special feature of neutrinos is that during their propagation through space their flavor can change. The probability of the transition from a neutrino flavor into another one changes intermittently with time, so that a electron neutrino, propagating a certain distance, can be registered with a high probability as a muon neutrino. But after a certain amount of time, the probability of another transition is very small. This phenomenon is called neutrino oscillation because of the oscillating transition probability [5]. A reason for this phenomenon is that the neutrino flavor states ν_α are different than the neutrino mass eigenstates. These eigenstates are related by a unitary matrix V:

$$\nu_\alpha = \sum V_{\alpha i} \nu_i. \tag{3}$$

The mixing matrix V is often named V_{PMNS} after its authors Pontecorvo-Maki-Nakagawa-Sakata. Especially for three neutrinos, the matrix V contains the three rotation angles Θ_{23},Θ_{13}, $\Theta_{12}(0 \leqslant \Theta_i \leqslant \frac{\pi}{2})$ and three CP-violating phases δ, ϕ_2 and $\phi_3(0 \leqslant \delta,\, \phi_i \leqslant 2\pi)$. V is defined as the matrix product

$$V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{pmatrix} \cdot \begin{pmatrix} c_{13} & 0 & s_{13}e^{-i\delta} \\ 0 & 1 & 0 \\ -s_{13}e^{i\delta} & 0 & c_{13} \end{pmatrix} \cdot \begin{pmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{i\phi_2/2} & 0 \\ 0 & 0 & e^{i\phi_3/2} \end{pmatrix} \tag{4}$$

where c_{ij} and s_{ij} stand for $\cos\Theta_{ij}$ and $\sin\Theta_{ij}$. The angles Θ_{ij} are denoted as the mixing angles. Because the oscillation probabilities are independent of the Majorana phases and only depend on the Dirac phase δ, the unitary matrix with ($\phi_2 = \phi_3 = 0$ can be defined as:

$$V = \begin{pmatrix} c_{13}c_{12} & c_{13}s_{12} & s_{13}e^{-i\delta} \\ -s_{12}c_{23} - c_{12}s_{23}s13e^{i\delta} & c_{12}c_{23} - s_{12}s_{23}s13e^{i\delta} & s_{23}c13 \\ s_{12}s_{23} - c_{12}c_{23}s13e^{i\delta} & -c_{12}s_{23} - s_{12}c_{23}s13e^{i\delta} & c_{23}c_{13} \end{pmatrix}. \tag{5}$$

As a result the vacuum oscillation probabilities can be approximated by:

$$
\begin{aligned}
P(\nu_e \to \nu_e) &\simeq 1 - \sin^2 2\Theta_{13} \sin^2 \Delta_{31} \\
P(\nu_e \to \nu_\mu) &\simeq s_{23}^2 \sin^2 2\Theta_{13} \sin^2 \Delta_{31} \\
P(\nu_\mu \to \nu_\mu) &\simeq 1 - (c_{13}^4 \sin^2 2\Theta_{23} + s_{23}^2 \sin^2 2\Theta_{13} \sin^2 \Delta_{31} \\
P(\nu_\mu \to \nu_\tau) &\simeq c_{13}^4 \sin^2 2\Theta_{23} \sin^2 \Delta_{31},
\end{aligned}
$$

with the oscillation arguments for atmospheric and solar neutrinos

$$\Delta_{31} = \frac{\delta m_{31}^2 L}{4E_\nu},\ \Delta_{21} = \frac{\delta m_{21}^2 L}{4E_\nu}, \tag{6}$$

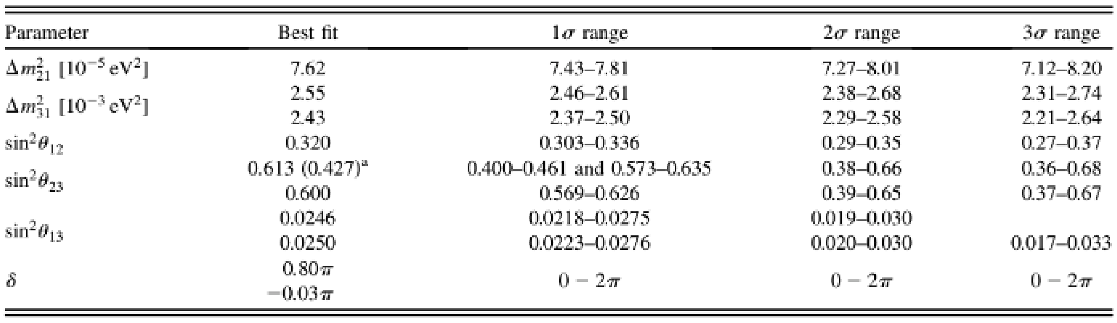
and the neutrino mass differences

$$\delta m_{31}^2 = m_3^2 - m_1^2,\ \delta m_{21}^2 = m_2^2 - m_1^2. \tag{7}$$

For oscillations of atmospheric neutrinos the argument Δ_{31} is dominant, such that the mass difference δm_{31} is relevant.

3.3 Determination of the neutrino mass hierarchy with ORCA

Figure 3 shows the neutrino oscillation parameter which were determined by other neutrino experiments.



Parameter	Best fit	1σ range	2σ range	3σ range
Δm_{21}^2 [10^{-5} eV^2]	7.62	7.43–7.81	7.27–8.01	7.12–8.20
Δm_{31}^2 [10^{-3} eV^2]	2.55	2.46–2.61	2.38–2.68	2.31–2.74
	2.43	2.37–2.50	2.29–2.58	2.21–2.64
$\sin^2\theta_{12}$	0.320	0.303–0.336	0.29–0.35	0.27–0.37
$\sin^2\theta_{23}$	0.613 (0.427)[a]	0.400–0.461 and 0.573–0.635	0.38–0.66	0.36–0.68
	0.600	0.569–0.626	0.39–0.65	0.37–0.67
$\sin^2\theta_{13}$	0.0246	0.0218–0.0275	0.019–0.030	
	0.0250	0.0223–0.0276	0.020–0.030	0.017–0.033
δ	0.80π -0.03π	$0 - 2\pi$	$0 - 2\pi$	$0 - 2\pi$

Figure 3: The observed neutrino parameters from the PMNS matrix [7]

With the help of neutrino oscillation experiments, like ORCA, the differences of the squared masses $\Delta m_{ij}^2 = m_i^2 - m_j^2$(with $i, j = 1, 2, 3$) can be derived. The absolute neutrino mass and the neutrino mass hierarchy (seen in figure 4), specifying the order of the neutrino mass eigenstates, are still unknown.

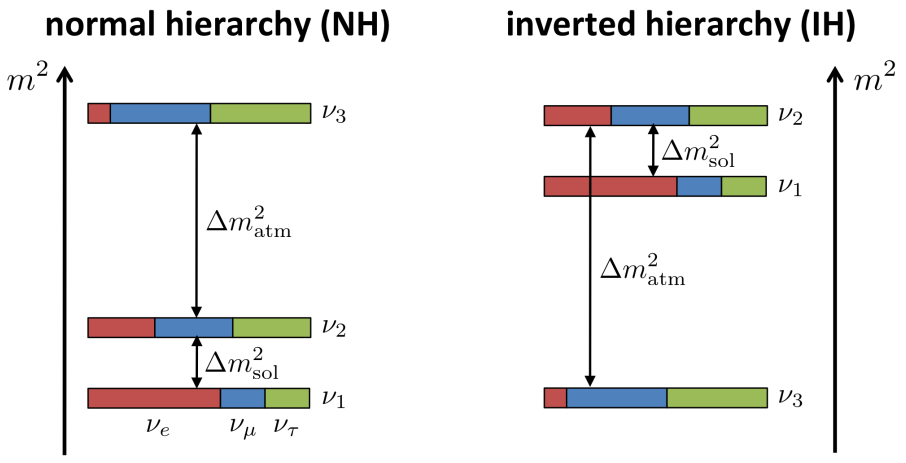


Figure 4: Illustration of the normal and the inverted neutrino mass hierarchy [8]

Because it is only known that $m_1 < m_2$, two models have to be differentiated, which are called the normal hierarchy (NH) and the inverted hierarchy (IH):

- NH: $m_1 < m_2 < m_3$ with $\Delta m^2_{21} = \Delta m^2_{sol}$ and $\Delta m^2_{31} = \Delta m^2_{atm}$
- IH: $m_3 < m_1 < m_2$ with $\Delta m^2_{21} = \Delta m^2_{sol}$ and $\Delta m^2_{13} = \Delta m^2_{atm}$

