Bachelorarbeit

15. September 2022

Low Energy Neutrino Event Reconstruction Using Graph Neural Networks

Contents

1 Introduction

The IceCube Neutrino Observatory is the largest neutrino experiment ever conducted. It consists of over 5000 light-detecting digital optical modules (DOMs), which are burried in a volume of over $1\,\mathrm{km}^3$ in the ice of Antarctica. Such an experiment with as many individual detectors naturally yields tons of raw data, hence manual interpretation/analysis is not appropriate. One way to automate analysis of the data utilizes Machine Learning (ML) models, more specifically in our case the use of Graph Neural Networks (GNN). These networks model the sets of all DOMs, which are triggered together (close in time), as point clouds which are then connected into a graph representing the geometry and data gained from the measurement. For the interpretation/evaluation the data fed to the GNN originally only stemmed from the measurements of the DOMs. In this thesis an effort is made to feed additional data to the network, which described local properties of the ice, in order to increase the performance of the network.

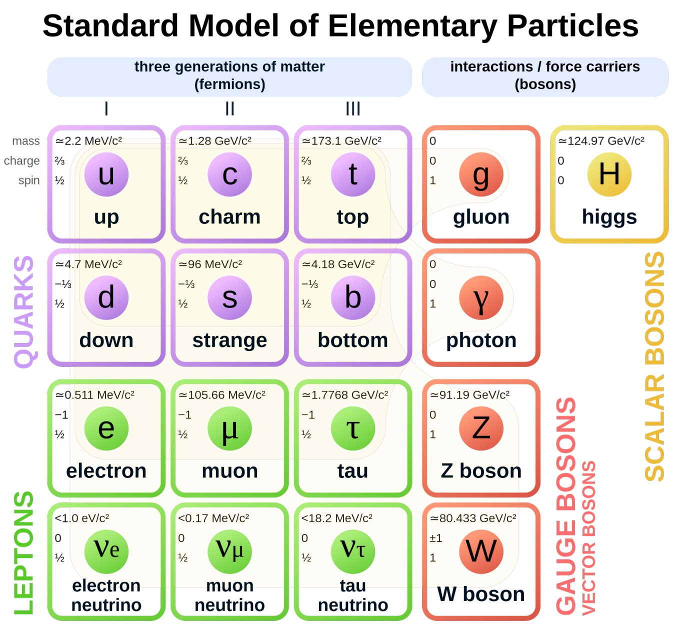
2 Particle Physics

Search for the building blocks of matter has long been an interesting field for many scientists and philosophers. For a long time humans believed elements to be fundamentally different from another. Actually not too long ago, in the late 1800s-1900s it was discovered that matter is made from atoms, which in turn consist of protons, neutrons and electons. The idea that these, long thought of as elementary particles, are dividable even further into quarks has not even been around for more than 60 years at the time this work is written.

2.1 Standard Model

The Standard Model of particle physics (shown in figure 1) is a theory to describe all currently known elementary particles. It also describes three of the four fundamental forces we have yet discovered, them being electromagnetic, weak and strong interactions, with gravity being the force that was not been able to be integrated consistently into the theory.

Although the Standard Model explains most phonomena fully and logically, gravity, dark matter, as well as a couple other phenomena, are not able to be properly explained/modelled by the Standard Model. The model is a great achievement of science, and new particles continue to be discovered by great scientists to this day. But the weaknesses of it mark the theory as incomplete.



Standard Model of Elementary Particles
three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
mass
charge
spin
≃2.2 MeV/c² ⅔ ½ u up
≃1.28 GeV/c² ⅔ ½ c charm
≃173.1 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≃124.97 GeV/c² 0 0 H higgs
≃4.7 MeV/c² −⅓ ½ d down
≃96 MeV/c² −⅓ ½ s strange
≃4.18 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≃0.511 MeV/c² −1 ½ e electron
≃105.66 MeV/c² −1 ½ μ muon
≃1.7768 GeV/c² −1 ½ τ tau
≃91.19 GeV/c² 0 1 Z Z boson
<1.0 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≃80.433 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
VECTOR BOSONS
SCALAR BOSONS

Figure 1: The Standard Model of particle physics.

2.2 Neutrinos

Neutrinos, located at the bottom of the Standard Model, are uncharged leptons first hypothesized by Wolfgang Pauli in 1930 in order to fix the observed possible variation of electron energy during beta decay. He reasoned, that since energy must be conserved, the 'missing' energy is carried by a particle previously not detected/thought of by anyone.

$$n^0 \longrightarrow p^+ + e^- \tag{1}$$

Therefore he extended the known equation of the beta decay (above) with his new electron neutrino particle, which solved the until then continous beta decay spectrum.

$$n^0 \longrightarrow p^+ + e^- + \bar{\nu}_e^0 \tag{2}$$

Although neutrinos are present everywhere in huge numbers brings up the questions why they have not been detected before. This is due to them being not only exeptionally tiny, but they also only interact via the weak force (as well as gravity) over short distances, which makes them very hard to detect. The estimated cross section is $\sigma \sim 10^{-38}\,\mathrm{cm}^2 = 10^{-42}\,\mathrm{m}^2$ is absolutely tiny, even compared to the standard unit of particle physics, the `barn`, of which $1\,\mathrm{b} = 100\,\mathrm{fm}^2 = 10^{-28}\,\mathrm{m}^2$.

2.3 Neutrino Oszillation

In the 1960s the solar neutrino flux was measures in the Homestake experiment performed by Ray Davis and John N. Bahcall. They measured the amount of neutrinos interacting inside a $380\,\mathrm{m}^3$ tank of perchloroethylene. The solar neutrinos originate from the nuclear fusion process, mostly from the pp-chain (proton-proton).

$$p^+ + p^+ \longrightarrow {}^2H + e^+ + \nu_e \tag{3}$$

The observed neutrino flux was only about $\frac{1}{3}$ of the theoretically expected value. This problem puzzled scientists for decades until it was resolved around 2002 through the explanation of neutrino oscillation. Neutrino oscillation is a flavour transition of the neutrino, where any neutrino given will oscillate/change its flavour (electron, muon, tau) when traveling through space. Therefore statistically only about every third neutrino will be a electron neutrino, and only those are being measured by the Homestake experiment - explaining the measured flux deficit.

TODO: Neutrino Oscillation figure

TODO: Neutrino Oscillation explanation

2.4 Cosmic Rays

TODO:

3 IceCube

3.1 Location and Structure

The IceCube Neutrino Observatory is located at the geographic South Pole, where is makes use of the abundance of natural ice as material for the incoming particles to react with. Over an volume of over $1\,\mathrm{km}^3$ 5160 Digital Optical Modules (DOMs) have been burried over a span of 10 years. The DOMs are positioned on 86 individual strings at depths between 1450 m and 2450 m (shown in figure 2). The strings are placed about 125 m apart horizontally and all DOMs are spaced about 17 m vertically. In the center of the detector, where the ice is clearest, additional strings have been placed, which increases the detector density in that area called DeepCore.

The detector was built to study neutrino and muon reactions, which is why it is build so large. Not only to increase the probability of a neutrino reaction within its borders, which is minimal (see section 2.2), but also to provide enough space in order to hopefully capture full traces of reactions. Which can be spread out in space, especially cascase muon reactions with high energies can involve over 100 DOMs.

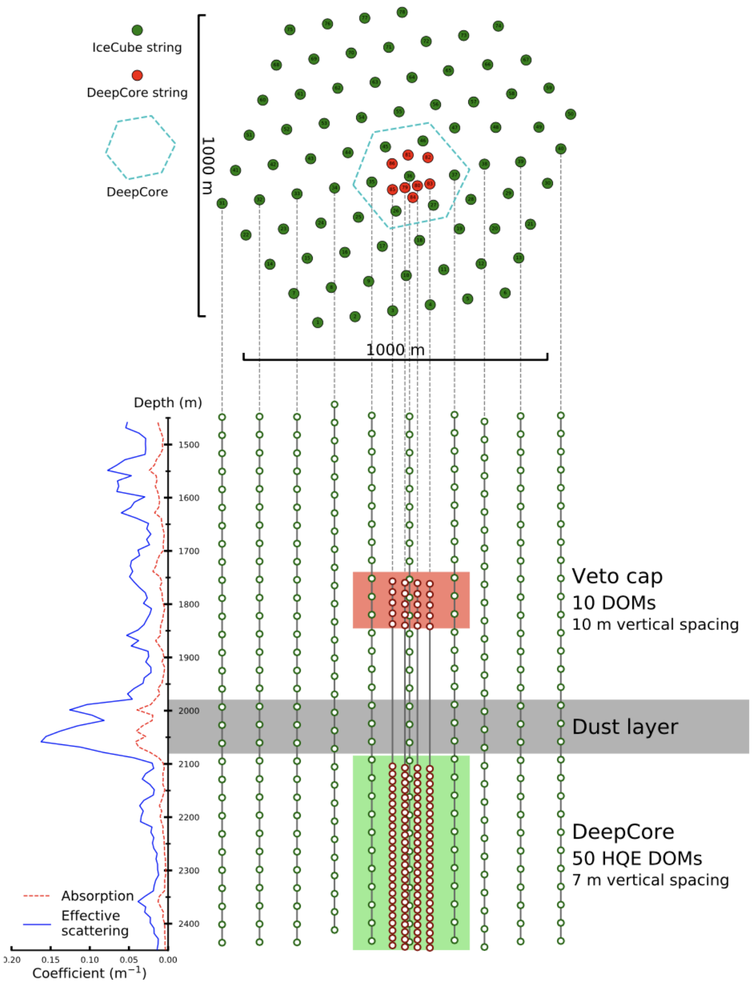


Figure 2: The structure of IceCube.

3.2 Noisy Data

The data that IceCube DOMs records is natually noisy, since the detectors are very sensitive, many of them trigger due to noise (> 1000 Hz). These events are of course not wanted, and many ways of filtering are implemented to combat noise in the recorded data.