Water Cherenkov telescopes like ORCA can help to identify the neutrino mass hierarchy because the atmospheric neutrinos have to propagate through the earth and get a longer baseline. When neutrinos move through matter, they can spread electrons and nucleons such that the effective potential is risen by these interactions. As a result, neutrinos have a different effective masses in matter. By identifying the matter induced resonance, occurring for neutrinos (for NH) or for antineutrinos (for IH)), it is possible to determine the neutrino mass hierarchy. In addition, the energy range of the atmospheric neutrinos is quite big so that there are many different ratios of the distance and energy, because the neutrinos can propagate either through the core or the mantle [5]. Moreover, the neutrinos can only be detected indirectly, as they only interact weakly. During a CC-interaction, a neutrino and a nucleus exchange a $W^{\pm}$-boson because of deep inelastic scattering. Due to that, charged leptons are produced [6]. Then, the charged particles emit Cherenkov light because their velocity is faster than the effective speed of light c/n (c:speed of light, n: fraction index) in water. The opening angle of the Cherenkov cone of the particle depends on the velocity $\beta = \frac{v}{c}$ [5]:

$$\cos\Theta = \frac{1}{\beta n} \tag{8}$$

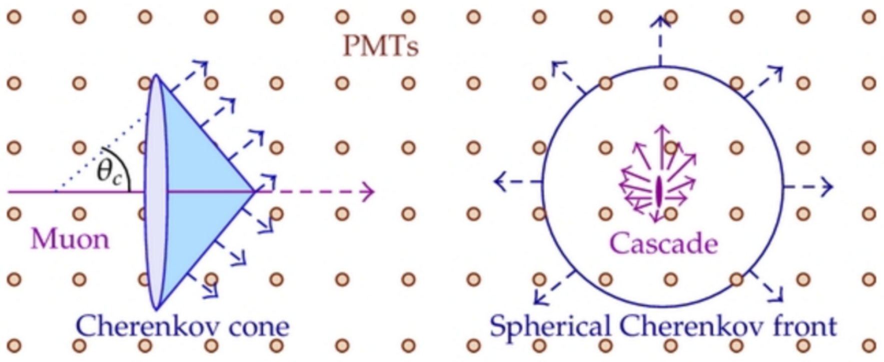


Figure 5: The event topologies of charged particles in a Water Cherenkov detector, which are the secondary products of neutrinos [9]

In a Water Cherenkov detector like ORCA, two characteristic signatures can be observed in the water: tracks and cascades(see figure 5). Muons produce a track in the water during their propagation through the neutrino detector and cascades only occur by electron and tau neutrinos and for the neutral current interaction also with muon neutrinos [9].

4 Introduction of Deep Learning

An event reconstruction of specific events like neutrinos requires Machine Learning algorithms,e.g. Deep Learning for that. The Deep Learning approach uses deep neural networks, which basic element is a neuron, which has some similar abilities as the biological one. Figure 6 visualizes the neuron in a Deep Learning network, viewed in a mathematical way, compared to a neuron in the biology.

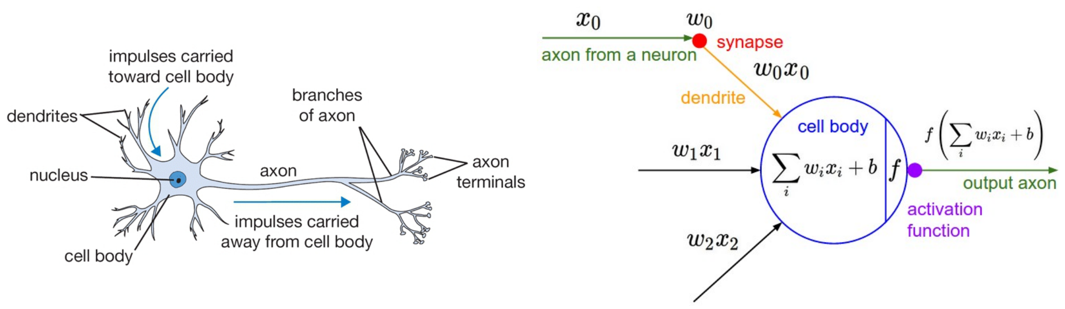


Figure 6: Visualization of a neuron in a biological way(left) and in Deep Learning way(right)[12]

The neural network is a system of fully connected layers. These layers consist of neurons which are densely connected with the neurons from the previous and the next layer, but not connected within a single layer. Each of neurons applies an arbitrary function, called the activation function, to its input information. The output of the activation function can be described as:

$$f(x_i, W, b) = \sigma(\sum_i W_i x_i + b) \tag{9}$$

Before the activation is applied, every input of the neuron x_i which is multiplied with a weight W_i, is summed up and a constant bias b is added. The last layer of a fully connected neural network doesn't have an activation, because it represents the class scores. In the end, a loss function, in some papers also called cost function, measures the quality of the chosen parameters on how well the scores agree with the labels of the training data. To minimize the loss function, the neural network is optimized by gradient descent. The gradient descent computes the gradient and updates the parameter in a loop. The weights and biases of the network are adjusted, because the parameters (weight and bias) of the neural network are changed in the direction of the negative gradient of the loss function:

$$W \mapsto W - \eta \frac{\partial L(f((x_i, W, b))}{\partial \mathbf{W}} \tag{10}$$

with η defining the stepsize, also called the learning rate. As a result, the loss function decreases. [10]

4.1 The Architecture of a Convolutional Neural Network

Deep neural networks tend to struggle with computational complexity which are required to analyze input data for image recognition, because their training is time-consuming. Besides that, the effect of the overfitting occurs because of the high number of parameters required to train. An analogous to fully-connected neural networks are Convolutional Neural Networks, which are more suitable for image-classifications.

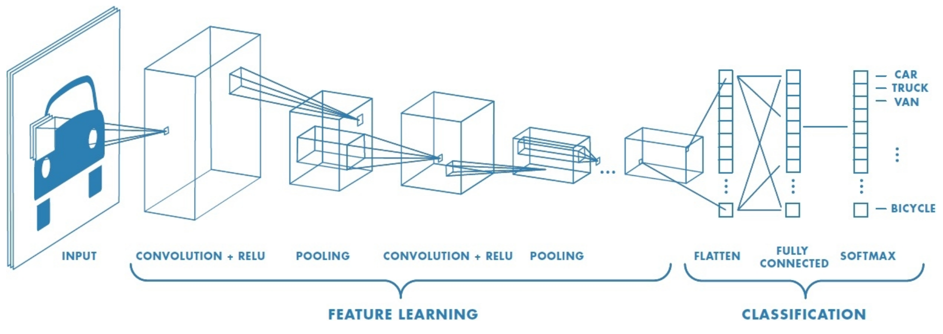


Figure 7: Artwork of a Convolutional Neural Network with all its layers trying to identify a car

Convolutional Neural Networks (ConvNets) are made of neurons, which have the ability to learn and optimize their weights and biases. In the layers of a ConvNet these neurons

are arranged in three dimensions: width, height and depth. The third dimension depth symbolizes the third dimension of an activation volume and does not refer to depth of the whole network. The layers' function is to transform one volume of activations to another one through a function which is differentiable. For building a ConvNet three types of layers are used: convolutional layer, pooling layer, fully-connected layer.
The convolutional layer is the main part of a ConvNet, because it does the most of the computational heavy work, and its parameters consist of a set of learnable kernels. A kernel has the same depth as the input volume, but only a small extent in width and height. Each of these filters produces a two dimensional activation map which represents the responses of the kernel in every position. As a result, the network will learn filters which activate if they recognize a specific visual feature, like a shape of an object. The property that differentiates ConvNets from other neural networks is that each neuron is only connected to a local region of the input volume. This spatial extent is a hyperparameter, also called the receptive field of the neuron or filter size. In addition, the depth of the receptive field equals the depth of the input volume. In summary, there is an asymmetry between the spatial dimension (width and height) and the depth dimension in ConvNets. Other hyperparameters are the stride, which can produce smaller volume of the output spatially, and the zero-padding, which helps to control the spatial size of the output. Another way to minimize the number of parameters is the principle of parameter sharing. This method works on the supposition that a useful feature region which is computed at a spatial region of the ConvNet, can be used a different region. During the backpropagation only single sets of weights have to be updated rather than every single one.
A pooling layer is commonly inserted between convolutional layers in a ConvNet, because it reduces the spatial size of its input volume. Compared to the convolutional layers, the pooling layer has no trainable weights, so it can't be effected by a training of the ConvNet. As a result, the amount of parameters and the complexity of the model are reduced. In this work, only the average pooling operation is used and half sizes the given dimension of the layer [11].

4.2 Unsupervised learning with autoencoders

In image recognition, there are two learning schemes in the Deep Learning approach: the supervised learning and the unsupervised learning. The supervised learning, which has been described in the previous chapter, is learning through labeled inputs and trained on a known output. The negative aspect of the supervised learning using for image-focused pattern-recognition is that the neural network has to be trained on simulations. In reality, the real data and the simulations are not the same. Therefore, a convolutional neural network can be trained with the unsupervised learning method. This means that its training doesn't require any labels. For the unsupervised learning a Convolutional neural Network is required that can encode the raw input and chooses its own crucial features of the input. This ConvNet is called the autoencoder. Seen in figure 8, an autoencoder can be divided in three separate parts: encoder, bottleneck and the decoder.

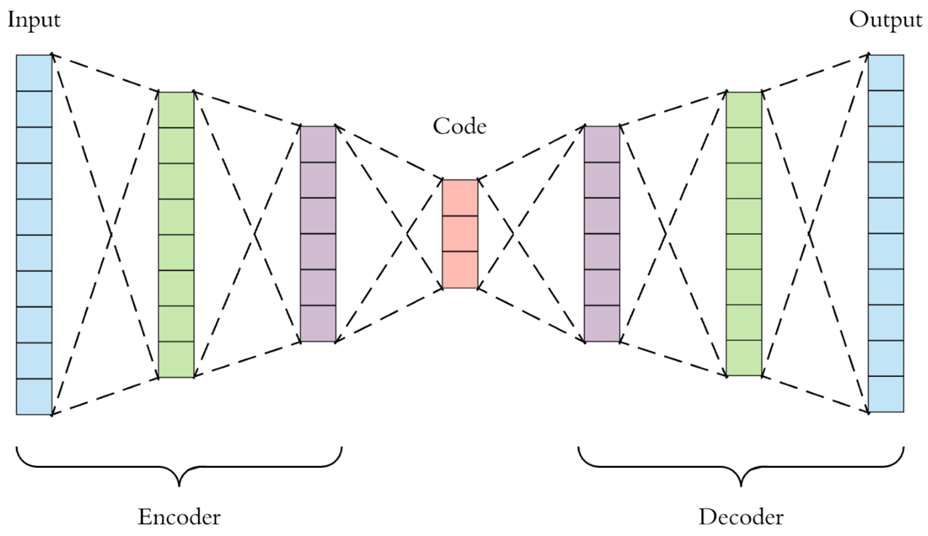


Figure 8: Artwork of the architecture of an autoencoder with the encoder, decoder and bottleneck(red layer)[13]

The unsupervised learning approach for image classification is based on two steps. First of all, during an autoencoder training the encoder extracts the fundamental features of the raw input data by minimizing the dimension of the convolutional layers. The convolutional layer with the smallest number of neurons is called the bottleneck. With the information of the bottleneck, the ConvNet tries to decode the crucial features by Upsampling until the input image is reconstructed as the output. The autoencoder loss function gives a statement, how well the autoencoder reconstructed the input image. As a result, the use of the autoencoder is to return a perfect reconstruction of the input, using only the most important prosperities. To perform a classification of an image, the second step is that the weights of the convolutional layers from the encoder are copied and frozen and a fully-connected layer like a dense layer is added [13].

5 Unsupervised learning for particle identification in ORCA

Due to the great results of using Deep Learning algorithms for image recognition, this technique is used for identification of event topologies in the neutrino detector KM3Net/ORCA. The neural network will be trained unsupervised to avert from simulations. The goal of the work is to get a good separation of tracks and showers.
The code for the implementation of the convolutional neural networks is written in Python and uses the imported Deep Learning package Keras. Keras is a high-level neural networks API, which supports convolutional neural networks and runs trouble-free on CPUs and GPUs. Besides Keras, also TensorFlow is used for basic Machine Learning algorithms. For more detailed information about Keras and TensorFlow their documentation can be looked up [14] and [15]. The code is forked from Stefan Reck and is modified for the track/shower-recognition. Also the model vgg-5-4000, which will be described later in the work, is created new. The whole code can be found in [16] .
Each tested model is constructed according to the principle of the VGG-Net. The VGG-Net introduced by Karen Simonyan and Andrew Zisserman has been showing that the depth of a neural network is an important feature for good performance [17]. The input data set, which is split into about 80% of the data for training and 20% for testing, is a simulation of only triggered neutrino events in ORCA. The energy range of these events is from 3 GeV to 100 GeV. In reality the data of ORCA are four-dimensional(the spatial dimensions x, y, z and the time t), but because of TensorFlow the dimension of simulated data is limited on x,z and the time t. The data set "xzt" has the dimension (11x18x50) and approximately about 2 Million events. In addition, the 50% of the events are ν_μ neutrinos, the rest are ν_e neutrinos. The training procedure of a ConvNet used in the work, has been an autoencoder training, then a supervised encoder training.
For the autoencoder training, the learning rate is set on 0.001 for every tested autoencoder. For an convergence of the autoencoder performance the learning rate has been manually increased. After every epoch the mean squared error of the autoencoder loss has been calculated and saved. For the supervised encoder training, the encoder has been copied with its weights and biases from the autoencoder training. A fully-connected layer has been added to the encoder and to prevent the neural network from overfitting, a dropout value for each dense layer is set. The learning rate decay for the supervised encoder training has been fixed by a learning rate schedule. Before epoch 14 the learning rate decays by 5% per epoch down to half:

$$\text{learning rate} = 0.001 \cdot 0.95^{(\text{epoch}-1)} \tag{11}$$

After epoch 13 the learning rate decay is constant on 0.0005. After each training, no matter whether the autoencoder training or the supervised encoder training, the neural network tests itself on the test data and returns its evaluation in the form of a test point. To validate the best performance of the supervised encoder, the aim is to reach the maximum of a fully supervised training, which will be discussed later in the thesis. The fully supervised training reaches a test accuracy of 68%.

5.1 The effect of the ϵ-parameter of the optimizer ADAM on the autoencodertraining

For Deep Learning, there are various optimizers. In the following, the optimizer ADAM will be introduced.
ADAM (short for Adaptive Moment Estimation) sets the learning rate η from equation (10) not to be constant for the entire gradient step, so that the convergence of the network training is fasten up. Because of the adjustment to the current convergence ADAM allows different weights to have different step sizes. For this purpose, the decaying averages over the past gradients are followed. These calculate the mean $\vec{m}_t$ and the uncentered variances $\vec{v}_t$ of the gradient updates at step t:

$$\begin{aligned}\vec{m}_t =& \beta_1 - \vec{m}_{t-1} + (1-\beta_1) \cdot \vec{\nabla}_\omega C(\hat{F}_{\omega,\vec{b}}) & (12)\\ \vec{v}_t =& \beta_2 - \vec{m}_{t-1} + (1+\beta_2) \cdot (\vec{\nabla}_\omega C(\hat{F}_{\omega,\vec{b}}))^2. & (13)\end{aligned}$$

The two constants β_1 and β_2 are hyperparameters that define the rate, at which the gradient and the variance are exponentially decayed in the saved momentum guess. In the original paper, they are typically set to $\beta_1 = 0.9$ and $\beta_1 = 0.999$. Because the equations of $\vec{m}_t$ and $\vec{v}_t$ are heavily biased towards zero because of initialization as zero vectors at the beginning of the training, they can be written as:

$$\begin{aligned}\vec{\hat{m}}_t =& \frac{\vec{m}_t}{1-\beta_1^t} & (14)\\ \vec{\hat{v}}_t =& \frac{\vec{v}_t}{1-\beta_2^t} & (15)\end{aligned}$$

As a result, the gradient update with ADAM is defined as:

$$\omega \mapsto \omega - \alpha \frac{\vec{\hat{m}}_t}{\sqrt{\vec{\hat{v}}_t + \epsilon}}, \tag{16}$$

with step size α. The parameter ϵ functions as a numerical stability and is decided small. The paper of ADAM recommends a value of $\epsilon = 10^{-8}$ [18]. In the following, it will turn out that the parameter has an influence on the network training.
The first tested autoencoder model has been the model vgg-5-2000 with an $\epsilon = 10^{-8}$. The encoder consists of six convolutional layers and and three average pooling layers, which reduce the dimension of input layer to (2x3x5). The used filter size in the bottleneck of the model is 64, which activates about 2000 neurons in this layer. The decoders architecture is mirrored to the encoder. Figure 9 represents the autoencoder training of the model vgg-5-2000 with the train loss(lines) and the test loss(points). In the last Autoencoder epochs, the learning rate has changed several times trying to optimize the autoencoder cost function, but

it didn't effect the loss. As a result, the Autoencoder is fully trained and can't be optimized. The vgg-5-2000 with $\epsilon = 10^{-8}$ has a great autoencoder performance because it converges to a loss value of the output, compared to the input, of under 2%.

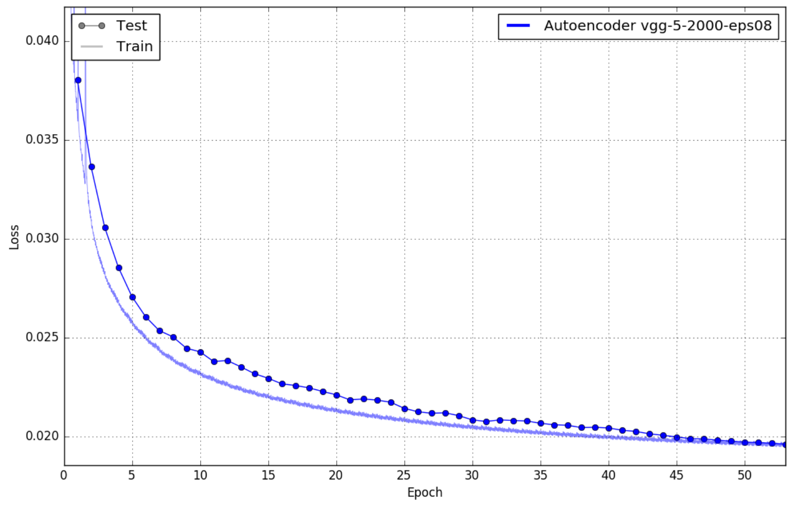


Figure 9: Autoencoder training of the autoencoder model vgg-5-2000-eps08 with test loss (points) and train loss (lines)

On recommendation of [19], the ϵ-parameter is set on 10^{-1}(see figure 10). It is noticeable that the loss of the training and the test data are almost the same. To fasten up the learning process the learning rate has been changed, which can be observed in the form of kinks of the curve in 10. In addition, the autoencoder epoch 44 is missed. A possible reason for this information loss is that the Autoencoder training was ended during the overwriting of the data because of a technical problem.

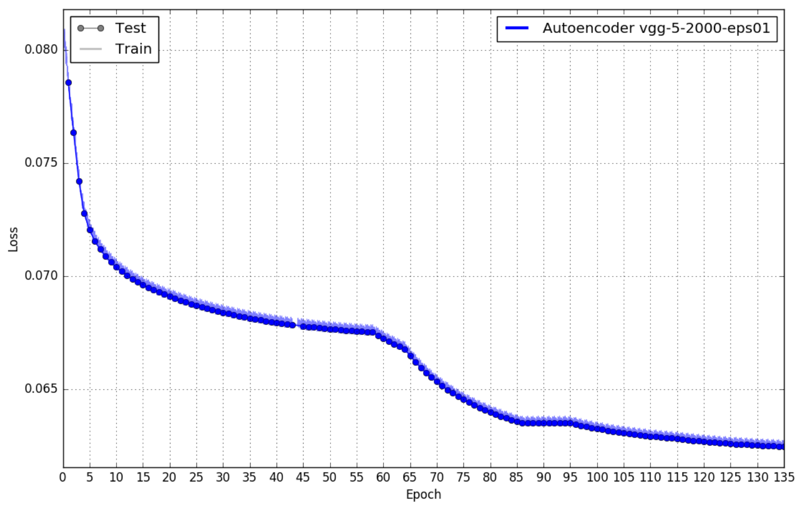


Figure 10: Autoencoder training of the Autoencoder model vgg-5-2000-eps01 with test loss (points) and train loss (lines)

In comparison, figure 11 shows the Autoencoder model vgg-5-2000 with two different settings.