At the lowest level the condition of a minimum number of DOMs has been implemented, which all have to fire within a specified time interval (8 DOMs within 5 μs). This is why no events in the databases used lateron in this work contain events with fewer than 8 sensors involved.

Additionally a local filtering is applied to all events, it assured and filteres events that happen closeby (150 m radius with a given time frame of 1000 ns). This makes an effort to assure that all DOMs recorded in a single event are correlated and for example a single noise-influenced DOM at the outside of the detector is not included in an actual event at the center of the detector.

TODO: Check with Rasmus whether all above is actually true/accurate

4 Machine Learning

The field of Machine Learning studies mathematical models that are able to 'learn'. These mathematical models, or networks, are systems of equations, which map a set of inputs onto a set of outputs. The process of mapping input to output may involve many steps, decided on and designed by the network architect.

4.1 Simple Neural Networks

A simple Neural Network consists of a fixed number of layers with a fixed number of nodes (can vary by layer). The layers between the input and output layers are called hidden layers. The more exist, the deeper the network gets. As represented in figure 3, the layers are interconnected by edges, which connect all nodes from the previous layer to all nodes of the next layer. This structure and dataflow is supposed to simulate what happens in the brain. As long as edged only forward data into one direction the network is called feed-forward.

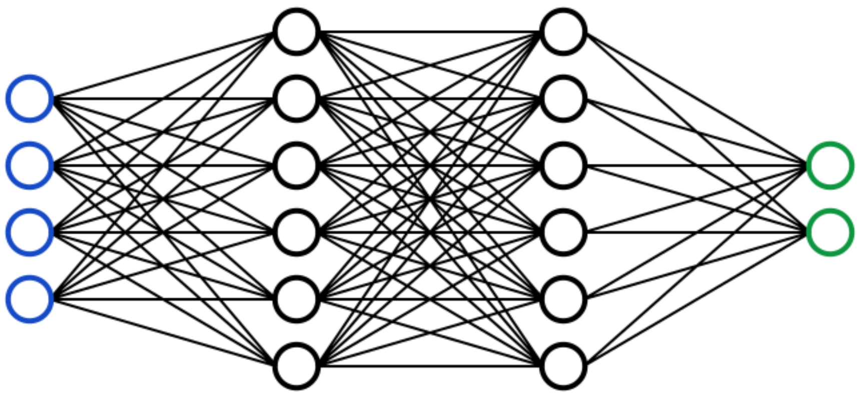


Figure 3: A graphical representation of a simple neural network.

To more mathematical describe how the network operates: Data in nodes x_i of the previous layer is transformed/scaled by the edges weight w_{ji} and offset by the bias b_i of the next node. Therefore the data h_i of the next layer is given by:

$$h_i = \sum_j w_{ji} x_j + b_j \tag{4}$$

In matrix notation the transformation for all nodes in a layer can be expressed in one formula, where for any given layer index l: h_l is the data values, W_l is a matrix of edge weights and b_l is the bias (offset).

$$h_{l+1} = W_l h_l + b_l \tag{5}$$

Therefore a simple NN can be thought of as a repeated process of matrix-vector multiplications and vector additions. This algorithm produces quite flexible behaviour, as long as the size of the network is adequate to the problem at hand. But on second thought this process can only model linear behaviour, since it only contains linear operations, which is a limitation that will be resolved in the next section.

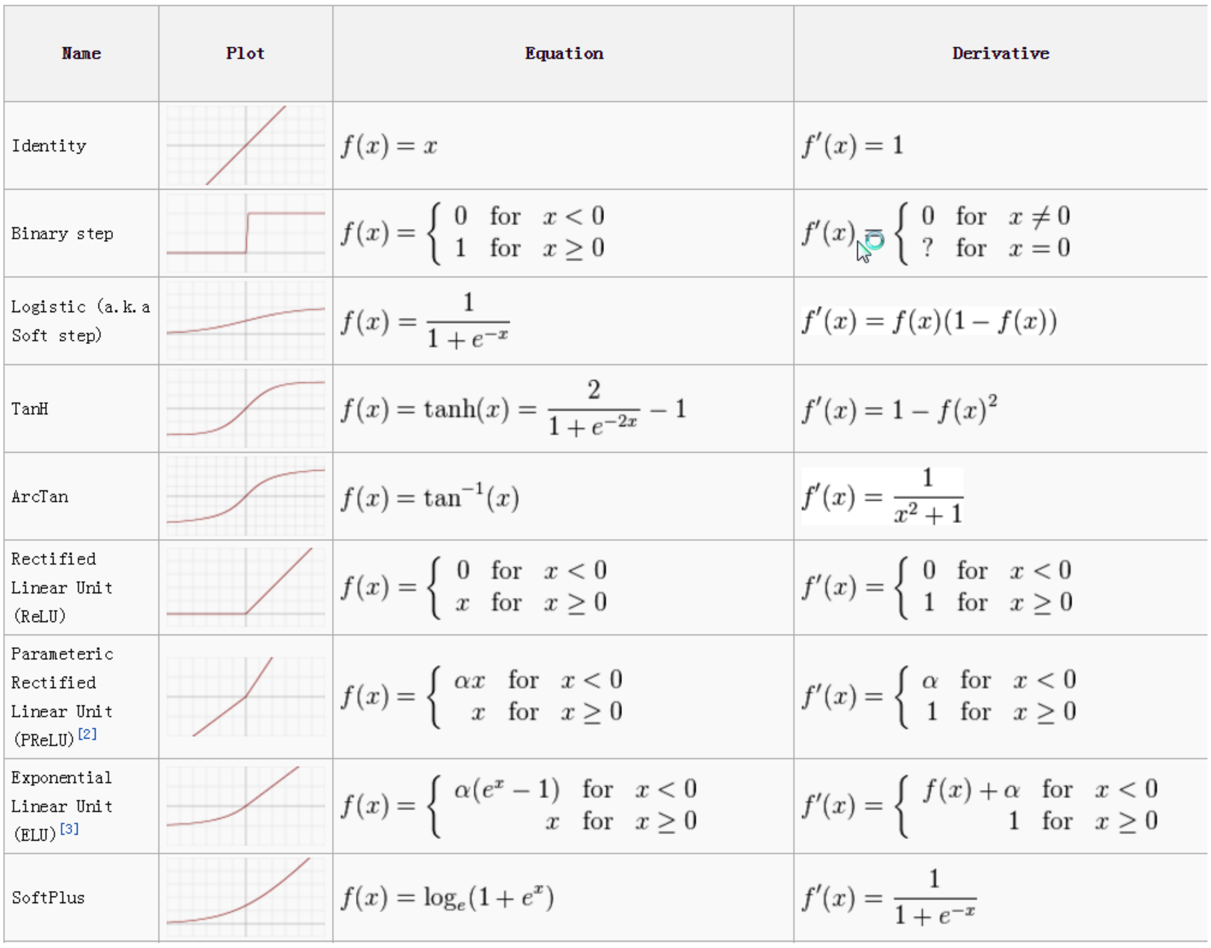
4.2 Activation Layers

In order to introduce some non linearity to the network, Activation Layers are applied to the data at selected layers. Activation Layers are differentiable but non-linear functions, we will see in a next chapter as to why they have to be diffentiable. The choice of what kind of Activation Layers to use and how many lies with the network architect.

TODO: Replace collection of activation functions

4.3 Learning

The type of NNs we discuss relevant to this work all learn by the principle of Supervised Learning, in which the network is provided a set of training data, which is fed into the network and its output is then compared to the expected result (truth), which has to be known beforehand. This limits the applicability of this approach to certain problems, for example the best next chess move is not easily



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 4: A collection of common Actionation Functions.

known, therefore this approach would not be feasable for a chess problem. By comparing the result with the truth a Loss is calculated by the Loss Function. This loss represent the 'wrongness' of the network. It is then used by the Optimizer, an algorithm that adjusts the weights and biases (parameters) of the network in such a way, that the loss decreases and therefore the results are getting closer to the truth each iteration. This adjustment of parameters based of the loss is only possible because all the operations the network performs are differentiable and the gradient of the 'loss landscape' (n-dimensional space) can be calculated.

4.4 Loss Functions

There are many Loss Functions a network architect can choose from. Depending on the characteristics of the function, the networks learning behaviour changes, as we understand in the following sections.

In general a loss function produces a minimal value (0) if the networks output y_i matches the expected truth $\hat{y}_i$.

Mean Absolute Error / L1 Loss

L1 Loss or Mean Absolute Error Loss behaves exactly as its name states. It returns the absolute difference beween the network output and the truth, which is averaged over the individual training data values.

$$MAE = \frac{\sum_i^n |y_i - \hat{y}_i|}{n} \tag{6}$$

Mean Squared Error / L2 Loss

L2 Loss or Mean Squared Error Loss behaves different to L1 Loss in the way that it does not calculate the absolute difference, but instead squares the difference of each value before averaging.

$$MSE = \frac{\sum_i^n (y_i - \hat{y}_i)^2}{n} \tag{7}$$

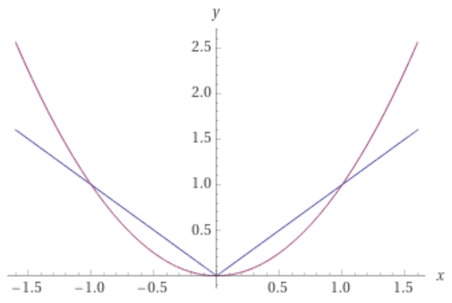


Figure 5: A comparison of L1 and L2 Loss.

As seen in the figure above, the behaviour of L1 is different when compared to L2. This has consequences for the learning behaviour of the network, in the case of $|y_i - \hat{y}_i| > 1$ L2 Loss has a greater values, which will cause the optimizer to change the networks parameters more drastically than in the case of L1. This can be an advantage or disadvantage depending on what you are interested in achieving and the scale of output data of your network. Another advantage of MSE loss is the tangent and differentiable behaviour around 0. By having the gradient approach 0 near the origin this loss functions ensures a 'homing in onto the minima' effect, which makes it harder for the optimizer to overshoot the minima.