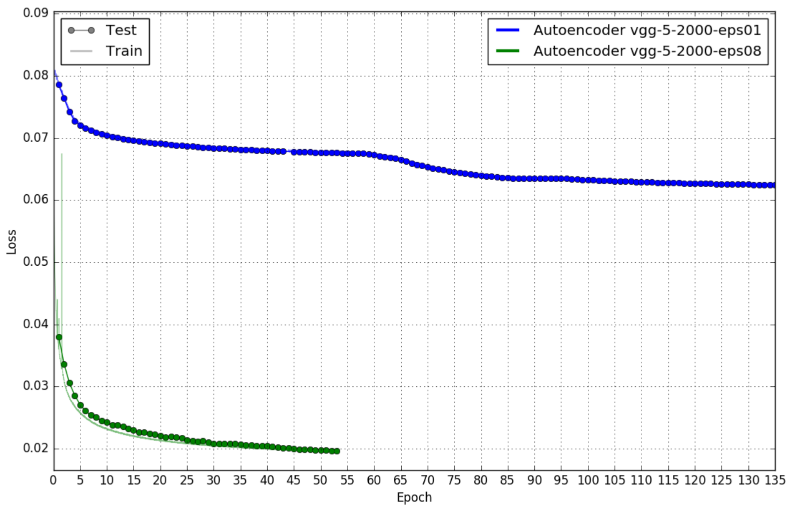


Figure 11: Variation of the ϵ-parameter on the Autoencoder vgg-5-2000(Blue curve for $\epsilon = 10^{-1}$, green curve for $\epsilon = 10^{-8}$) with test loss (points) and train loss (lines)

The autoencoder training with $\epsilon = 10^{-8}$ (vgg-5-2000-eps08) has been stopped after 53 epochs because of the constant convergence. As a result, the autoencoder with the lower epsilon has a significantly lower loss value than the other model. To evaluate their performance for

track/shower-separation, each encoder has been copied for the supervised encoder training. During the encoder training, each weight of every neuron has been frozen. At the bottleneck a dense structure has been added, which consists of three dense layers, each having a dropout value of 10%. In the end, the accuracy for an identification of event topologies has been calculated.
Figure 12 and 13 show the supervised encoder training of the vgg-5-2000 with different ϵ-parameters

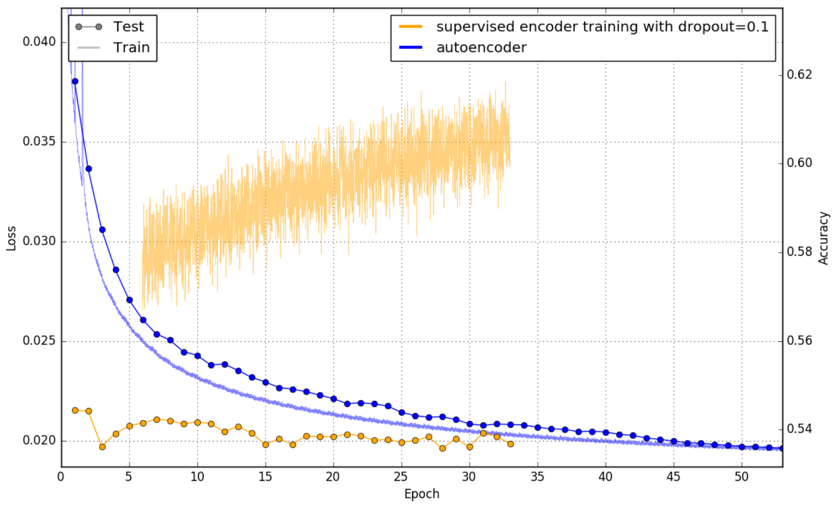


Figure 12: Supervised encoder training (orange) of vgg-5-2000 with $\epsilon = 10^{-8}$ (vgg-5-2000-eps08) with test accuracy (points) and train accuracy (lines), autoencoder training (blue) of vgg-5-2000-eps08 with test loss (points) and train loss (lines)

Significant is the encoder performance of the vgg-5-2000 with $\epsilon = 10^{-8}$ because of the strong overfitting between the test accuracy and the train accuracy. The overfitting can occur because the model memorizes the training data too well. As a result, in the test phase it cannot differentiate between a muon or a electron because the classification has been memorized very well. Therefore, the neural network predicts to early.

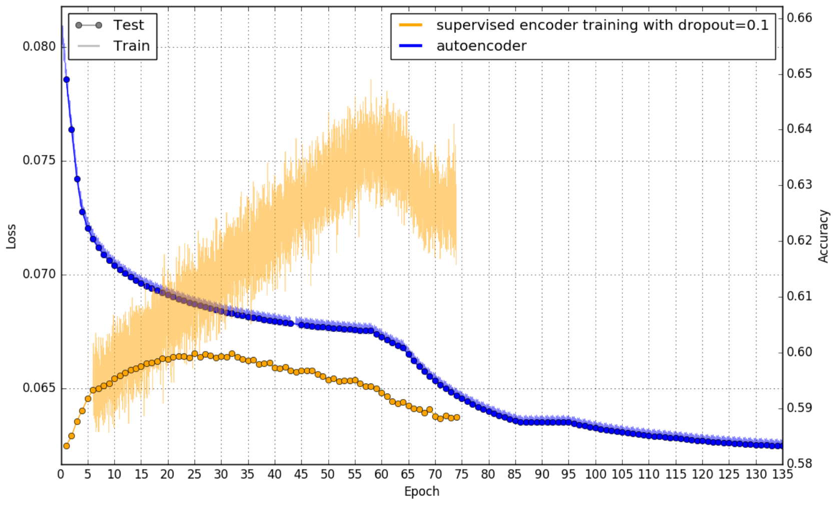


Figure 13: Supervised encoder training (orange) of vgg-5-2000 with $\epsilon = 10^{-1}$ (vgg-5-2000-eps01) with test accuracy (points) and train accuracy (lines), autoencoder training (blue) of vgg-5-2000-eps01 with test loss (points) and train loss (lines)

The test accuracy reaches a maximum at 60% at epoch 25, After that, the supervised encoder training overfits, significantly shown through the rising train accuracy and the falling test accuracy. The plot also reveals a change after the first change of the learning rate in epoch 58. In average the train accuracy reaches its maximum performance at an accuracy of 63.5% at epoch 58 and then falls. The test curve is decreasing to 59%.
As the results show, it is much wiser to use $\epsilon = 10^{-1}$ rather than $\epsilon = 10^{-8}$ for preventing a faster learning of the ConvNet, which negatively affects the performance for a track/shower identification.

5.2 Comparison of autoencoder models with and without Dropout

As the previous chapter has shown, the slower an autoencoders performance is, the better is the identification of track/shower. Due to these results, the question come up: if a dropout has been added in every convolutional layer of the vgg-5-2000 model, what will change in the performance for the track/shower-separation. This has been verified for the autoencoder model with $\epsilon = 10^{-1}$ and $\epsilon = 10^{-8}$.
Figure 14 shows the autoencoder training with dropout after every convolutional layer of the autoencoder model vgg-5-2000 with $\epsilon = 10^{-8}$, compared to the autoencoder training without dropout. The dropout value has been set to 10%.

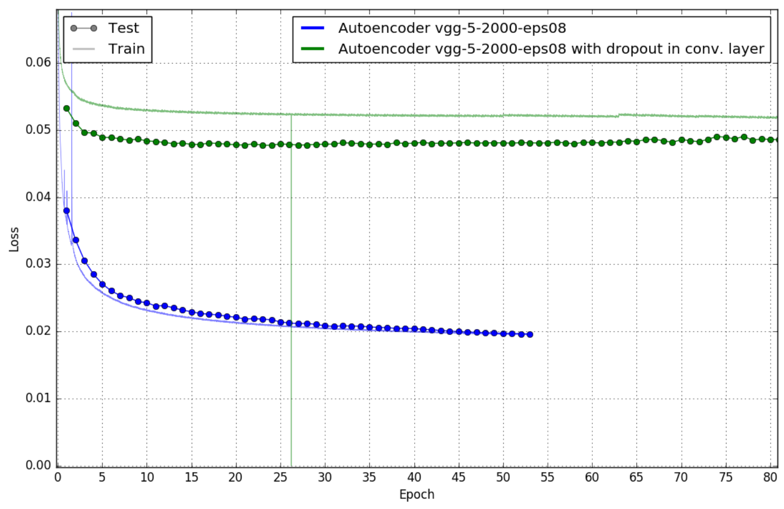


Figure 14: Autoencoder training of vgg-5-2000($\epsilon = 10^{-8}$) with dropout (green) and without dropout (blue) in every convolutional layer with train loss (lines) and test loss (points)

Compared to the autoencoder performance of the vgg-5-2000-eps08, the model with the dropout after every convolutional layer increases the loss value for about 3%. Besides that the model is fully trained after a few autoencoder epoch, because even after several learning rate changes,e.g. at epoch 63, the loss value is not decreasing.
Figure 15 shows the autoencoder training with dropout after every convolutional layer of the autoencoder model vgg-5-2000 with $\epsilon = 10^{-1}$, compared to the autoencoder training without Dropout. In addition, the Dropout value was set on 10%.