Log-Cosh Loss

Another popular loss function is the Log-Cosh Loss Function. It combines the low gradient near 0 advantage of L2 Loss with linear loss behaviour of L1 Loss.

$$LC = \frac{\sum_i^n log(cosh(y_i - \hat{y}_i))}{n} \tag{8}$$

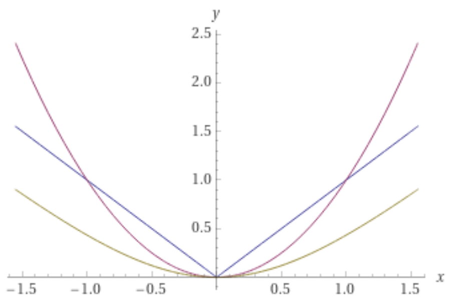


Figure 6: A comparison of L1, L2 and Log-Cosh Loss.

TODO: There are many more loss functions to chose from.

4.5 Performance Metrics

TODO:

4.6 Overfitting and Early Stopping

Following the loss gradient 'downhill' does indeed reduce the loss observed each iteration. However since the loss does get smaller and smaller each step, how would one know when to stop the training? One would easily believe that the lower our observed loss the better - so just keep on training, right?

Overfitting

Logically when training of the training dataset for a long time the network becomes better and better on it. Which means that its accuracy increases as the loss decreases over time. But this does indeed introduce a problem - Overfitting. The process of a network becoming better on a certain dataset (usually the training dataset, which is picked to be a subset of a larger set of data) while becoming worse on another subset of the whole dataset is known as Overfitting. It describes the network becoming overly specific to the details of the training dataset, while losing the vagueness of generalizing to data as a whole.

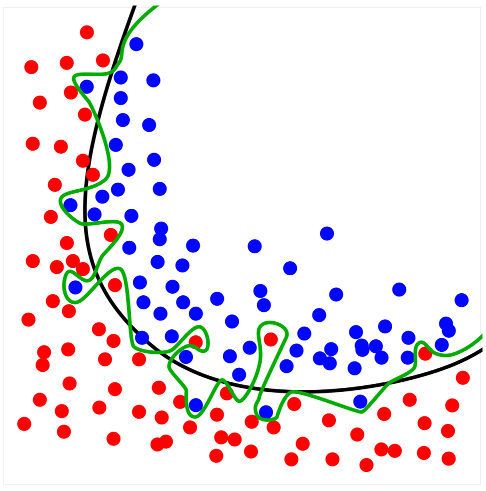


Figure 7: A graphical representation of Overfitting. In this case the network adjusts to the details of the noisy training data (red and blue points) instead of picking up on the general black division of the colors.

Early Stopping

To combat this problem of not knowing when to stop iterative training in order to suppress Overfitting, the logic of Early Stopping is often applied. As its name suggests Early Stopping stops the training loop at a specific point, at which it deems the network to generalize best. As vaguely introduces in the previous section, the whole dataset that is available to the user is usually split into three parts: Training, Validation and Test Datasets.

The Training Dataset is fed to the network in order to calculate gradients to descent. The Validation Dataset is only fed to the network to evaluate its performance, but never to train (calculate gradients). Finally the Test Dataset is used to evaluate the networks final performance, it is used seperately from the Validation Dataset in order to suppress any influence of it being used as the indicator of when to stop training.

Early Stopping works in the following way: After each Epoch (the network has seen all of the Training Data once) it calculates and logs the networks loss on the Validation Dataset. After a certain number of Epochs, this loss is guaranteed to incease again due to Overfitting, after first descresing together with the Training Datasets loss. When this happens Early Stopping waits for a set number of epochs (patience) before stopping the training and rolling the networks parameters back to the state it was when the Validation Loss was minimal.

4.7 Graph Neural Networks

Building heavily on the basis of 'simple' NNs, Graph Neural Networks expand heavily on the concept of what shapes are in. Edges can be bidirectional, which means that the logic from before when the previous layer incluences the next, is not valid anymore. Instead two or more nodes can influence another through the bidirectional edges. Also the number of connections (edges) to each node can vary based off the connection logic (e.g. k nearest neightbours) and does not necessarily have to be constant over all nodes.

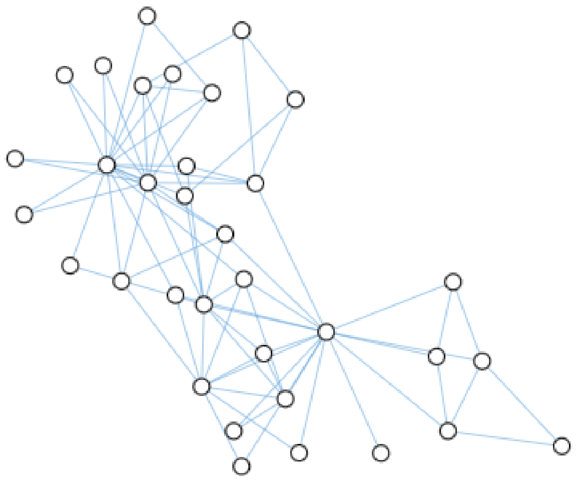


Figure 8: A graphical representation of a graph neural network.

The shape of GNNs (nodes and edges) are often generated from the data they are supposed to learn on, which makes them custom to every case. In contrast to the "one shape fits all" aproach of the networks discussed above, this allows them to adjust much better to certain scenarios, especially if the size of the data in question is of different scales (e.g. the average persons vs. my social circle). In GNNs the decision of which nodes are connected via edges is an important one, it heavily influences how the network behaves and provides another flexibility that often represents some reality about the data (e.g. social connections, molecular connections).

GNNs have proven great performance in many field, such as social modeling at e.g. Facebook, interpretation and classificatons of point clouds objects, as well as protein folding predictions.

5 GraphNeT

GraphNeT is a machine learning library aimed towards making applying Graph Neural Network to data from IceCube easy. It is based off PyTorch, one of the most popular open source machine learning frameworks. Its modular approach, as well as implementing the developer interface in Python, but implementing the actual heavy lifting code in C and C++, it allows for a great development experience, while still being wildly performant. GraphNeT uses PyTorch as a basis and inplements modules and adapters which interface with the IceCube data. Because PyTorch itself does not implement graphs, an additional library PyTorch Geometric is used to generate and learn on graphs which are generated by GraphNeT from the IceCube data.

TODO: Description of the model...

6 IceCube Data

TODO: Data from simulation description

7 Icedata

As one could expect the properties of ice is not constant across the whole cubic kilometer of IceCube. The optical properties vary noticably due to changing densities of the ice, but also because of impurities like dust. In figure 2 a graph visualizing the absorption and scattering is shown, giving an overview of how the ice changes vertically.

The project SPICE measured the ice properties (icedata) at 6 strings across IceCube, measurements have been taken every 10 m vertically, which will allow us to interpolate the values across the whole detector. Most interestingly the ice has mostly constant properties horizontally, but the ice layers are actually slightly tilted. The tilt actually not constant across depth, therefore the measurements have to be offset by the layer tilt at the point of interest.

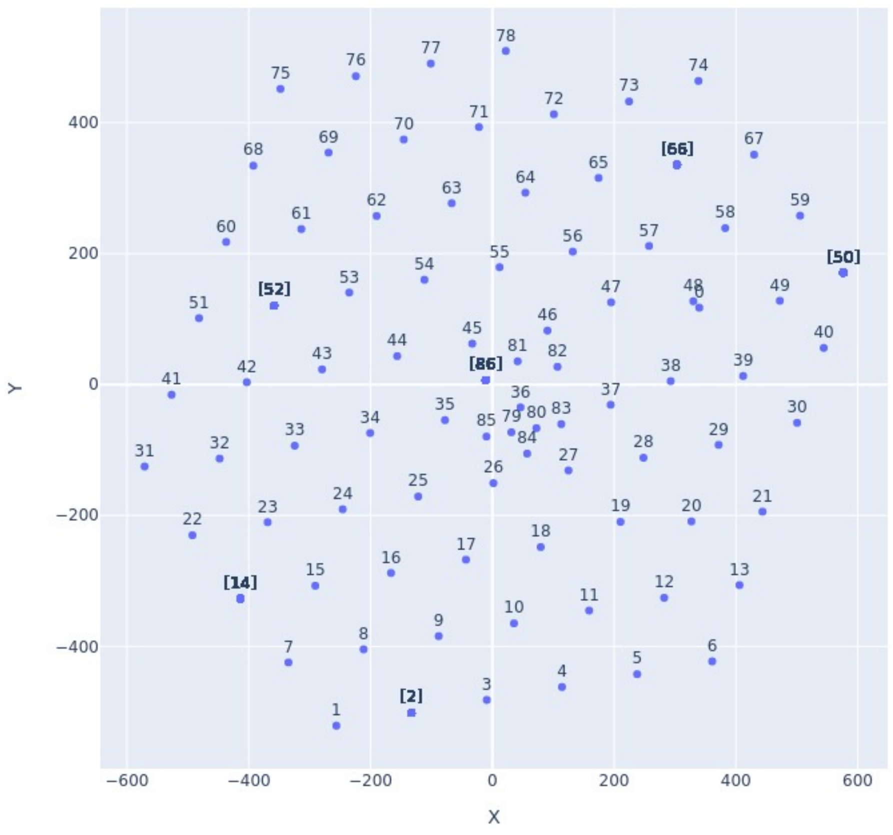


Figure 9: An overview of where SPICE measurements have been taken. String where measurements have been made are marked with square brackets.

The SPICE data comes as a directory of many files, only some of which will be of interest. The dataset is organized and labeled/keyed by depth, which is 0 at the surface of the ice and grows positive downwards. We will call this z-coordinate z_{Depth} or simply Depth.

Most of IceCube uses the IceCube coordinate system which has its origin 1948,07 m below the surface, its z-coordinate will be called $z_{IceCube}$ and grows positive upwards.