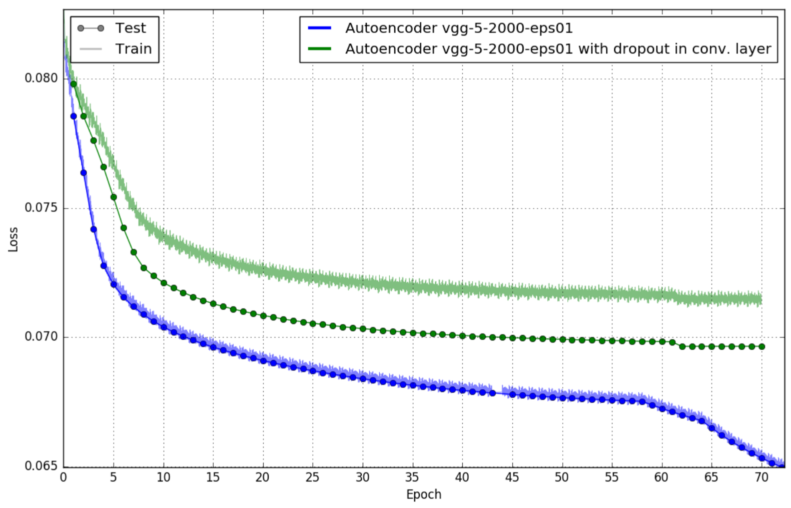


Figure 15: Autoencoder training of vgg-5-2000($\epsilon = 10^{-1}$) with dropout (green) and without dropout (blue) in every convolutional layer with train loss (lines) and test loss (points)

The statistics reveal the same observation as in the autoencoder model with $\epsilon = 10^{-8}$ that with adding a dropout after every convolutional layer the loss value increases, here about $2-5\%$. The learning rate of vgg-5-2000-eps01 network with the dropout is changed in epoch 61, significantly shown through a kink of the training and test curve.
To give a statement about their behavior in the classification of tracks and showers of the simulated ORCA data, both encoder parts istrained supervised on the classification of muon-CC and electron-CC interactions (see figure 16 and 17). The dropout value for the dense layer and after the all convolutional layers is set to 10%.

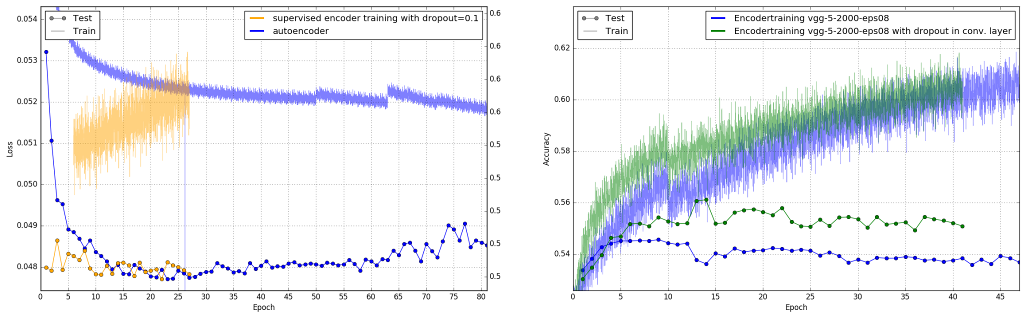


Figure 16: Autoencoder training and supervised encoder training of vgg-5-2000($\epsilon = 10^{-8}$) with Dropout in every convolutional layer(right) and encoder training of vgg-5-2000($\epsilon = 10^{-8}$) with and without Dropout in every convolutional layer(left)

Even with the added dropout, the overfitting in the vgg-5-2000-eps08 supervised encoder training cannot be prevented, but compared to the encoder performance of the autoencoder without dropout the test accuracy increased approximately over 1%(referencing to figure 16 right). The data seems to suggest that a good autoencoder with added dropout after all convolutional layer improves the behavior in the track/shower identification. A possible explanation could be that the autoencoder forgets a few parameters, so that the neural networks predicts the classification slower than the network without dropout.

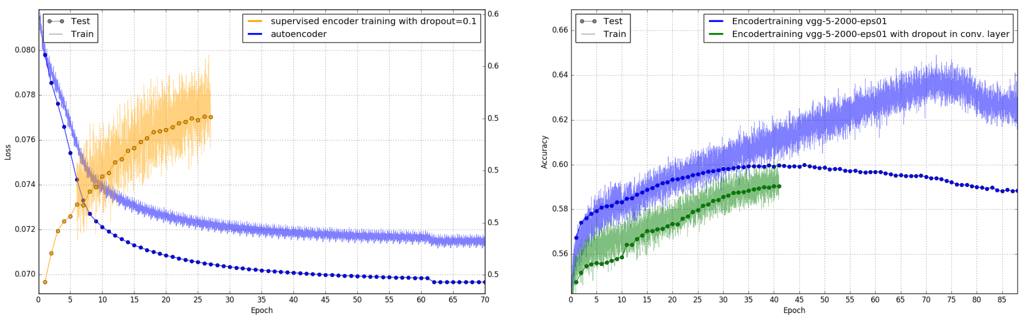


Figure 17: Autoencoder training and supervised encoder training of vgg-5-2000($\epsilon = 10^{-1}$) with Dropout in every convolutional layer(right) and encoder training of vgg-5-2000($\epsilon = 10^{-1}$) with and without Dropout in every convolutional layer(left)

From Figure 17 (left) it can be seen that the accuracy of the encoder of the autoencoder vgg-5-2000-eps01 with a dropout of 10% after every convolutional layer increases along the x-axis. Also, it can be assumed that the encoder performance in the test curve reaches a plateau at 59%. Compared to the model vgg-5-2000-eps01 without any dropout in the encoder, the convolutional neural network is learning slower probably because of same explanation as above.
To sum up all the results, by adding a dropout after each convolutional layer the autoencoder performance of the neural network decreases. If the autoencoder is learning fast, such that the network is fully trained after a few epochs, the dropout can help to minimize the overfitting between the train and test accuracy. Moreover, the performance of the encoder for a classification between muon-CC events and electron-CC events increases. However, the problem has not vanished because the overfitting is still pretty heavy. With the help of dropout, the learning process of the ConvNet can be slowed down.

5.3 Variation of the dropout value in the supervised encoder training

In the previous chapter, the behavior of the encoder of the autoencoders with a dropout of 10% in the particle identification with the simulated ORCA neutrino events have been analyzed. The question comes up: if the dropout value is set high, will the supervised encoder performance become better. This chapter will look at the different effects of the value on the classification of muon-CC and electron-CC events, only in the supervised encoder training. Figure 18 represents the different dropout value variations of the encoder training of the models vgg-5-2000-eps01 without a dropout after every convolutional layer.

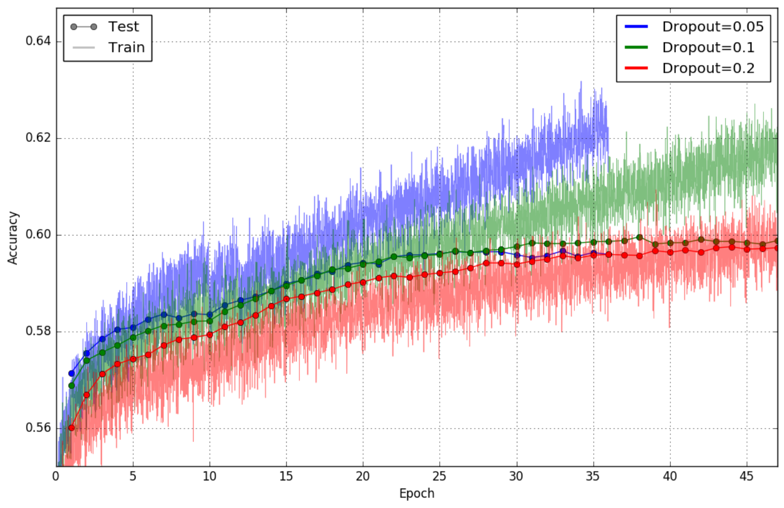


Figure 18: Variation of the dropout value (20% in red, 10% in green, 5% on blue) in the supervised encoder training of the model vgg-5-2000-eps01 with train accuracy (lines) and test accuracy (points)

The statistics in figure 18 show that at a dropout value of 5% an overfitting of the train and test accuracy can't be prevented. Besides that, with a rising value the overfitting becomes smaller, but the accuracy for the particle identification decreases. All three test curves converge to an accuracy of 60% Figure 19 represents the different dropout value variations of the encoder training of the models vgg-5-2000-eps01 with dropout after every convolutional layer.

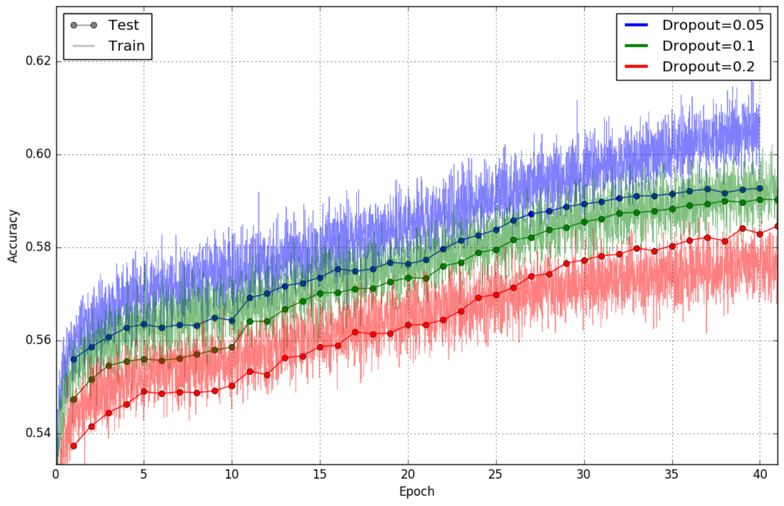


Figure 19: Variation of the dropout value (20% in red, 10% in green, 5% on blue) in the supervised encoder training of the model vgg-5-2000-eps01(dropout in every convolutional layer) with train accuracy (lines) and test accuracy (points)

The same trend can also be observed in the encoder training of the vgg-5-2000 with the dropout in every convolutional layer. Both plots negotiate the hypothesis, because the overfitting is still strong, no matter whether the dropout value is set high or low.

5.4 Variation of the bottleneck's dimension

In [20], the dimension of the bottleneck has shown an effect on identification of up-down neutrinos and of energy. In the following, the autoencoder of different bottleneck dimensions has been analyzed. Besides the autoencoder vgg-5-2000-eps01, the tested autoencoder models are vgg-5-64, vgg-5-200 and the vgg-5-4000. The vgg-5-64 has ten convolutional layers and its bottleneck is reduced to (1x1x1) by five average pooling operations. The filter size in the bottleneck is 64, so that the bottleneck consists of 64 neurons. The vgg-5-200 has eight convolutional layers and its bottleneck is reduced to (2x2x2) by four average pooling operations. Because of the filter size of 25, the number of neurons in the bottleneck is 200. The biggest neuron model is the vgg-5-4000, which has seven convolutional layers and is reduced to (4x3x5) by three average pooling layers. This model has the same filter size as the vgg-5-2000 and has approximately about 4000 neurons. Figure 20 shows all autoencoder performances.

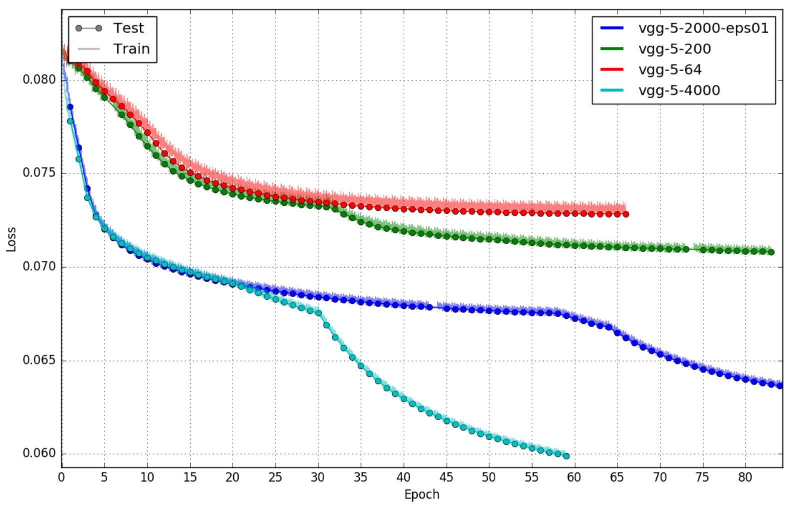


Figure 20: Autoencoder training of autoencoder models vgg-5-64(red), vgg-5-200(green), vgg-5-2000-eps01(blue) and vgg-5-4000(cyan) wih test loss (points) and train loss (lines)

After several changes of the learning rate to fasten up each autoencoder training, the more neurons are in the bottleneck, the better is the test loss of the autoencoder. Tabular 1 shows the autoencoder test loss and the test accuracy of each model.

Model name	test loss (autoencoder training)	test accuracy (supervised encoder training)
vgg-5-64	under 7.4%	under 58%
vgg-5-200	7.1%	60%
vgg-5-2000-eps01	under 6.3%	60%
vgg-5-4000	6%	59%

Table 1: Test autoencoder loss and the supervised Encoder test accuracy of the models vgg-5-64, vgg-5-200, vgg-5-2000-eps01 and vgg-5-4000

In conclusion, the dimension of the bottleneck is disproportionately to the loss of the autoencoders because the bigger the amount of neurons is, the smaller is the loss value. In the following, the supervised encoder performance of each bottleneck-model will be analyzed.

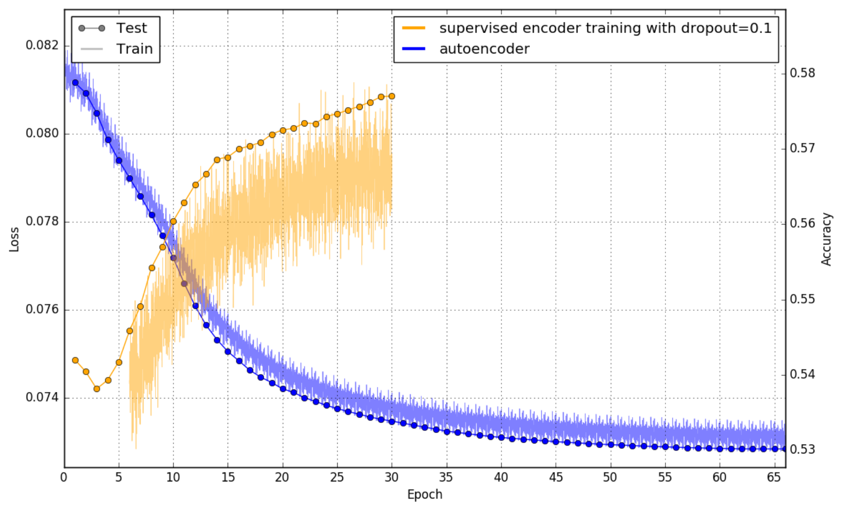


Figure 21: The supervised encoder training (orange) of vgg-5-64 with test accuracy (orange points) and train accuracy (orange lines) and the autoencoder training (blue) of vgg-5-64 with test loss (blue points) and train loss (blue lines)

It can be seen from figure 21 that the autoencoder converges to a loss value of under 7.4%. In addition, the autoencoder becomes better along the epochs, because after 65 epochs it reduces its loss by 0.8%. In the supervised encoder training, the convolutional network shows a better performance in the test phase than in the training and reaches at epoch 30 an accuracy of under 58%. The reason,why the supervised encoder training for the particle identification was stopped at this point, is that the vgg-5-64 has the lowest accuracy in the track/shower-separation among the other models.

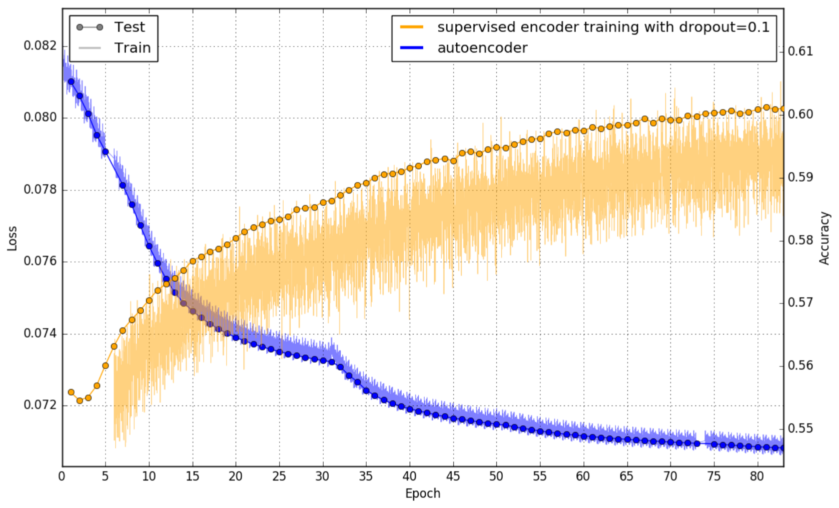


Figure 22: The supervised encoder training (orange) of vgg-5-200 with test accuracy (orange points) and train accuracy (orange lines) and the autoencoder training (blue) of vgg-5-200 with test loss (blue points) and train loss (blue lines)

After 80 autoencoder epochs, the vgg-5-200 (seen in figure 22) reaches a loss value of about 7.1%. Significantly at the epoch 31, the learning rate was changed which causes a drop in the loss curve. The same performance behavior like in the ConvNet vgg-5-64 can be observed in the supervised encoder training, the test curve is higher than the train curve. At the end of the supervised encoder training, the network reaches an accuracy of about 60%.