The z-coordinate we will be working with (and converting the other coordinates into) is z_{Global}, a coordinate system that has its origin at the surface and grows upwards. This definition results in the following conversions:

$$z_{Global} = -z_{Depth} \tag{9}$$

$$z_{Global} = z_{IceCube} - 1948{,}07\,\mathrm{m} \tag{10}$$

7.1 Reading

The files of interest are all in the space-seperated-values format and are read by using the Python `pandas` library. For example, the file `tilt.par` is read like so:

```
    pd.read_csv(root.joinpath('tilt.par'), sep=' ', names=['string_no', 'dist_sw_225'])
```

icemodel.dat

This file holds the icedata per depth. This data can be understood to be accurate

Depth	be(400)	adust(400)	k1	k2	BFRA	BFRB

tilt.par

This file describes where the ice layer tilt offsets are measures at, this means it gives a horizontal (xy) description of where the offsets are located. `SW_255` is the horizontal distance in the SouthWest (225deg) direction with its origin located at string 86.

Due to the natural relationship between string number and its location the data is redundant and only `SW_255` will be used from now on.

String	SW_225
50	-531.419
66	-454.882
86	0
52	165.202
2	445.477
14	520.77

tilt.dat

This file describes how the layers of ice are tilted relative to the flat assumption at the locations specified in `tilt.par`. It holds the 6 offset values at all depths. Together with the locations from above, it will be used to build a model of tilted layers.

Depth	Offset 50	Offset 66	Offset 86	Offset 52	Offset 2	Offset 14

7.2 Interpolation

Overview: Layer heights

Now that we know what data is available and useful to us, we can start to think about how we can interpolate this data in order to compute the ice properties at any given point. First a function/algorithm is required to decide on what layer a given point is located in, or more accurately, what layers a given point is located between. To build such a function, for each layer we interpolate the positions where offsets were measured SW_225 with the offsets *dzs* themselves. By adding/subtracting the base layer depth to the offset we get the actual layer heights at each point.

The described functionality is performed for all layers, and the function `interpolation_zs(sw)` returns a array containing the global height of all layers at `sw`.

```
interpolation_dzs = interp1d(
    df_tilt_par.dist_sw_225.to_numpy(),  # positions where offsets were measured (6,)
    df_tilt_dat.iloc[:, 1:].to_numpy(),  # offsets = dzs (125, 6)
    kind='quadratic',
    axis=1,
    bounds_error=False,
    fill_value='extrapolate',
)
def interpolation_zs(sw): return -(df_tilt_dat.z - interpolation_dzs(sw))
```

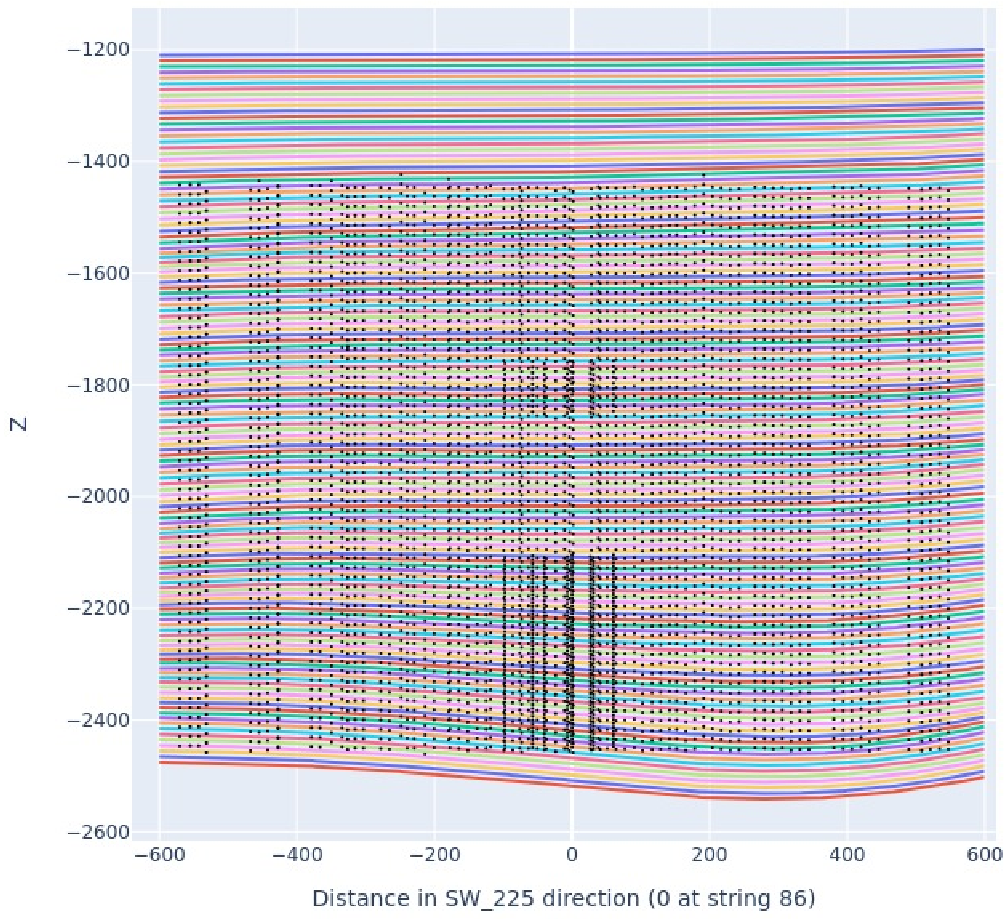


Figure 10: The interpolated ice layers (color) and all DOMs (black).