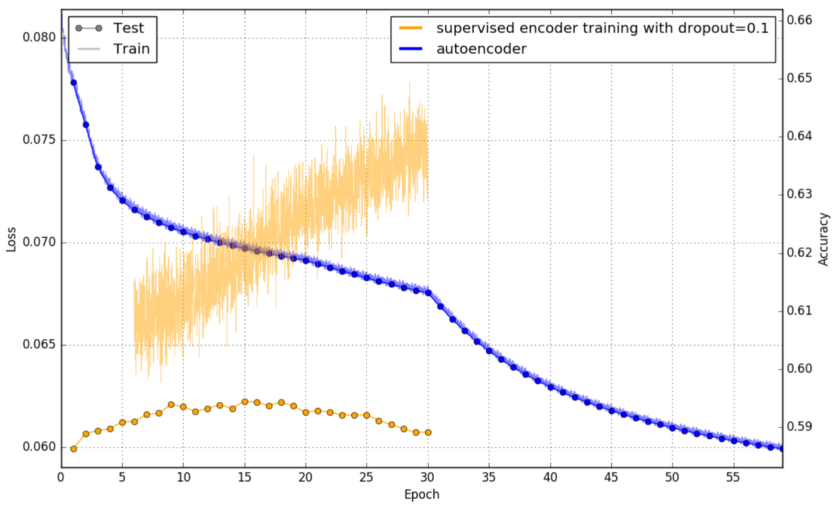


Figure 23: The supervised encoder training (orange) of vgg-5-4000 with test accuracy (orange points) and train accuracy (orange lines) and the autoencoder training (blue) of vgg-5-4000 with test loss (blue points) and train loss (blue lines)

In the autoencoder training of the ConvNet vgg-5-4000 (refers to figure 23) the learning rate has been changed in the autoencoder epoch 10 and 30 to fasten up the learning process and to reduce the loss which reaches at epoch 60 6%. But the network is not fully trained yet because it does not converge to an exact loss value. After all, this autoencoder model shows the best performance in the autoencoder training. However, there is a strong overfitting between the training accuracy and the test accuracy in the supervised encoder training. It only reaches its maximum at 59%. Compared to the vgg-5-200, the PID performance of this convolutional neural network is worse than the performance of a model with a much smaller bottleneck. Due to the high dimension of the bottleneck the autoencoder extracted too many features for the identification between an electron and muon and memorizes them very well. As a result, the network predicts too early. In addition, the strong overfit can be compensated with a higher dropout value to prevent the network from good memorizing the training data. To make a statement, which autoencoder model,vgg-5-200 or vgg-5-2000-eps01, shows a better performance, the two models are compared with their fully supervised training performance. In the fully supervised training, the whole encoder and dense structure are completely built new without any known weights or biases. Figures 24 and 25 represent each model with their fully supervised trained performance.

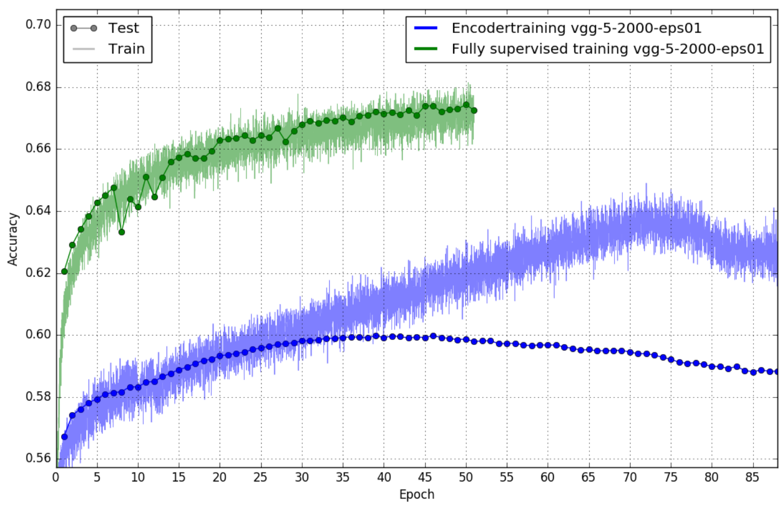


Figure 24: Supervised encoder training of vgg-5-2000-eps01 (blue) compared to its fully supervised performance (green) with test accuracy (points) and train accuracy (lines)

The fully supervised performance almost reaches an accuracy of 68% in the separation of muon-CC events and electron-CC events. That is approximately a difference of 8% to the supervised encoder training of the autoencoder model vgg-5-2000-eps01.

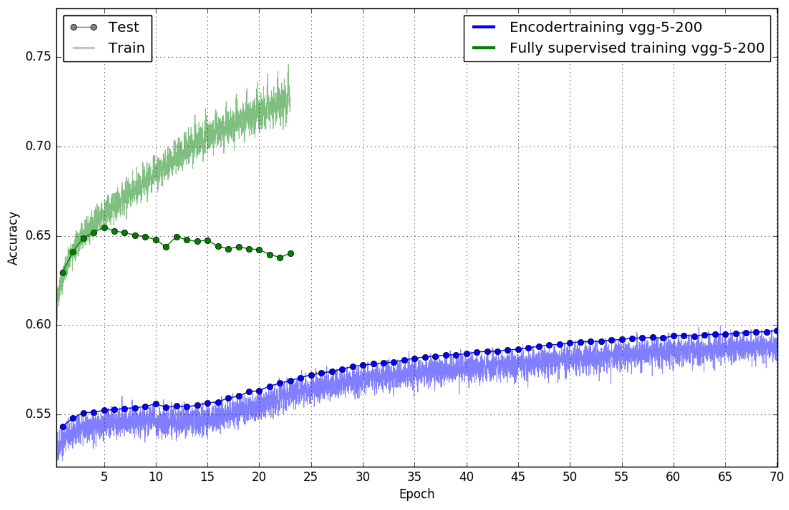


Figure 25: Supervised encoder training of vgg-5-200 (blue) compared to its fully supervised performance (green) with test accuracy (points) and train accuracy (lines)

But the fully supervised performance of vgg-5-200 shows an overfitting between the test and the training accuracy after the encoder epoch 5. Because of the low number of neurons, it seems like that it memorizes the training data too well and gives a false statement in the particle identification. Besides that, the networks accuracy is reached at about 65%, which is only 5% above the accuracy of the supervised encoder performance of the same model. In conclusion, the decision is not easy because the autoencoder of the vgg-5-200 is not fully trained. But only from comparing the autoencoders the vgg-5-200 is better than the vgg-5-2000-eps01 because the test accuracy is better than the training accuracy.

5.5 Robustness analysis of the autoencoder model vgg-5-2000-eps01

Robustness analysis are important to observe how the ConvNet is working with systematic errors, which can occur in reality, and how good it can compensate this difficulties. For the robustness analysis the autoencoder model vgg-5-2000-eps01 is taken because during the time of these tests the model has been almost fully trained and has been analyzed. To test the robustness of the autoencoder model vgg-5-2000-eps01, a new autoencoder has been trained on data which are manipulated. The data set has a reduced quantum efficiency of Photomultipliers of the DOMs by up to about 40% with a gradient in x-direction. After the autoencoder training, each weight of the neuron from the encoder has been frozen in the supervised encoder training. Moreover, the same manipulated data has been used for this procedure. Figure 26 shows the autoencoder training and the supervised encoder training of the autoencoder model vgg-5-2000-eps01, that works best compared to the other models in the classification of neutrino event topologies.

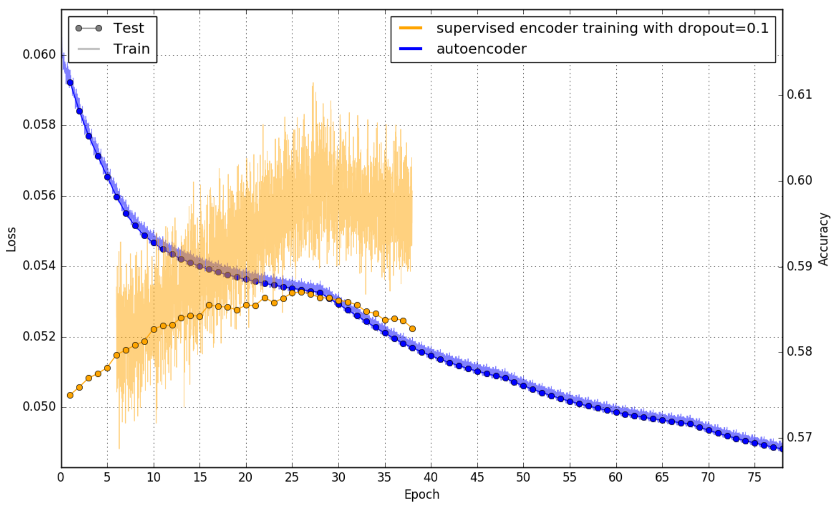


Figure 26: Autoencoder (blue) with test loss (blue points) and train loss (blue lines) and supervised encoder training (orange) of vgg-5-2000-eps01 test accuracy (orange points) and train accuracy (orange lines) on manipulated data

In the beginning of the autoencoder training, the learning rate has been set to 0.001. To fasten up the learning process of the autoencoder, the learning rate has been changed three times. However, the trend of the autoencoder is not converging to a significant loss value, so the autoencoder is not fully trained yet. The maximum of the encoder in the particle identification is reached after 25 epochs. After that, the training accuracy, as well as the test accuracy is decreasing.

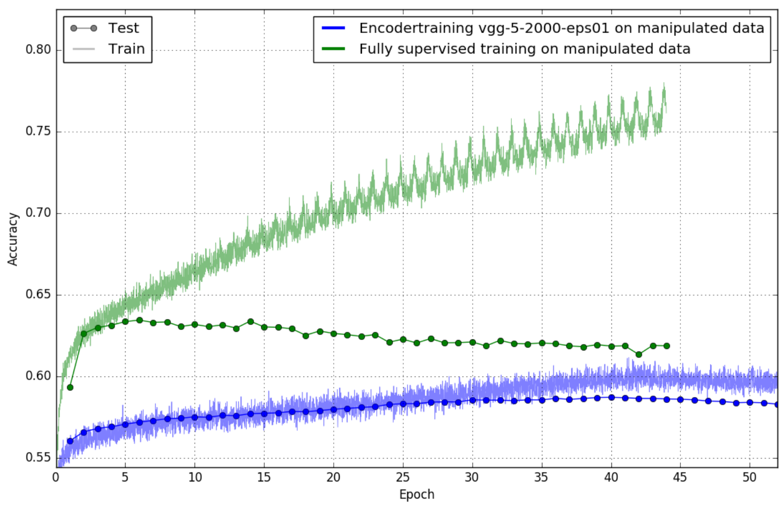


Figure 27: Comparison of the supervised encoder training of vgg-5-2000-eps01 (blue) and the fully supervised trained vgg-5-2000-eps01 (green) on manipulated data set with test accuracy (points) and train accuracy (lines)

The statistic of figure 27 show that the unsupervised learning approach with autoencoder training and supervised encoder training gives a better accuracy in the identification of muon-CC events and electron-CC events than the accuracy with the fully supervised training. Indeed, the encoder of the autoencoder gives a lower test accuracy, but it prevents the neural network from overfitting. The overfitting of the fully supervised training is heavy and with increasing number of epochs the train and test accuracy move away from each other, the overfitting of the convolutional neural network increases.
Next, a whole network was trained with supervised learning method to see which method works better with harder conditions. For maximizing the full performance in the track/shower identification and to compare the robustness analysis to maximum of the classification accuracy, the model was fully trained on correct data with the fully supervised learning method. Figure 28 shows in comparison all three performances.

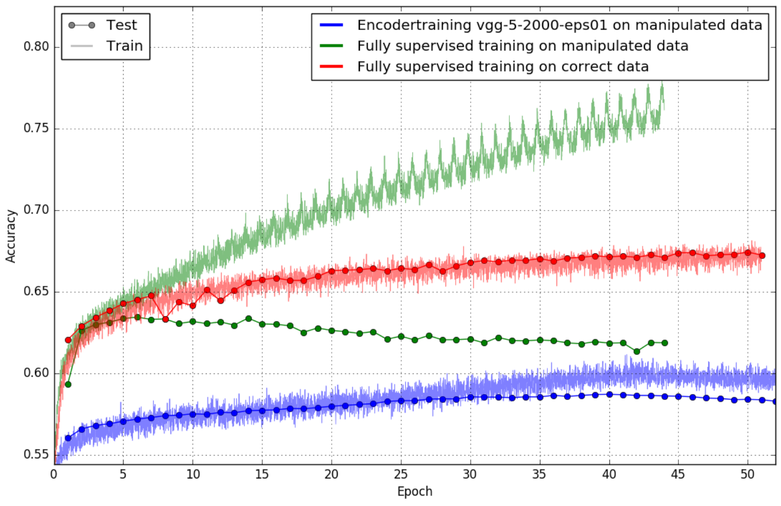


Figure 28: Comparison of the supervised encoder training of vgg-5-2000-eps01 (blue), the fully supervised trained vgg-5-2000-eps01 (green) on manipulated data set and the fully supervised trained vgg-5-2000-eps01 (red) on correct data set with test accuracy (points) and train accuracy (lines)

The encoder performance of the vgg-5-2000-eps01, trained with manipulated data, deviates about 10% from the fully supervised accuracy, trained with correct data. According to figure 28, a fully supervised training of vgg-5-2000-eps01 with manipulated data isn't compared to the accuracy of the fully supervised training with correct data wise because on the one hand of the overfitting and on the other hand of the lower test accuracy. Because of time issue, a missing case, which is a fully supervised network trained on not manipulated data but used on manipulated data, couldn't be observed. In [20], this robustness analysis has been done on up-down classification and on energy. The results of this thesis show that the unsupervised learning has a higher accuracy than the accuracy of the fully supervised network, trained on not manipulated data, used on manipulated data. In conclusion, it could be expected that the accuracy of the unsupervised learning for track/shower separation on manipulated data is better.

6 Conclusion

The aim of this thesis is to achieve a good identification of event topologies of neutrinos, like track and shower differentiation, in the neutrino detector KM3Net/ORCA. Therefore, the unsupervised learning technique has been used to train the convolutional neural network. For a training of the whole network, the first step to reach the goal has been to train an autoencoder. After that the encoder of the autoencoder has been retrained and its weights and biases have been frozen, before a dense structure has been added. With the help of the dense layers, a classification of muon-CC events and electron-CC events can be done. To train the network, the data set "xzt" has been split in a train set, which are 80% of the events, and a test set. In addition, the neutrino events of the whole data set are 50% muon-CC events and 50% electron-CC events. As a result, if the end performance accuracy has been near 50%, it can be concluded that the neuron is guessing the topology of the event.
For optimization of the weights and biases of the network, optimizers like ADAM are used to reduce the loss function in each optimization epoch. The original paper of ADAM recommends to set the ϵ-parameter, which prevents the parameter update from dividing zero, to 10^{-8}. However, the results show that if this parameter is decided to be very small, the autoencoder performance becomes significantly well, but the classification of the topologies becomes much worse. A possible reason could be that the network is learning to fast and memorizes the crucial features too good. Resulting from that, the network predicts the topology to early and the overfitting occurs. The first hypothesis has been set: the worser is the autoencoder performance, the better is the supervised encoder performance.
To prevent the network from overfitting, as seen as in the variation of the ϵ parameter of the optimizer, a dropout has been added after every convolutional layer. The statistics reveal that the autoencoder performance decreases by adding dropout after each convolutional layer. Besides that, the performance of the encoder for a classification between muon-CC events and electron-CC events increases. However, the problem has not vanished because the overfitting is still pretty heavy. With the help of dropout, the learning process of the ConvNet can be slowed down.
After adding the dropout in the autoncoder training, the value of dropout has been changed in the supervised encoder training, hoping to optimize the overall performance for the track/shower recognition. The variation of the value has been executed in the supervised encoder training of the ConvNet without dropout in the convolutional layers as well as in the supervised encoder training of the ConvNet with dropout in the convolutional layers. The statistics show that a variation of the dropout value does not improve the recognition, because the overfitting between the train accuracy and the test accuracy is high.
After that the next approach to optimize the event topology identification has been to change the dimension of the bottleneck, in order to find out how many crucial features of the input data are needed for a good performance. The tested models were the vgg-5-64, the vgg-5-200, the vgg-5-2000-eps01 and the vgg-5-4000. From the autoencoder training of each

autoencoder model, the more neurons are settled in the bottleneck, the lower is the test loss of the autoencoder. However, the only autoencoders, which have a great performance in the track/shower identification, are the vgg-5-2000-eps01 and the vgg-5-200. From the results of the bottleneck studies, a optimal size of the bottleneck could be between 200 and 2000 neurons because their performance in the supervised encoder training are comparable.
Furthermore, the unsupervised learning approach is a good way to handle systematic difficulties of the neutrino detector ORCA, for example the quantum efficiency of the Photomultipliers. In reality, the photomultipliers can show differences in their detection of the Cherenkov light, emitted by secondary charged particles, because each string will not be released at the same time. Because of the great with the training on manipulated data in the robustness analysis compared to the fully trained network, the unsupervised learning approach can simplify the analysis with systematic errors.
In conclusion of the results from this thesis, the unsupervised learning method for the track/shower identification of neutrino events in the neutrino telescope ORCA should be used and modified in the future.

7 Appendix

List of Figures

8 Bibliography

References

[1] Evan Racah, Seyoon Ko, Peter Sadowski et al.: Revealing Fundamental Physics from the Daya Bay Neutrino Experiment using Deep Neural Networks,
https://arxiv.org/pdf/1601.07621.pdf.

[2] Dan Guest, Kyle Cranmer, Daniel Whiteson, Deep Learning and Its Application to LHC Physics,
https://arxiv.org/pdf/1806.11484.pdf.

[3] Letter of intent for KM3NeT 2.0, J. Phys. G: Nucl. Part. Phys. 43 (2016) 084001 (130pp).

[4] new ORCA Layout, Plot from Patrick Lamar.

[5] Zuber, Kai: Neutrino Physics. Second Edition, Taylor & Francis Group, LLC, 2012.

[6] N.Schmitz. Neutrinophysik. Teubner, Stuttgart, 1997.

[7] D. V. Forereo, M. Tórtola and J. W. F. Valle. Global status of neutrino oscillation parameters after Neutrino 2012 Phys. Rev. D 86, 073012(2012), *https://arxiv.org/abs/1205.4018.*

[8] picture of the neutrino mass hierarchy
http://www.staff.uni-mainz.de/wurmm/wurm-home/mass-hierarchy.png.

[9] Francis Halzen and Uli Katz: The Era of Kilometer-Scale Neutrino Detectors. Review Article ID 680584, *http://dx.doi.org/10.1155/2013/680584.*

[10] Andrey Karpathy, Fei-Fei Li, Justin Johnson: CS231n Convolutional Neural Networks for Visual Recognition,
http://cs231n.github.io/.

[11] Keiron O'Shea and Ryan Nash. Introduction to Convolutional Neural Networks, *https://arxiv.org/abs/1511.08458.*

[12] Deep learning. Convolutional Neural Network
https://de.mathworks.com/solutions/deep-learning/convolutional-neural-network.html.

[13] Arden Dertat:Applied Deep Learning - Part 3: Autoencoders,
https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

[14] Keras:The Python Deep Learning library, *https://keras.io.*

[15] Tensorflow. *https://www.tensorflow.org/.*

[16] GitHub. *https://github.com/Jelmilc/Km3-Autoencoder.*

[17] Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Visual Recognition,
https://arxiv.org/abs/1409.1556.

[18] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization, *https://arxiv.org/abs/1412.6980.*

[19] Stefan Reck. *private communication.*

[20] Stefan Reck. *Master Thesis*, October 2018.

Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe.

Erlangen, den 30. Juli 2018

Jelena Mila Ćelić