Overview: Ice properties

To now compute the icedata at any point we first execute the function designed above and get the heights of all layers at the location of the DOM. We then do another interpolation, this time interpolating the layers heights from before together with the icedata of all layers. This gives us a function that will interpolate the icedata between the layer above and below the point of interest.

```
interpolation_features = interp1d(
    layer_zs_global,  # layer heights from before (125,)
    layer_features,  # icedata from icemodel.dat (125, 6)
    kind='quadratic',
    axis=0,
    bounds_error=False,
    fill_value='extrapolate',
)
```

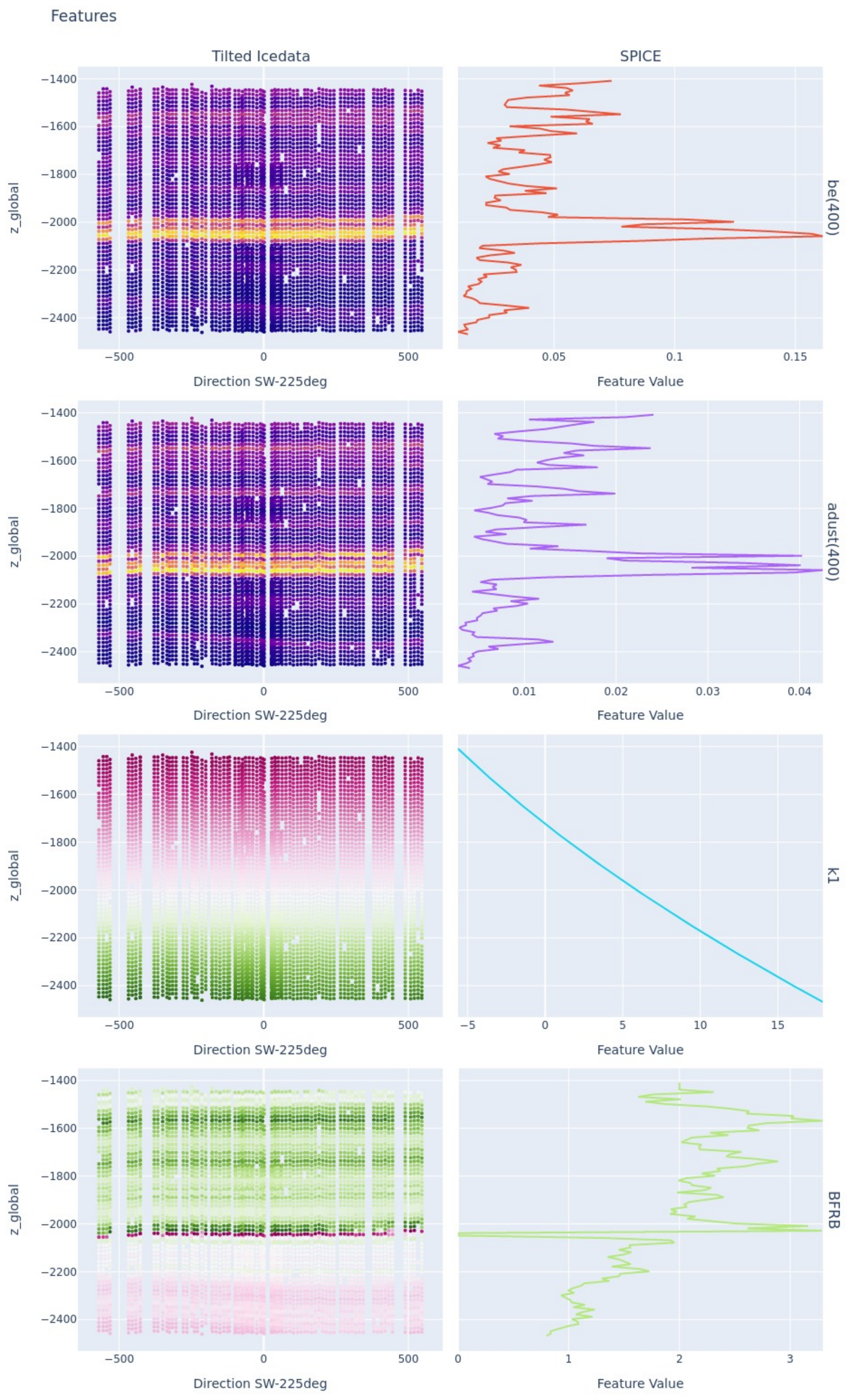


Figure 11: A visualisation of icedata that has been transformed/interpolated to all DOM locations.

8 Training

abc

9 Results

TODO: Results + Variation problem with each training... + Weighted loss over energy + new results

10 Conclusion

abc